SIEMENS

SIMATIC

S7
S7-200 SMART V3

System Manual

Preface

Product overview 1

Safety information 2

Industrial cybersecurity 3

Getting started 4

Installation 5

PLC concept 6

Programming concepts 7

PLC device configuration 8

Web server 9

Program instructions 10

Communication 11

Libraries 12

Debugging and troubleshooting 13

PID loops and tuning 14

Open loop motion control(PLCopen) 15

Open loop motion control(Legacy) 16

Technical specifications A

Calculating a power budget B

Error codes C

Special memory (SM) and system symbol names D

References E

Ordering information F

V3.0.1, 06/2025
A5E03822230-AN

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

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

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

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens Aktiengesellschaft. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens Aktiengesellschaft
Digital Industries
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E03822230-AN
Ⓟ 06/2025 Subject to change

Copyright © Siemens 2025.
All rights reserved

Preface

Purpose of the manual

The S7-200 SMART series is a line of micro-programmable logic controllers (Micro PLCs) that can control a variety of automation applications. Compact design, low cost, and a powerful instruction set make the S7-200 SMART a perfect solution for controlling small applications. The wide variety of S7-200 SMART models and the Windows-based programming tool give you the flexibility you need to solve your automation problems.

This manual provides information about installing and programming the S7-200 SMART CPUs and is designed for engineers, programmers, installers, and electricians who have a general knowledge of programmable logic controllers.

Required basic knowledge

To understand this manual, it is necessary to have a general knowledge of automation and programmable logic controllers.

Scope of the manual

This manual describes the following products:

- STEP 7-Micro/WIN SMART V3
- S7-200 SMART CPU firmware V3

For a complete list of the S7-200 SMART products and article numbers described in this manual, see Technical Specifications (Page 981).

Certification, CE, UKCA labels and other standards

Refer to the General specifications (Page 981) for more information.

ID link for the digital nameplate

The ID link is a globally unique identifier according to IEC 61406-1, which you will find as a QR code on your product.

You can recognize the ID link by the frame with a black frame corner at the bottom right. The ID link takes you to the digital nameplate of your product.

Scan the QR code on the product or on the packaging label with a smartphone camera, barcode scanner, or reader app. Access the ID link.

In the digital nameplate, you can find product data, manuals, declarations of conformity, certificates and other helpful information about your product.

Service and support

In addition to our documentation, we offer our technical expertise on the Internet on the customer support web site (http://www.siemens.com/automation/).

Contact your Siemens distributor or sales office for assistance in answering any technical questions, for training, or for ordering S7 products. Because your sales representatives are technically trained and have the most specific knowledge about your operations, process and industry, as well as about the individual Siemens products that you are using, they can provide the fastest and most efficient answers to any problems you might encounter.

Table of contents

Preface ... **3**

1 Product overview ... **23**

1.1 S7-200 SMART CPU ... 23

1.2 New features ... 24
1.2.1 What's new in S7-200 SMART V3.0.1? ... 24
1.2.2 What's new in S7-200 SMART V3.0? ... 25

1.3 S7-200 SMART expansion modules ... 27

1.4 HMI devices for S7-200 SMART ... 28

1.5 Communications options ... 29

1.6 Programming software ... 30

2 Safety information ... **31**

2.1 Warnings in this document ... 31

2.2 Safety-related symbols ... 31
2.2.1 Devices without explosion protection ... 31
2.2.2 Devices with explosion protection ... 32

2.3 Intended use ... 33

2.4 Changes to the device and spare parts ... 33

2.5 Knowledge and personnel qualification ... 34

2.6 Personal protective equipment ... 34

2.7 Open Source Software ... 34

2.8 Safe working ... 35
2.8.1 Working on electrical parts ... 35

2.9 Behavior in case of emergency ... 36

2.10 Residual risks ... 36
2.10.1 Residual risks ... 36
2.10.2 Live parts ... 36
2.10.3 External power supply to modules ... 37
2.10.4 Voltage dangerous to touch due to permissible electric potential difference ... 37
2.10.5 Conductive pollution ... 38
2.10.6 Overheating ... 38
2.10.7 Uncontrolled movements ... 38
2.10.8 Unsafe operating states ... 39

2.11 Material damage ... 39
2.11.1 Transport and storage ... 39
2.11.2 Installation and connection ... 39

3 Industrial cybersecurity ... 41

3.1 Cybersecurity information ... 41

3.2 Security note ... 41

3.3 Security overview ... 44

3.4 Network security ... 45

3.5 Communications portocols and port numbers used for Ethernet communication ... 46

3.6 CPU access protection ... 49

3.7 Configuring PLC security ... 49
3.7.1 PLC protection (Legacy) ... 50
3.7.1.1 Disable PLC Access Control ... 52
3.7.1.2 Enable PLC Protection ... 52
3.7.2 Enable User Management ... 53
3.7.3 Communicaiton setting ... 58

3.8 Configuring communication ... 59

3.9 Protect the project ... 62
3.9.1 Set Project Password ... 62
3.9.2 POU protection ... 63

3.10 Secure decommissioning ... 67
3.10.1 Securely removing data ... 67
3.10.2 Recycling and disposal ... 67

4 Getting started ... 69

4.1 Connecting to the CPU ... 69
4.1.1 Configuring the CPU for communication ... 70
4.1.1.1 Overview ... 70
4.1.1.2 Establishing the Ethernet hardware communication connection ... 71
4.1.1.3 Setting up Ethernet communication with the CPU ... 71
4.1.1.4 Establishing the RS485 hardware communication connection ... 74
4.1.1.5 Setting up RS485 communication with the CPU ... 74

4.2 Creating the sample program ... 77
4.2.1 Network 1: Starting the timer ... 78
4.2.2 Network 2: Turning the output on ... 79
4.2.3 Network 3: Resetting the timer ... 80
4.2.4 Setting the CPU type and version for your project ... 80
4.2.5 Saving the sample project ... 81

4.3 Downloading the sample program ... 82

4.4 Changing the operating mode of the CPU ... 82

5 Installation ... 85

5.1 Guidelines for installing S7-200 SMART devices ... 85

5.2 Power budget ... 87

5.3 Installation and removal procedures ... 89
5.3.1 Mounting dimensions for the S7-200 SMART devices ... 89
5.3.2 Installing and removing the CPU ... 90

5.3.3 Installing and removing a signal board or battery board ... 93
5.3.4 Removing and reinstalling the terminal block connector ... 95
5.3.5 Installing and removing an expansion module ... 97
5.3.6 Installing and removing the expansion cable ... 99

5.4 Wiring guidelines ... 100

6 PLC concept ... 107

6.1 Execution of the control logic ... 107
6.1.1 Reading the inputs and writing to the outputs ... 108
6.1.2 Executing the user program ... 109

6.2 Accessing data ... 111
6.2.1 Accessing memory areas ... 112
6.2.2 Format for Real numbers ... 119
6.2.3 Format for strings ... 119
6.2.4 Assigning a constant value for instructions ... 120
6.2.5 Addressing the local and expansion I/O ... 120
6.2.6 Using pointers for indirect addressing ... 121
6.2.7 Pointer examples ... 123

6.3 Saving and restoring data ... 124
6.3.1 Downloading project components ... 124
6.3.2 Uploading project components ... 127
6.3.3 Types of storage ... 128
6.3.4 Using a memory card ... 129
6.3.5 Inserting a memory card in a standard CPU ... 131
6.3.6 Transferring your program with a memory card ... 131
6.3.7 Restoring data after power on ... 136

6.4 Changing the operating mode of the CPU ... 136

6.5 Status LEDs ... 136

7 Programming concepts ... 139

7.1 Guidelines for designing a PLC system ... 139

7.2 Elements of the user program ... 141

7.3 Creating your user program ... 143
7.3.1 Earlier versions of STEP 7-Micro/WIN projects ... 143
7.3.2 Using STEP 7-Micro/WIN SMART V3 user interface ... 148
7.3.3 Using STEP 7-Micro/WIN SMART V3 to create your programs ... 149
7.3.4 Using wizards to help you create your control program ... 150
7.3.5 Features of the LAD editor ... 151
7.3.6 Features of the FBD editor ... 152
7.3.7 Features of the STL editor ... 152

7.4 Data block (DB) editor ... 154

7.5 Global variable table ... 156

7.6 Local variable table ... 159

7.7 UDT table ... 164
7.7.1 Overview ... 164
7.7.2 Using UDT table ... 166

7.8 PLC error reaction ... 169
7.8.1 Error handling in the S7-200 SMART CPU ... 169
7.8.2 System information ... 170
7.8.2.1 System ... 170
7.8.2.2 CPU ... 171
7.8.2.3 PROFINET device ... 172
7.8.3 PLC log ... 174
7.8.4 Scan rates ... 175
7.8.5 Non-fatal errors and I/O errors ... 175
7.8.6 Fatal errors ... 176

7.9 Program edit in RUN mode ... 177

7.10 Features for debugging your program ... 180

8 PLC device configuration ... 181

8.1 Configuring the operation of the PLC system ... 181
8.1.1 System block ... 181
8.1.2 Configuring communication ... 183
8.1.3 Configuring the digital inputs ... 186
8.1.4 Configuring the digital outputs ... 189
8.1.5 Configuring the retentive ranges ... 190
8.1.6 Configuring the startup options ... 192
8.1.7 Configuring the analog inputs ... 193
8.1.8 Reference to the analog inputs technical specifications ... 195
8.1.9 Configuring the analog outputs ... 196
8.1.10 Reference to the analog outputs technical specifications ... 197
8.1.11 Configuring the RTD analog inputs ... 198
8.1.12 Configuring the TC analog inputs ... 202
8.1.13 Configuring the RS485/RS232 CM01 communications signal board ... 205
8.1.14 Configuring the BA01 battery signal board ... 207
8.1.15 Clearing PLC memory ... 208
8.1.16 Creating a reset-to-factory-defaults memory card ... 210

8.2 High-speed I/O ... 210

9 Web server ... 213

9.1 Configuring the Web server with the Web server wizard ... 215
9.1.1 Activating the Web server ... 215
9.1.2 Configuring the watch tables on the Web server ... 216

9.2 Configuring the HTTPS certificates for the Web server with certificate management ... 218
9.2.1 External certificate ... 219
9.2.1.1 Manage the certificate authority ... 219
9.2.1.2 Downloading the certificate by external mode ... 221
9.2.2 Downloading the certificates by Internal certificate ... 224

9.3 Downloading and installing the certificate of CA ... 225
9.3.1 Get the certificate of CA ... 225
9.3.2 Installing the certificate for Windows ... 226
9.3.3 Installing the certificate for Mac OS ... 228
9.3.4 Installing the certificate for mobile devices ... 228

9.4 Accessing Web server from PC or mobile device ... 228

9.5 Standard Web pages ... 230
9.5.1 Layout of web pages ... 230
9.5.2 Introduction ... 231
9.5.3 Start page ... 231
9.5.4 Module information ... 233
9.5.5 Event log ... 233
9.5.6 Status chart ... 234
9.5.7 Watch table ... 234
9.5.8 Data log ... 235
9.5.9 Set clock ... 236

9.6 User defined Web pages ... 236
9.6.1 Downloading user-defined Web resource files ... 237

9.7 Web API (Application Programming Interface) ... 239

10 Program instructions ... 241

10.1 Bit logic ... 241
10.1.1 Standard inputs ... 241
10.1.2 Immediate inputs ... 242
10.1.3 Logic stack overview ... 243
10.1.4 STL logic stack instructions ... 245
10.1.5 NOT ... 247
10.1.6 Positive and negative transition detectors ... 248
10.1.7 Coils: output and output immediate instructions ... 249
10.1.8 Set, reset, set immediate, and reset immediate functions ... 249
10.1.9 Set and reset dominant bistable ... 251
10.1.10 NOP (No operation) instruction ... 252
10.1.11 Alternating output and alternating output on pulse ... 253
10.1.12 Bit logic input examples ... 253
10.1.13 Bit logic output examples ... 255

10.2 Clock ... 257
10.2.1 Read and set real-time clock ... 257
10.2.2 Read and set real-time clock extended ... 259

10.3 Communications ... 262
10.3.1 GET and PUT (Ethernet) ... 262
10.3.2 Transmit and receive (Freeport on RS485/RS232) ... 270
10.3.3 Get port address and set port address (PPI protocol on RS485/RS232) ... 282
10.3.4 Get IP address and set IP address (Ethernet) ... 283
10.3.5 Open user communication ... 285
10.3.5.1 OUC instructions ... 285
10.3.5.2 OUC instruction error codes ... 296

10.4 Compare ... 297
10.4.1 Compare number values ... 297
10.4.2 Compare character strings ... 300

10.5 Convert ... 302
10.5.1 Standard conversion instructions ... 302
10.5.2 ASCII character array conversion ... 305
10.5.3 Number value to ASCII string conversion ... 311
10.5.4 ASCII sub-string to number value conversion ... 315
10.5.5 Encode and decode ... 317

10.5.6 Scale ... 319

10.6 Counters ... 321
10.6.1 Counter instructions ... 321
10.6.2 High-speed counter instructions ... 325
10.6.3 High-speed counter summary ... 327
10.6.4 Noise reduction for high-speed inputs ... 327
10.6.5 High-speed counter programming ... 330
10.6.6 Example initialization sequences for high-speed counting mode ... 340
10.6.7 Example initialization sequences for high-speed frequency mode ... 347

10.7 Pulse output ... 351
10.7.1 Pulse output instruction (PLS) ... 351
10.7.2 Pulse train output (PTO) ... 352
10.7.3 Pulse width modulation (PWM) ... 354
10.7.4 Using SM locations to configure and control the PTO/PWM operation ... 355
10.7.5 Calculating the profile table values ... 358

10.8 Math ... 362
10.8.1 Add, subtract, multiply, and divide ... 362
10.8.2 Multiply integers to double integer and divide integer with remainder ... 366
10.8.3 Trigonometry, natural logarithm/exponential, and square root ... 368
10.8.4 Increment and decrement ... 370
10.8.5 Get the maximum or minimum value ... 371
10.8.6 Get a random value ... 373
10.8.7 Sort the array ... 374
10.8.8 Get the average of a array ... 375
10.8.9 Get absolute values ... 376
10.8.10 Low-pass filter ... 377

10.9 PID ... 378
10.9.1 PID loop instruction ... 378
10.9.2 Using the PID wizard ... 380
10.9.3 PID algorithm ... 386
10.9.4 Converting and normalizing the loop inputs ... 389
10.9.5 Converting the loop output to a scaled integer value ... 390
10.9.6 Forward- or reverse-acting loops ... 391
10.9.7 Integral anti-windup ... 393

10.10 Interrupt ... 393
10.10.1 Interrupt instructions ... 393
10.10.2 Interrupt routine overview and CPU model event support ... 395
10.10.3 Interrupt programming guidelines ... 397
10.10.4 Types of interrupt events that the S7-200 SMART CPU supports ... 399
10.10.5 Interrupt priority, queuing, and example program ... 400

10.11 Logical operations ... 406
10.11.1 Invert ... 406
10.11.2 AND, OR, XOR, NAND, NOR, and NXOR ... 407

10.12 Move ... 411
10.12.1 Move ... 411
10.12.2 Move byte, word, double word, or real ... 412
10.12.3 Block move ... 413
10.12.4 Swap bytes ... 414
10.12.5 Move byte immediate (read and write) ... 415

10.13 Program control ... 415
10.13.1 FOR-NEXT loop ... 415
10.13.2 JMP (jump to label) ... 417
10.13.3 SCR (sequence control relay) ... 418
10.13.4 END, STOP, and WDR (watchdog timer reset) ... 425
10.13.5 GET_ERROR (Get non-fatal error code) ... 426

10.14 Shift and rotate ... 428
10.14.1 Shift and rotate ... 428
10.14.2 Shift register bit ... 433

10.15 String ... 435
10.15.1 String (Get length, copy, and concatenate) ... 435
10.15.2 Copy substring from string ... 437
10.15.3 Find string and first character within string ... 438

10.16 Table ... 441
10.16.1 Add to table ... 441
10.16.2 First-in-first-out and last-in-first-out ... 442
10.16.3 Memory fill ... 444
10.16.4 Table find ... 445

10.17 Timers ... 449
10.17.1 Timer instructions ... 449
10.17.2 Timer programming tips and examples ... 451
10.17.3 Interval timers ... 457

10.18 Subroutine ... 458
10.18.1 CALL (subroutine) and RET (conditional return) ... 458
10.18.2 Subroutine call parameters and return examples ... 459

10.19 Function block ... 463
10.19.1 CALL (function block) and RET (conditional return) ... 463
10.19.2 Function block call parameters ... 465

10.20 PROFINET ... 470
10.20.1 Features of the programming instruction "PROFINET" ... 470
10.20.2 Read and Write data record ... 470
10.20.2.1 Input and output interface of RDREC and WRREC instruction ... 470
10.20.2.2 System-defined error code of the instructions RDREC and WRREC ... 472
10.20.3 Read and Write multiple bytes between physical PROFINET and memory address ... 474
10.20.3.1 Input and output interface of BLKMOV_BIR and BLKMOV_BIW ... 474
10.20.3.2 Error code of the instructions BLKMOV_BIR and BLKMOV_BIW ... 474

11 Communication ... 475

11.1 CPU communication connections ... 476

11.2 CPU communication ports ... 477

11.3 HMIs and communication drivers ... 477

11.4 Ethernet ... 479
11.4.1 Overview ... 479
11.4.2 Local/partner connection ... 479
11.4.3 Sample Ethernet network configurations ... 480
11.4.4 Assigning Internet Protocol (IP) addresses ... 481
11.4.4.1 Assigning IP addresses to programming and network devices ... 481

11.4.4.2 Configuring or changing an IP address for a CPU or device in your project ... 482
11.4.4.3 Searching for CPUs and devices on your Ethernet network ... 489
11.4.5 Locating the Ethernet (MAC) address on the CPU ... 490
11.4.6 HMI-to-CPU communication ... 491
11.4.7 Open user communication ... 492
11.4.7.1 Protocols ... 492
11.4.7.2 Connections ... 493
11.4.7.3 Ports and TSAPs ... 493
11.4.8 PROFINET Communication ... 495
11.4.8.1 Introduction ... 495
11.4.8.2 I-Device ... 496
11.4.8.3 Device database file in XML: GSDML ... 498
11.4.8.4 GSDML Management ... 499
11.4.8.5 PROFINET device naming ... 501
11.4.8.6 LED status indicators for PROFINET network ... 504
11.4.8.7 Configuring CPU as a controller ... 505
11.4.8.8 Configuring CPU as an I-Device without lower-level PROFINET IO system ... 513

11.5 PROFIBUS ... 530
11.5.1 EM DP01 PROFIBUS DP module ... 531
11.5.1.1 Distributed Peripheral (DP) standard communications ... 531
11.5.1.2 Using the EM DP01 to connect an S7-200 SMART as a DP device ... 532
11.5.1.3 Configuring the EM DP01 ... 534
11.5.1.4 Data consistency ... 534
11.5.1.5 Supported configurations ... 535
11.5.1.6 Installing the EM DP01 GSD file ... 536
11.5.1.7 Configuring the EM DP01 I/O ... 538
11.5.1.8 Example of V memory and I/O address area ... 540
11.5.1.9 User program considerations ... 541
11.5.1.10 LED status indicators for the EM DP01 PROFIBUS DP ... 543
11.5.1.11 Using HMIs and S7-CPUs with the EM DP01 ... 544
11.5.1.12 Device database file: GSD ... 545
11.5.1.13 PROFIBUS DP communications to a CPU example program ... 549
11.5.1.14 Reference to the EM DP01 PROFIBUS DP module technical specifications ... 551

11.6 RS485 ... 551
11.6.1 PPI protocol ... 551
11.6.2 Baud rate and network address ... 552
11.6.2.1 Definition of baud rate and network address ... 552
11.6.2.2 Setting the baud rate and network address for the S7-200 SMART CPU ... 553
11.6.3 Sample RS485 network configurations ... 555
11.6.3.1 Single-master PPI networks ... 555
11.6.3.2 Multi-master and multi-slave PPI networks ... 556
11.6.4 Assigning RS485 addresses ... 557
11.6.4.1 Configuring or changing an RS485 address for a CPU or device in your project ... 557
11.6.4.2 Searching for CPUs and devices on your RS485 network ... 561
11.6.5 Building your network ... 562
11.6.5.1 General guidelines ... 562
11.6.5.2 Determining the distances, transmission rates, and cable lengths for your network ... 562
11.6.5.3 Repeaters on the network ... 563
11.6.5.4 Specifications for RS485 cable ... 564
11.6.5.5 Connector pin assignments ... 564
11.6.5.6 Biasing and terminating the network cable ... 565

11.6.5.7 Biasing and terminating the CM01 signal board ... 566
11.6.5.8 Using HMI devices on your RS485 network ... 567
11.6.6 Freeport mode ... 568
11.6.6.1 Creating user-defined protocols with Freeport mode ... 568
11.6.6.2 Using the RS232/PPI Multi-Master cable and Freeport mode with RS232 devices ... 569

11.7 RS232 ... 570

12 Libraries ... 573

12.1 Library types (Siemens and user-defined) ... 573

12.2 Creating a user-defined library of instructions ... 575

12.3 Overview of Modbus communication ... 576
12.3.1 Modbus addressing ... 576
12.3.2 Modbus read and write functions ... 578

12.4 Modbus RTU library instructions ... 579
12.4.1 Modbus RTU communication overview ... 579
12.4.1.1 Modbus RTU library features ... 579
12.4.1.2 Requirements for using Modbus instructions ... 580
12.4.1.3 Initialization and execution time for Modbus protocol ... 582
12.4.2 Modbus RTU master ... 582
12.4.2.1 Using the Modbus RTU master instructions ... 582
12.4.2.2 MBUS_CTRL / MB_CTRL2 / MB_CTRL3 instruction (initialize master) ... 584
12.4.2.3 MBUS_MSG / MB_MSG2 / MB_MSG3 instruction ... 586
12.4.2.4 Modbus RTU master execution error codes ... 589
12.4.3 Modbus RTU slave ... 591
12.4.3.1 Using the Modbus RTU slave instructions ... 591
12.4.3.2 MBUS_INIT instruction (initialize slave) ... 592
12.4.3.3 MBUS_SLAVE instruction ... 594
12.4.3.4 Modbus RTU slave execution error codes ... 595
12.4.4 Modbus RTU master example program ... 595
12.4.5 Modbus RTU advanced user information ... 597

12.5 Modbus TCP library ... 599
12.5.1 Modbus TCP library features ... 599
12.5.2 Modbus TCP client ... 601
12.5.2.1 MBUS_CLIENT instruction ... 601
12.5.2.2 Modbus TCP client execution error codes ... 604
12.5.3 Modbus TCP server ... 605
12.5.3.1 MBUS_SERVER instruction ... 605
12.5.3.2 Modbus TCP server execution error codes ... 606
12.5.4 Example: Modbus TCP application ... 607
12.5.5 Modbus TCP advanced user information ... 612
12.5.6 Modbus TCP general exception codes ... 614
12.5.7 Modbus TCP general communication exception codes ... 615

12.6 Open user communication library ... 616
12.6.1 Overview ... 616
12.6.2 Parameters common to the OUC library instructions ... 617
12.6.3 Open user communication library instructions ... 619
12.6.3.1 TCP_CONNECT instruction ... 619
12.6.3.2 ISO_CONNECT instruction ... 622
12.6.3.3 UDP_CONNECT instruction ... 625

12.6.3.4 TCP_SEND instruction ... 627
12.6.3.5 TCP_RECV instruction ... 629
12.6.3.6 UDP_SEND instruction ... 632
12.6.3.7 UDP_RECV instruction ... 635
12.6.3.8 DISCONNECT instruction ... 637
12.6.4 Open user communication library instruction error codes ... 639
12.6.5 Open user communication library example ... 640
12.6.5.1 Active partner (client) ... 640
12.6.5.2 CheckErrors subroutine ... 649
12.6.5.3 Active partner variable table ... 650
12.6.5.4 Passive partner (server) ... 651
12.6.5.5 CheckErrors subroutine ... 658
12.6.5.6 Passive partner variable table ... 658

12.7 PN Read Write Record library ... 659
12.7.1 PN Read Write Record features ... 659
12.7.2 Input and output interface of PN Read Write Record library ... 659
12.7.3 Definition of parameters for input signal "STATUS" ... 660
12.7.4 System_defined error code of the library PN Read Write Record ... 661

12.8 USS Protocol library instructions ... 662
12.8.1 USS communication overview ... 662
12.8.1.1 USS protocol overview ... 662
12.8.1.2 Requirements for using the USS protocol ... 663
12.8.1.3 Calculating the time required for communicating with the drive ... 663
12.8.2 USS program instructions ... 664
12.8.2.1 Using the USS protocol instructions ... 664
12.8.2.2 USS_INIT instruction ... 665
12.8.2.3 USS_CTRL instruction ... 667
12.8.2.4 USS_RPM_x instruction ... 670
12.8.2.5 USS_WPM_x instruction ... 673
12.8.2.6 USS protocol example program ... 675
12.8.3 USS protocol execution error codes ... 677

12.9 SINAMICS Library ... 678
12.9.1 SINAMICS library features ... 678
12.9.2 SINA_POS instruction ... 679
12.9.2.1 Overview ... 679
12.9.2.2 Prerequisite of using the SINA_POS instruction ... 679
12.9.2.3 Input and output interface of SINA_POS instruction ... 683
12.9.2.4 Mode selection of SINAMICS with SINA_POS instruction ... 690
12.9.2.5 Relative positioning ... 691
12.9.2.6 Absolute positioning ... 694
12.9.2.7 Setup mode ... 699
12.9.2.8 Referencing (active referencing) ... 703
12.9.2.9 Referencing (set reference point) ... 706
12.9.2.10 Traversing blocks ... 709
12.9.2.11 Jog ... 712
12.9.2.12 Incremental jog ... 715
12.9.2.13 Encoder adjustment (for telegram 112 only) ... 719
12.9.3 SINA_SPEED instruction ... 720
12.9.3.1 Prerequisite of using the SINA_SPEED instruction ... 720
12.9.3.2 Input and output interface of SINA_SPEED instruction ... 723
12.9.3.3 Definition of "ConfigAxis" parameters ... 725

12.9.3.4 Example of SINA_SPEED instruction ... 725
12.9.4 SINA_PARA_S instruction ... 727
12.9.4.1 Prerequisite of using the SINA_PARA_S instruction ... 727
12.9.4.2 Input and output interface of SINA_PARA_S instruction ... 729
12.9.4.3 Example of the SINA_PARA_S instruction ... 734

13 Debugging and troubleshooting ... 737

13.1 Debugging your program ... 737
13.1.1 Bookmarks ... 737
13.1.2 Cross reference table ... 737

13.2 Displaying program status ... 739
13.2.1 Displaying status in the program editor ... 739
13.2.2 Configuring the STL status options ... 742

13.3 Using a status chart to monitor your program ... 742

13.4 Forcing specific values ... 744

13.5 Writing and forcing outputs in STOP mode ... 745

13.6 How to execute a limited number of scans ... 746

13.7 Hardware troubleshooting guide ... 748

14 PID loops and tuning ... 751

14.1 Pretuning ... 752

14.2 Fine tuning ... 754

14.3 PID loop definition table ... 755

14.4 Prerequisites ... 761

14.5 Auto-tune sequence ... 761

14.6 Exception conditions ... 763

14.7 Notes concerning PV out-of-range (result code 3) ... 764

14.8 PID tune control panel introduction ... 765

15 Open loop motion control(PLCopen) ... 769

15.1 State changes ... 771
15.1.1 Single axis state machine ... 771
15.1.2 Axes group state machine ... 773
15.1.3 Single-axis command buffered mode ... 774
15.1.4 Achieve multi-axis control with arrays ... 778

15.2 Adding and Configuring the motion control component ... 779
15.2.1 Adding a motion control component ... 779
15.2.2 Configuring the Virtual Axis ... 780
15.2.2.1 Virtual Axis Basic Setting ... 780
15.2.2.2 Mechanical and Limits ... 781
15.2.2.3 Parameters ... 783
15.2.2.4 Diagnose Information ... 785
15.2.3 Configuring the PTO Axis ... 786
15.2.3.1 PTO Axis Basic Settings ... 786
15.2.3.2 Mechanical and Limits ... 788

15.2.3.3 Parameters ... 790
15.2.3.4 Homing ... 793
15.2.3.5 Others ... 794
15.2.3.6 Diagnose Information ... 796
15.2.4 Configuring the Encoder Axis ... 796
15.2.4.1 Encoder Axis Basic Settings ... 796
15.2.4.2 Mechanical and Limits ... 799
15.2.4.3 Touch Probe ... 799
15.2.4.4 Diagnose Information ... 800
15.2.5 Configuring the Cam ... 800
15.2.5.1 Configuring the cam table ... 800
15.2.5.2 Set the parameters ... 803
15.2.5.3 View the curve ... 804
15.2.5.4 Import/Export cam table ... 805
15.2.6 Configuring Axes Group ... 807
15.2.6.1 Axes Group Basic Setting ... 807
15.2.6.2 Diagnose information ... 809

15.3 Commissioning with control panel(For PTO axis only) ... 809

15.4 Motion control instructions ... 813
15.4.1 Overview of variable ... 813
15.4.2 Motion control system variable ... 814
15.4.3 Basic Instructions ... 816
15.4.3.1 MC_Power ... 816
15.4.3.2 MC_TouchProbe ... 817
15.4.3.3 MC_AbortTrigger ... 818
15.4.3.4 MC_Reset ... 819
15.4.3.5 MC_Home ... 820
15.4.3.6 MC_Stop ... 822
15.4.3.7 MC_Halt ... 823
15.4.3.8 MC_MoveAbsolute ... 825
15.4.3.9 MC_MoveRelative ... 827
15.4.3.10 MC_MoveVelocity ... 829
15.4.3.11 MC_ImmediateStop ... 831
15.4.3.12 MC_MoveJog ... 832
15.4.4 Cam and Gear Instruction ... 835
15.4.4.1 MC_CamIn ... 835
15.4.4.2 MC_CamOut ... 843
15.4.4.3 MC_GearIn ... 844
15.4.4.4 MC_GearOut ... 846
15.4.4.5 MC_PhasingRelative ... 847
15.4.4.6 MC_GenerateCamTable ... 848
15.4.5 AxesGroup instructions ... 857
15.4.5.1 MC_GroupEnable ... 857
15.4.5.2 MC_GroupDisable ... 858
15.4.5.3 MC_GroupReset ... 859
15.4.5.4 MC_GroupStop ... 860
15.4.5.5 MC_MoveLinearAbsolute ... 861
15.4.5.6 MC_MoveLinearRelative ... 864

15.5 Error codes ... 866
15.5.1 Error code for single axis motion ... 866
15.5.2 Error code for axes group ... 869

15.5.3 Error codes for electronic cam and gear ... 869

15.6 Example programs ... 871
15.6.1 Single Axis example ... 871
15.6.2 Axes group example ... 874
15.6.3 Cam example ... 876

16 Open loop motion control(Legacy) ... 879

16.1 Using the PWM output ... 880
16.1.1 Using the PWM (pulse width modulation) output ... 880
16.1.2 Configuring the PWM output ... 881
16.1.3 PWMx_RUN subroutine ... 882

16.2 Using motion control ... 882
16.2.1 Maximum and start/stop speeds ... 883
16.2.2 Entering the acceleration and deceleration times ... 884
16.2.3 Configuring the motion profiles ... 884

16.3 Features of motion control ... 886

16.4 Using Axis of motion ... 888
16.4.1 Programming an Axis of Motion ... 888
16.4.2 Configuring an Axis of Motion ... 889
16.4.3 Subroutines created by the Motion wizard for the Axis of Motion ... 903
16.4.3.1 Guidelines for using the Motion subroutines ... 904
16.4.3.2 AXISx_CTRL subroutine ... 904
16.4.3.3 AXISx_MAN subroutine ... 906
16.4.3.4 AXISx_GOTO subroutine ... 907
16.4.3.5 AXISx_RUN subroutine ... 909
16.4.3.6 AXISx_RSEEK subroutine ... 910
16.4.3.7 AXISx_LDOFF subroutine ... 911
16.4.3.8 AXISx_LDPOS subroutine ... 912
16.4.3.9 AXISx_SRATE subroutine ... 913
16.4.3.10 AXISx_DIS subroutine ... 914
16.4.3.11 AXISx_CFG subroutine ... 914
16.4.3.12 AXISx_CACHE subroutine ... 915
16.4.3.13 AXISx_RDPOS subroutine ... 916
16.4.3.14 AXISx_ABSPOS subroutine ... 917
16.4.4 Using the AXISx_ABSPOS subroutine to read the absolute position from a SINAMICS servo drive ... 918
16.4.4.1 AXISx_ABSPOS and AXISx_LDPOS subroutines usage examples ... 919
16.4.4.2 Interconnections ... 920
16.4.4.3 Commissioning ... 921
16.4.4.4 Important facts to know ... 924
16.4.5 Axis of Motion example programs ... 925
16.4.5.1 Axis of Motion simple relative move (cut-to-length application) example ... 925
16.4.5.2 Axis of Motion AXISx_CTRL, AXISx_RUN, AXISx_SEEK, and AXISx_MAN example ... 927
16.4.6 Monitoring the Axis of Motion ... 932
16.4.6.1 Displaying and controlling the operation of the Axis of Motion ... 932
16.4.6.2 Displaying and modifying the configuration of the Axis of Motion ... 937
16.4.6.3 Displaying the profile configuration for the Axis of Motion ... 937
16.4.6.4 Error codes for the Axis of Motion (WORD at SMW620, SMW670, SMW720 or SMW770) ... 938
16.4.6.5 Error codes for the Axis of Motion instruction (seven LS bits of SMB634, SMB684, SMB734 or SMB784)) ... 939

16.4.7 Advanced topics ... 941
16.4.7.1 Understanding the configuration/profile table for the Axis of Motion ... 941
16.4.7.2 Special memory (SM) locations for the Axis of Motion ... 948
16.4.8 Understanding the RP Seek modes of the Axis of Motion ... 950
16.4.8.1 Selecting the work zone location to eliminate backlash ... 954

16.5 Using Axes group of Motion ... 955
16.5.1 Programming an Axes group of Motion ... 955
16.5.2 Configuring an Axes group of Motion ... 955
16.5.3 Subroutines created by the Motion wizard for the Axes group of Motion ... 962
16.5.3.1 Guidelines for using the Motion subroutines ... 963
16.5.3.2 GRPx_2/3D_MOVELINEAR subroutine ... 963
16.5.3.3 GRPx_MOVEPATH subroutine ... 965
16.5.3.4 GRPx_RESET subroutine ... 966
16.5.3.5 GRPx_STOP subroutine ... 968
16.5.4 Axes group of Motion example programs ... 969
16.5.4.1 Example of using GPRx_STOP subroutine to stop axes group movement ... 976

16.6 Stopping the Axes group movement ... 977

16.7 Error codes for the Axes group of Motion (WORD at SMW820 or SMW890) ... 978

16.8 Error codes for the Axes group of Motion instruction (seven LS bits of SMB823 or SMB893) ... 978

16.9 Interactions between Axis and Axis group ... 979

A Technical specifications ... 981

A.1 General specifications ... 981
A.1.1 General technical specifications ... 981

A.2 S7-200 SMART CPUs ... 986
A.2.1 CPU ST40 and CPU SR40 ... 986
A.2.1.1 General specifications and features ... 986
A.2.1.2 Digital inputs and outputs ... 990
A.2.1.3 Wiring diagrams ... 992
A.2.2 CPU ST60 and CPU SR60 ... 994
A.2.2.1 General specifications and features ... 994
A.2.2.2 Digital inputs and outputs ... 999
A.2.2.3 Wiring diagrams ... 1001
A.2.3 Wiring diagrams for sink and source input, and relay output ... 1003

A.3 Digital inputs and outputs expansion modules (EMs) ... 1004
A.3.1 EM DE08 and EM DE16 digital input specifications ... 1004
A.3.2 EM DT08, EM DR08, EM QR16, and EM QT16 digital output specifications ... 1006
A.3.3 EM DT16, EM DR16, EM DT32, and EM DR32 digital input/output specifications ... 1010

A.4 Analog inputs and outputs expansion modules (EMs) ... 1016
A.4.1 EM AE04 and EM AE08 analog input specifications ... 1016
A.4.2 EM AQ02 and EM AQ04 analog output module specifications ... 1019
A.4.3 EM AM03 and EM AM06 analog input/output module specifications ... 1021
A.4.4 Step response of the analog inputs ... 1025
A.4.5 Sample time and update times for the analog inputs ... 1025
A.4.6 Measurement ranges of the analog inputs for voltage and current (SB and EM) ... 1026
A.4.7 Measurement ranges of the analog outputs for voltage and current (SB and EM) ... 1027

A.5 Thermocouple and RTD expansion modules (EMs) ... 1028
A.5.1 Thermocouple expansion modules (EMs) ... 1028

A.5.1.1 EM AT04 thermocouple specifications 1028
A.5.2 RTD expansion modules (EMs) 1033

A.6 Digital signal boards 1038
A.6.1 SB DT04 digital input/output specifications 1038

A.7 Analog signal boards 1040
A.7.1 SB AE01 analog input specifications 1040
A.7.2 SB AQ01 analog output specifications 1043

A.8 RS485/RS232 signal boards 1044
A.8.1 SB RS485/RS232 specifications 1044

A.9 Battery board signal boards (SBs) 1046
A.9.1 SB BA01 Battery board 1046

A.10 EM DP01 PROFIBUS DP module 1048
A.10.1 S7-200 SMART CPUs that support the EM DP01 PROFIBUS DP module 1049
A.10.2 Connector pin assignments for EM DP01 1050
A.10.3 EM DP01 PROFIBUS DP module wiring diagram 1051

A.11 S7-200 SMART cables 1051
A.11.1 S7-200 SMART I/O expansion cable 1051
A.11.2 RS-232/PPI Multi-Master Cable and USB/PPI Multi-Master Cable 1052
A.11.2.1 Overview 1052
A.11.2.2 RS-232/PPI Multi-Master Cable 1053
A.11.2.3 USB/PPI Multi-Master Cable 1055

B Calculating a power budget 1057

B.1 Power budget 1057

B.2 Calculating a sample power requirement 1058

B.3 Calculating your power requirement 1059

C Error codes 1061

C.1 Timestamp mismatch 1061

C.2 PLC non-fatal error codes 1062

C.3 PLC non-fatal error SM flags 1064

C.4 PLC fatal error codes 1065

C.5 PROFINET alarm 1066

D Special memory (SM) and system symbol names 1067

D.1 SM (Special Memory) overview 1067

D.2 SMB0: System status 1069

D.3 SMB1: Instruction execution status 1070

D.4 SMB2: Freeport receive character 1070

D.5 SMB3: Freeport character error 1071

D.6 SMB4: Interrupt queue overflow, run-time program error, interrupts enabled, freeport transmitter idle, and value forced 1071

D.7 SMB5: I/O error status 1072

D.8 SMB6-SMB7: CPU ID, error status, and digital I/O points 1073
D.9 SMB8-SMB19, SMB131-SMB134: I/O module ID and errors..... 1073
D.10 SMW22-SMW26: Scan times 1075
D.11 SMB28-SMB29, SMB32-SMB33: Signal board ID and errors 1075
D.12 SMB30: (Port 0), SMB130: (Port1) and SMB31: (Port 2)..... 1076
D.13 SMB34-SMB35: Time intervals for timed interrupts 1078
D.14 SMB36-SMB45 (HSC0), SMB46-SMB55 (HSC1), SMB56-SMB65 (HSC2), SMB136-SMB145 (HSC3), SMB146-SMB155 (HSC4), SMB156-SMB165 (HSC5), SMB360-SMB369 (HSC6), SMB370-SMB379 (HSC7): high-speed counters 1078
D.15 SMB66-SMB85 (PTO0/PWM0, PTO1/PWM1), SMB166-SMB169 (PTO0), SMB176-SMB179 (PTO1), SMB566-SMB579 (PTO2/PWM2), SMB580-SMB593 (PTO3/PWM3), SMB2000-SMB2013 (PTO4/PWM4): high-speed outputs 1080
D.16 SMB86-SMB94, SMB186-SMB194 and SMB405-SMB413: Receive message control..... 1087
D.17 SMW98: Expansion I/O bus communication errors 1091
D.18 SMW100-SMW120 System alarms 1091
D.19 SMB195-SMB210: HSC Type and Frequency measurement sampling period 1092
D.20 SMB220-SMB235: PID control Error code 1093
D.21 SMB380-SMB404: HSC User-defined pin mapping enable 1093
D.22 SMB480-SMB515: Data log status 1094
D.23 SMB600-SMB799: Axis (0, 1, 2 and 3) open loop motion control (for Motion(V2.x)) 1094
D.24 SMB600-SMB649: Axis 0 open loop motion control 1096
D.25 SMB650-SMB699: Axis 1 open loop motion control 1096
D.26 SMB700-SMB749: Axis 2 open loop motion control 1097
D.27 SMB750-SMB799: Axis 3 open loop motion control 1097
D.28 SMB800-SMB858: Axes group 0 open loop motion control..... 1097
D.29 SMB870-SMB928: Axes group 1 open loop motion control..... 1098
D.30 SMB1000-SMB1049: CPU hardware/firmware ID 1099
D.31 SMB950-SMB998, SMB1050-SMB1099: SB (signal board) hardware/firmware ID 1099
D.32 SMB1100-SMB1399, SMB250-SMB349: EM (expansion module) hardware/firmware ID... 1100
D.33 SMB1400-SMB1799: EM (expansion module) module-specific data..... 1103
D.34 SMB1800-SMB1939: PROFINET device status..... 1104
D.35 SMB1940-SMB1946: Web server status 1104
E References 1107
E.1 Often-used special memory bits 1107
E.2 Interrupt events in priority order..... 1108
E.3 High-speed counter summary 1110

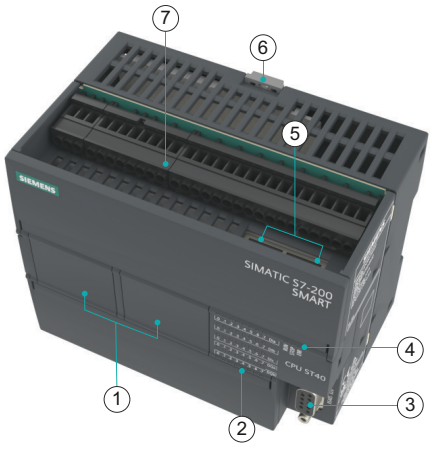
E.4 STL instructions 1110
E.5 Memory ranges and features 1117
F Ordering information 1121
F.1 CPU modules 1121
F.2 Expansion modules (EMs) and signal boards (SBs) 1121
F.3 Programming software 1122
F.4 Spare parts and other hardware 1122
Index 1125

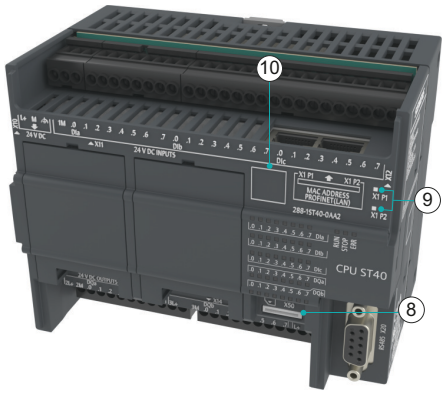
Product overview

1

1.1 S7-200 SMART CPU

The CPU combines a microprocessor, an integrated power supply, input circuits, and output circuits in a compact housing to create a powerful Micro PLC. After you have downloaded your program, the CPU contains the logic required to monitor and control the input and output devices in your application.





① Optional signal board
② LEDs for the I/O
③ RS485 Communication port
④ Status LEDs: RUN, STOP and ERROR
⑤ Ethernet communication port
⑥ Clip for installation on a standard (DIN) rail
⑦ Terminal connectors
⑧ Memory card reader (under door)
⑨ Ethernet status LEDs (under door): LINK, Rx/Tx
⑩ ID link (Page 3)

The CPU provides different models with a diversity of features and capabilities that help you create effective solutions for your varied applications. The different models of CPUs are shown below. For detailed information about a specific CPU, see the technical specifications (Page 986).

The S7-200 SMART V3 CPU family includes four CPU models. The second letter of the designator indicates AC power supply / relay outputs (R) or DC power supply / DC transistor (T). The number in the designator indicates the total onboard digital I/O count.

Table 1-1 S7-200 SMART CPUs

	SR40	ST40	SR60	ST60
Standard, expandable	X	X	X	X
Relay output	X		X	
Transistor output (DC)		X		X
I/O points (built-in)	40	40	60	60

Table 1-2 Standard expandable CPUs

Features		CPU SR40, CPU ST40	CPU SR60, CPU ST60
Dimensions: W x H x D (mm)		125 x 100 x 81	175 x 100 x 81
User memory	Program	80 Kbytes	100 Kbytes
	User data	80 Kbytes	100 Kbytes
	Retentive	40 Kbytes max.[1]	40 Kbytes max.[1]
	Memory for unbinded global variables and FB instances (DB2)	1 MB	1 MB
On-board digital I/O	• Inputs • Outputs	• 24 DI • 16 DQ	• 36 DI • 24 DQ
Expansion modules		8 max.	8 max.
Signal board		2	2
High-speed counters (8 total)	Single phase / Dual phase	8 at 200 kHz	8 at 200 kHz
	A/B phase	8 at 200 kHz 4x frequency support up to 800 kHZ	8 at 200 kHz 4x frequency support up to 800 kHZ
Pulse outputs [2]		5 at 200 kHz	5 at 200 kHz
PID loops		16	16
Real-time clock with 7-day back-up		Yes	Yes

[1] You can configure areas of V memory, M memory, C memory (current values), DB2 and portions of T memory (current values on retentive timers) to be retentive, up to the specified maximum amount.

[2] The specified maximum pulse frequency is possible only for CPU models with transistor outputs. Pulse output operation is not recommended for CPU models with relay outputs.

Refer to the technical specifications (Page 981) for the power requirements of the CPU and the expansion modules. Use the worksheets in Appendix B, Calculating a power budget (Page 1059) to calculate your power budget (Page 1057).

1.2 New features

1.2.1 What's new in S7-200 SMART V3.0.1?

STEP 7-Micro/WIN SMART V3.0.1 release provides the following new features:

New fuction for motion control

- Axes group support in motion control
  - Configuring Axes Group (Page 807)
  - Axes group state machine (Page 773)
  - Axes Group instructions (Page 857)
- New instruction for modifying cam table MC_GenerateCamTable (Page 848)

Software fault capture

Add SW fault capture for colleting faults after program crash.

Communication

After downloading PPI Fix IP, the changes are applied immediately without requiring a system power cycle (power off and power on).

1.2.2 What's new in S7-200 SMART V3.0?

Only the following CPU models with firmware V3.0 support the new features described in this chapter:

Table 1-3 CPU models affected by firmware V3.0

CPU model	Article number
CPU ST40, DC/DC/DC	6ES7288-1ST40-0AA2
CPU SR40, AC/DC/Relay	6ES7288-1SR40-0AA2
CPU ST60, DC/DC/DC	6ES7288-1ST60-0AA2
CPU SR60, AC/DC/Relay	6ES7288-1SR60-0AA2

Note

CPU models with firmware V3.0 cannot be downgraded to V2.8.3 or any previous versions.

STEP 7-Micro/WIN SMART V3.0 release provides the following new features:

Enlarged memory ranges

The CPU memory (Page 986) of CPU ST40, SR40, ST60 and SR60 increased.

Security

- Add user management and security configuration for accessing the PLC (Page 49)

Brand New Motion control

Brand new motion control (Page 769), Support variable programming and support for PLCopen standard instruction set.

Introducing brand new motion control:

- Strong support for PLCopen and industrial process objects
- PTO and PWM support 200 kHz frequency
- Support up to 16 controlled axes, including a maximum of 5 PTO axis, 2 encoder axis and 16 virtual axis.
- Add cam and gear functions

Programming

- DB2 user storage area is added based on the programmable variables
- Advanced variable programming functionality
- UDT and Array data type added
- Funtion block added

HSC enhancement

- Support up to 8 HSC (Page 325)
  - Unidirectional support 200 kHz
  - A/B phase support 200 kHz, 4 times frequency support to 800 kHz
- The pulse, direction, and reset signals corresponding to each HSC channel can be selected within the range of I0.0 ~ I1.7.

PTO enhancement

- Support up to five PTO
- Multi-segment PTO and PWM support 200 kHz frequency

Add new features to menus

- Open Log Folder
- Login
- Migrate
- Card encryption

Instruction

Move (Page 411)

1.3 S7-200 SMART expansion modules

To better solve your application requirements, the S7-200 SMART family includes a wide variety of expansion modules, signal boards, and a communications module. You can use these expansion modules with the standard CPU models (SR40, ST40, SR60 or ST60) to add additional functionality to the CPU. The following table provides a list of the expansion modules that are currently available. For detailed information about a specific module, see the technical specifications (Page 981).

Table 1-4 Expansion modules and signal boards

Type	Input only	Output only	Combination In/Out	Other
Digital expansion module	• 8 x DC In • 16 x DC In	• 8 x DC Out • 8 x Relay Out • 16 x Relay Out • 16 x DC Out	• 8 x DC In / 8 x DC Out • 8 x DC In / 8 x Relay Out • 16 x DC In / 16 x DC Out • 16 x DC In / 16 x Relay Out	
Analog expansion modules	• 4 x Analog In • 8 x Analog In • 2 x RTD In • 4 x RTD In • 4 x TC In	• 2 x Analog Out • 4 x Analog Out	• 4 x Analog In / 2 x Analog Out • 2 x Analog In / 1 x Analog Out	
Signal boards	• 1 x Analog In	• 1 x Analog Out	• 2 x DC In / 2 x DC Out	• RS485/RS232 • Battery Board

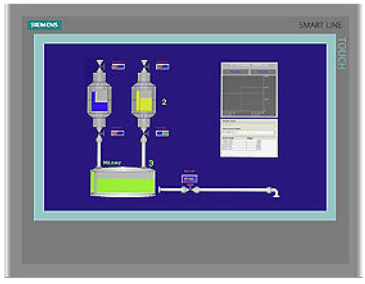
Table 1-5 Communication expansion modules

Module	Type	Description
Communication expansion module (EM)	PROFIBUS DP SMART module	EM DP01 PROFIBUS DP

1.4 HMI devices for S7-200 SMART

The S7-200 SMART supports Comfort HMIs, SMART HMIs, Basic HMIs and Micro HMIs. The TD400C and the SMART LINE Touch Panel are shown below. Refer to "HMIs and communication drivers" (Page 477) for a list of supported devices.

Table 1-6 HMI devices

	Text Display unit: The TD400C is an RS485-only display device that can be connected to the CPU. Using the Text Display wizard, you can easily program your CPU to display text messages and other data pertaining to your application. The TD400C device provides a low-cost interface to your application by allowing you to view, monitor, and change the process variables pertaining to your application.
	**SMART HMIs:** The SMART LINE Touch Panel provides operating and monitoring functions for small-scale machines and plants. Short configuration and commissioning times, their configuration in WinCC flexible (ASIA version), and a double-port Ethernet/RS485 interface form the highlights of these HMIs.

The Text Display wizard in STEP 7-MicroWIN SMART V3 helps you configure Text Display messages quickly and easily for the TD400C. To start the Text Display wizard, select the "Text Display" command from the "Tools" menu.

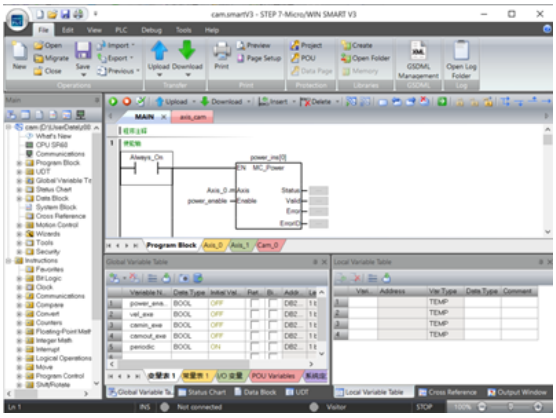
The SIMATIC Text Display (TD) User Manual can be downloaded from the Siemens customer support web site (http://www.siemens.com/automation/).

1.5 Communications options

The S7-200 SMART offers several types of communication between CPUs, programming devices, and HMIs:

- Ethernet:
  - Exchange of data from the programming device to the CPU
  - Exchange of data between HMIs and the CPU
  - S7 peer-to-peer communication with other S7-200 SMART CPUs
  - Open User Communication (OUC) with other Ethernet-capable devices
  - PROFINET communication with PROFINET devices
- PROFIBUS:
  - High speed communications for distributed I/O (up to 12 Mbps)
  - One bus master connects to many I/O devices (supports 126 addressable devices).
  - Exchange of data between the master and I/O devices
  - EM DP01 module is a PROFIBUS I/O device.
- RS485:
  - Provides a STEP 7-Micro/WIN SMART V3 connection for programming when using a USB-PPI cable
  - Supports a total of 126 addressable devices (32 devices per network segment)
  - Supports PPI (point-to-point interface) protocol
  - Exchange of data between HMIs and the CPU
  - Exchange of data between devices and the CPU using Freeport (XMT/RCV instructions)
- RS232:
  - Supports a point-to-point connection to one device
  - Supports PPI protocol
  - Exchange of data between HMIs and the CPU
  - Exchange of data between devices and the CPU using Freeport (XMT/RCV instructions)

1.6 Programming software



STEP7-Micro/WIN SMART V3 provides a user-friendly environment to develop, edit, and monitor the logic needed to control your application.

At the top is a quick access toolbar for frequent tasks, followed by menus for all common functions. At the left is the project tree and navigation bar for easy access to components and instructions. The program editor and other components that you open occupy the remainder of the user interface.

STEP7-Micro/WIN SMART V3 provides three program editors (LAD, FBD, and STL) for convenience and efficiency in developing the control program for your application.

To help you find the information you need, STEP7-Micro/WIN SMART V3 provides an extensive online help system.

Computer requirements

STEP 7-MicroWIN SMART V3 runs on a personal computer. Your computer should meet the following minimum requirements:

- Operating system: Windows 10 (both 32 bit and 64 bit versions) or Windows 11
- At least 500M bytes of free hard disk space
- Mouse (recommended)

Note

Windows patches requirement

The latest security patches from Microsoft should always be used!

Installing STEP 7-Micro/WIN SMART V3

Contact your Siemens distributor or sales office to download STEP7-Micro/WIN SMART V3 from the customer support web site (Page 4). Installation starts automatically and prompts you through the installation process. Refer to the Readme file for more information about installing STEP 7-Micro/WIN SMART V3.

Note

To install STEP 7-Micro/WIN SMART V3 on a Windows 10 or Windows 11 operating system, you must log in with Administrator privileges.

Safety information 2

2.1 Warnings in this document

You can find explanations of the warnings used in this document in the "Legal information" section.

2.2 Safety-related symbols

2.2.1 Devices without explosion protection

The following table contains an explanation of the symbols located in your SIMATIC device, its packaging or the accompanying documentation.

Symbol	Meaning
	General warning sign **Caution/Notice** You must read the product documentation. The product documentation contains information about the potential risks and enable you to recognize risks and implement countermeasures.
	Read the information provided by the product documentation. ISO 7010 M002
	Ensure the device is only installed by electrically skilled person. IEC 60417 No. 6182
CABLE SPEC.	Note that connected mains lines must be designed according to the expected minimum and maximum ambient temperature.
EMC	Note that the device must be constructed and connected in accordance with EMC regulations.
230V MODULES	Note that a 230 V device can be exposed to electrical voltages which can be dangerous. ANSI Z535.2
III 24V MODULES	Note that a device of Protection Class III may only be supplied with a protective low voltage according to the standard SELV/PELV. IEC 60417-1-5180 "Class III equipment"

Symbol	Meaning
INDOOR USE ONLY INDUSTRIAL USE ONLY	Be aware that the device is only approved for the industrial field and only for indoor use.
	Note that an enclosure is required for installing the device. Enclosures are considered: • Standing control cabinet • Serial control cabinet • Terminal boxes • Wall enclosure

2.2.2 Devices with explosion protection

The following table contains an explanation of the symbols located in your SIMATIC device, its packaging or the accompanying documentation.

Symbol	Meaning
ONLY EX MODULES	The assigned safety symbols apply to devices **with Ex approval**. You must read the product documentation. The product documentation contains information about the potential risks and enable you to recognize risks and implement countermeasures.
	Read the information provided by the product documentation. ISO 7010 M002
	Ensure the device is only installed by electrically skilled person. IEC 60417 No. 6182
F<2N DISPLAY F<4N HOUSING	Observe the mechanical rating of the device.
CABLE SPEC.	Note that connected mains lines must be designed according to the expected minimum and maximum ambient temperature.
EMC	Note that the device must be constructed and connected in accordance with EMC regulations.
U = 0V	When the device is under voltage, note that it may not be installed or removed, or plugged or pulled.
230V MODULES	Note that a 230 V device can be exposed to electrical voltages which can be dangerous. ANSI Z535.2

Symbol	Meaning
24V MODULES	Note that a device of Protection Class III may only be supplied with a protective low voltage according to the standard SELV/PELV. IEC 60417-1-5180 "Class III equipment"
INDOOR USE ONLY INDUSTRIAL USE ONLY	Be aware that the device is only approved for the industrial field and only for indoor use.
ZONE 2 INSIDE CABINET IP54	For Zone 2 potentially explosive atmospheres, be aware that the device may only be used when it is installed in an enclosure with a degree of protection ≥ IP54.
ZONE 22 INSIDE CABINET IP6x	For Zone 22 potentially explosive atmospheres, be aware that the device may only be used when it is installed in an enclosure with a degree of protection ≥ IP6x.

2.3 Intended use

The system is used to control machines and plants.

Intended use also includes observance of this documentation, in particular the safety instructions and conditions of use. See Technical specifications (Page 981) section.

WARNING

Download the document from ID link

This manual is delivered by electronic media. You can check and download the document from ID link.

2.4 Changes to the device and spare parts

Modifications to the device can impair the safety and function of the device:

- Do not make any changes or modifications to the device.
- Do not remove or paste over safety instructions on the device.
- Do not tape over, cover or install ventilation slots.
- Only use original spare parts and accessories.
- Only use the software provided by Siemens.

2.5 Knowledge and personnel qualification

All persons working with this device require the following knowledge:

- Contents of this document as well as contents of the enclosed documents.
- Handling the device (after instruction)
- Relevant standards and regulations
- Accident prevention regulations

The following activities are reserved for specially qualified personnel:

Working on electrical parts

Work on electrical parts may only be performed by the following persons:

- Qualified electricians
- Persons trained in electrical engineering under the direction and supervision of qualified personnel.

Commissioning and configuration

Commissioning and configuration requires general knowledge in the field of automation technology.

2.6 Personal protective equipment

The personal protective equipment depends on the activity and is determined by the operator.

To avoid material damage during packing, unpacking and installation activities we recommend:

- ESD wrist strap
- ESD shoes

2.7 Open Source Software

This product, solution or service ("Product") contains third-party software components. These are either open source software licensed under a license recognized by the Open Source Initiative (https://opensource.com/) or a license defined by Siemens as comparable ("OSS") and/or commercial software or freeware. With respect to the OSS components, the applicable OSS license conditions prevail over any other terms and conditions covering the Product. The OSS portions of this Product are provided royalty-free and can be used at no charge.

If SIEMENS has combined or linked certain components of the Product with/to OSS components licensed under the GNU LGPL version 2 or later as per the definition of the applicable license, and if use of the corresponding object file is not unrestricted ("LGPL Licensed Module", whereas the LGPL Licensed Module and the components that the LGPL Licensed Module is combined with or linked to is the "Combined Product"), the following additional rights apply, if the relevant LGPL license criteria are met: (i) you are entitled to modify the Combined Product for your own use, including but not limited to the right to modify the Combined Product to relink modified versions of the LGPL Licensed Module, and

(ii) you may reverse-engineer the Combined Product, but only to debug your modifications. The modification right does not include the right to distribute such modifications and you shall maintain in confidence any information resulting from such reverse-engineering of a Combined Product.

Certain OSS licenses require SIEMENS to make source code available, for example, the GNU General Public License, the GNU Lesser General Public License and the Mozilla Public License. If such licenses are applicable and this Product is not shipped with the required source code, a copy of this source code can be obtained by anyone in receipt of this information during the period required by the applicable OSS licenses by contacting the following address:

Siemens Aktiengesellschaft
LC DI FA SL
Werner-von-Siemens Str. 60
91052 Erlangen
Germany

Keyword: Open Source Request (please specify Product name and version, if applicable)

SIEMENS may charge a handling fee of up to 5 EUR to fulfil the request.

Warranty regarding further use of the Open Source Software

SIEMENS' warranty obligations are set forth in your agreement with SIEMENS. If you modify the product or the OSS components or use them in a way other than that specified by SIEMENS, the warranty will be voided and no technical support will be provided. The following license terms may contain limitations of liability that apply between you and the respective licensor. For the avoidance of doubt, SIEMENS does not make any warranty commitment on behalf of or binding upon any third-party licensor. The Open Source Software used in the product and the license agreements concerning this software can be found in the Readme_OSS.

2.8 Safe working

2.8.1 Working on electrical parts

- Only work on electrical parts if you are a qualified specialist (see sectionKnowledge and personnel qualification (Page 34)).
- Always observe the country-specific safety rules.
- Notify all those who will be affected by the procedure.
- Follow the 5 safety rules according to DIN EN 50110-1 (VDE 0105-1):

1. Disconnect
2. Secure to prevent reconnection
3. Verify safe isolation from the supply (all poles)
4. Ground and short-circuit
5. Erect barriers around or cover adjacent live parts

The 5 safety rules must be applied in the above order prior to starting work on an electrical system. After completing the work, proceed in the reverse order.

It is assumed that every electrician is familiar with the 5 rules.

2.9 Behavior in case of emergency

- Force EMERGENCY OFF.

When the safe operating state has been restored:

- Unlock the EMERGENCY OFF mechanism.
- The person responsible for the system ensures that the system starts up in a controlled and defined manner.

2.10 Residual risks

2.10.1 Residual risks

Despite all the technical and procedural risk reductions that have been carried out, not all dangers can be avoided.

The following sections describe these residual risks and measures to avoid them.

2.10.2 Live parts

Electrocution on contact with live parts leads to life-threatening injuries (death).

Operation

- Do not open the device.
- Never use damaged devices.
- Pull the plug and not the cable.

Installation and connection

- Only work on electrical parts if you are a qualified specialist (see section Knowledge and personnel qualification (Page 34)).
- Adhere to the protective measures for safe working on electrical parts (see section Working on electrical parts (Page 35))
- Fuse the connection cables according to the current-carrying capacity of the cable used, taking into account the applicable standards.

- Install the device in a control cabinet. The enclosures, cabinets or electrical equipment rooms must guarantee protection against electric shock and spread of fire.
- Ground the device in compliance with the applicable regulations.
- For the 24 V DC supply (SELV/PELV), only use power supply units that provide safe electrical extra-low voltage in accordance with IEC 61131-2 or IEC 61010-2-201.

Maintenance

- Only work on electrical parts if you are a qualified specialist (see section Knowledge and personnel qualification (Page 34)).
- Adhere to the protective measures for safe working on electrical parts (see section Working on electrical parts (Page 35))

2.10.3 External power supply to modules

Even if the main supply to the plant is disconnected, externally supplied modules can continue to carry voltage. An electric shock on contact with live parts can lead to death or serious injury.

For special SIMATICmodules, e.g. relay modules, you can connect or switch voltages larger than 24 V DC, e.g. 230 V AC. In some applications, these modules are supplied with voltages from outside the system (external power supply). In the event of an external power supply, a dangerous voltage may still be present on these modules even if the system is disconnected.

Maintenance

Ensure the following points before starting work:

- The plant is completely disconnected from the power supply.
- All externally supplied plant units are disconnected from the power supply

2.10.4 Voltage dangerous to touch due to permissible electric potential difference

Due to the permissible electric potential difference, a dangerous voltage may be present at the inputs of modules. An electric shock on contact with live parts can lead to death or serious injury.

Special SIMATIC modules, such as analog input modules, have isolated channels. In this case, there is no connection between the potential of the sensor and the potential of the module. If a dangerous voltage within the permissible electric potential difference occurs at the sensor, this voltage may also be present at the measurement channel of the module.

You can find the permissible electric potential differences in the technical specifications section of the modules' Equipment Manuals.

Maintenance

Ensure the following points before starting work:

- The plant is completely disconnected from the power supply.
- All externally supplied plant units are disconnected from the power supply

2.10.5 Conductive pollution

Electrocution during transmission of electrical energy via conductive contamination to the body leads to life-threatening injuries (death).

- Install the device in a control cabinet.
- Keep the control cabinet free of conductive contamination.

2.10.6 Overheating

Smoke development and fire due to overheating of the device and lines lead to burns and life-threatening injuries (death).

To avoid overheating:

- Ensure the correct installation position.
- Ensure sufficient air supply (for example, do not mask or cover ventilation slots, maintain mounting clearances).
- Only use undamaged cables.

Installation and connection

- Follow the instructions for the mounting position.
- Adhere to the specified ventilation clearances.
- Fuse the connecting cables according to the cable cross-section.

Maintenance

- Check plug-in connections and cables regularly for damage.

2.10.7 Uncontrolled movements

Uncontrolled movements of driven machine or system parts during commissioning, operation, maintenance and repair.

- Follow the safety instructions described in the function manuals.

2.10.8 Unsafe operating states

Unsafe operating states may result in personal injury of unknown extent.
The following factors can be triggers:

- Manipulation of the software, e.g. viruses, trojans or worms.

Manipulation of the software, e.g. viruses, trojans or worms.

- Adhere to the protective measures against tampering with the software (see section Industrial cybersecurity (Page 41)).
- Perform available updates in a timely manner.
- Protect files stored on removable media from malware with appropriate protective measures, e.g. virus scanner.
- Set up access protection for the CPU.

2.11 Material damage

2.11.1 Transport and storage

- Only pack, store, transport and ship electronic components, modules or devices in the original product packaging or in other suitable materials, e.g. conductive foam rubber or aluminum foil.
- Observe the limits during transport and storage. See Technical specifications (Page 981) section.

2.11.2 Installation and connection

- We recommend touching components, modules and devices only if they are grounded by one of the following measures:
  - With an ESD wrist strap.
  - With ESD shoes or ESD grounding strips in ESD areas with conductive flooring.
- Place electronic components, modules and devices only on conductive surfaces (e.g. tables with ESD coating, conducting ESD plastic foam, ESD packing bags, ESD transport containers).
- Ensure that there is sufficient overvoltage protection.
- Do not install/remove the following elements when the power is on:
  - Front connectors
  - I/O modules
  - CPU

Industrial cybersecurity 3

3.1 Cybersecurity information

Siemens provides products and solutions with industrial cybersecurity functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial cybersecurity concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial cybersecurity measures that may be implemented, please visit
https://www.siemens.com/cybersecurity-industry.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Cybersecurity RSS Feed under
https://new.siemens.com/cert.

3.2 Security note

Disclaimer

Please note that the following list of recommended risk-minimizing security measures is not intended to be exhaustive. Thus, please consult your security expert for final assessment and configuration. Further, as already mentioned in IndustrialSecurity (https://www.siemens.com/industrialsecurity).

Please note (i) that you are responsible for preventing unauthorized access to your plants, systems, machines, and networks, and (ii) that you should only connect such systems, machines, and components to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place. Since you are solely responsible for the conception, implementation, and maintenance of a holistic, state-of-the-art security concept to protect your enterprise, factories/plants, systems, machines, and networks (including the

products) against cyberthreats, you are liable for any damage caused by implementing no or insufficient security measures.

Data Protection

Siemens observes the data protection guidelines, especially the requirements regarding data minimization (privacy by design). This means the following for this product: The product does not process / save any personal information, but only technical functional data (e.g. time stamps). If the user links this data to other data (e.g. shift plans) or if the user saves personal information on the same medium (e.g. hard disk) and therefore creates a personal reference in the process, the user has to ensure meeting the guidelines regarding data protection.

Protection of the host PC

You are responsible for protecting your own host PCs and for preventing unauthorized access to their host PCs.

Siemens suggests taking the following measures to protect the host PC:

- Protect network access to the host PC with appropriate measures. If remote network connections to the host PC are required, Siemens recommends that you protect the communication to the PC with appropriate mechanisms, such as VPN connections.
- Enable the screen saver and lock the screen when leaving the computer.
- Install compatible antivirus software on the host PC. Make sure that your virus scanner and its database are always up-to-date.
- Install updates and patches for the operating system and software on the host PC to keep it up-to-date.
- In order to maintain an adequate security level, Siemens strongly recommends that you deactivate the unused interfaces, for example, USB ports or Ethernet.

Note

Note on using USB stick

- Only use dedicated USB stick
- Do a malware scan of the USB before using it
- Only plug in the USB stick when you use it

Notes on protecting administrator accounts

A user with administrator rights has extensive access and manipulation options available to the system.

Therefore, ensure there are adequate safeguards for protecting the administrator accounts in order to prevent unauthorized changes. To do this, use secure passwords and a standard user

account for normal operation. Other measures, such as the use of security policies, should be applied as needed.

NOTICE
Protection against access by unauthorized users An unauthorized user can operate the PC incorrectly. Operator actions by unauthorized Users jeopardize operational reliability. Restrict the rights for users by Windows user account.

Notes on the use

Note

To protect STEP 7-Micro/WIN SMART V3 and Smart Web Editor from any undesired manipulation when your PC suffers malicious attacks from the Internet, Siemens strongly recommends you to install a allow list tool on the PC. Then use the tool to manage the software installed on your PC.

Note

Ensure that you only use software installers that are obtained from official sources and are legally authorized, such as installation USB from an official sales channel or download from official Siemens website.

Note

To prevent password leakage, Siemens recommends you change the passwords used in the PLC periodically.

3.3 Security overview

Overview

This chapter describes the following security functions on S7-200 SMART:

Security function	Description
Network security	To protect your network communication, S7-200 SMART 3.0 (and later version) CPU uses HTTPS for the following connections: • Web API • Web browser If authentication, encryption, or integrity protection is required in the Network, Siemens recommends that you protect network and physical access to the S7-200 SMART devices with appropriate measures.
CPU access protection	The CPU protection methods can help you protect your CPU from unauthorized access. For more information, refer to Configuring PLC security (Page 49).
Communications protocols and port (Page 46)	This section provides an overview of the supported protocols and port numbers used for communication over PN/IE interfaces.
Activation and deactivation SNMP	You can enable the SNMP in Configuring communication (Page 59)
POU access protection	The POU protection method can help you protect the POU from being leaked. For more information, refer to POU protection (Page 63).
Project access protection	The project access protection method can help you protect your STEP 7-Micro/WIN SMART V3 project with password from being illegally accessed. For more information, refer to Set Project Password (Page 62).
Certificate	The certificate management wizard is used to configure the HTTPS certificates for the Web server. For more information, refer to Configuring the HTTPS certificates for the Web server with certificate management (Page 218)

Note

Once you changed the configuration in STEP 7-Micro/WIN SMART V3, the new configuration must be downloaded to the CPU 3.0 for the configuration to take effect.

WARNING

Protect the device against unauthorized access

An unauthorized user can operate the device incorrectly, read or write data and bypass logon by restarting the device.

Operation by unauthorized persons jeopardizes operational reliability.

You must protect these forms of communication by limiting physical access. Siemens recommend you lock the S7-200 SMART devices in a cabinet.

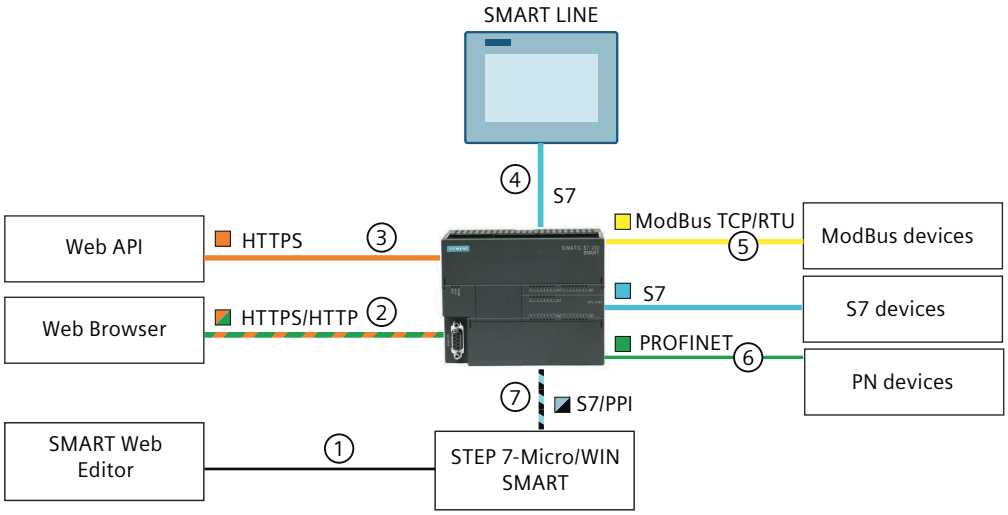
Note

The communication protocol for S7-200 SMART devices is designed for use in a trusted environment, and allows unauthenticated access to the devices. Siemens therefore strongly recommends protecting network access to the S7-200 SMART devices with appropriate mechanisms, for example, lock it in a cabinet.

3.4 Network security

With the enhanced network security of S7-200 SMART 3.0 (and later version) devices, you can access the S7-200 SMART CPU 3.0 (and later version) through a local area network or a remote area network.

Different devices/software connect to S7-200 SMART CPU with different channels.



Connec-tions	Supported APP/ Device	Port	Protocols	Remarks
①	SMART Web Editor			Virtual offline data transfer channel
②	Web Browser	TCP 443 TCP 80	HTTPS/HTTP	HTTP over TCP port 80 is used for the Open Source component claim only.
③	Web API	TCP 443	HTTPS	
④	SMART LINE S7 Device	TCP 102	S7	
⑤	Modbus devices	TCP 502	Modbus TCP/RTU	The default port is TCP 502, and you can set the port to use according to your actual business needs.
⑥	PN devices	UDP 34962/34963/34964	PROFINET	The UDP port for PROFINET is always open and cannot be disabled.

Connec-tions	Supported APP/ Device	Port	Protocols	Remarks
⑦	STEP 7-Micro/WIN SMART V3	TCP 102	S7	
		RS 485	PPI-cable	

In order to maintain an adequate security level, Siemens strongly recommends that you open the ports only at firewalls within the Secure Network. The table above lists all the port information for the S7200 SMART supported applications.

WARNING

Unauthorized access to the S7-200 SMART CPU through the Web server

Unauthorized access to the S7-200 SMART CPUs or changing S7-200 SMART variables to invalid values could disrupt process operation and could result in death, severe personal injury and/or property damage. Because enabling the Web server allows authorized users to perform operating mode changes, writes to S7-200 SMART data, and system time, Siemens recommends that you observe the following security practices:

- Protect PLC user IDs with a strong password. Keep the password secret and change it frequently.
- Perform error-checking and range-checking on your variables in your program logic because users can change PLC variables to invalid values.
- Use a secure Virtual Private Network (VPN) to connect to the SMART Web server when you use HTTP protocol.

WARNING

Unauthorized access to the S7-200 SMART through unsecure channel (TCP 502 for Modbus, TCP 102 for S7, TCP 80 for HTTP , UDP 34962/34963/34964 for PROFINET)

Unauthorized access to the S7-200 SMART devices or changing S7-200 SMART variables to invalid values could disrupt process operation and could result in severe personal injury and/or property damage.

Because enabling the unsecure port allows unauthorized users to perform operating mode changes, writes to S7-200 SMART data, and firmware updates, Siemens strongly recommends that you open the ports only at firewalls within the Secure Network.

3.5 Communications portocols and port numbers used for Ethernet communication

This section provides an overview of the supported protocols and port numbers used for communication over PN/IE interfaces. For each protocol the address parameters, the respective communications layer as well as the communications role and the communications direction are specified.

This information makes it possible to match the security measures for protection of the automation system to the used protocols (for example firewall). Because the security

measures are limited to Ethernet or PROFINET networks, the tables do not include PROFIBUS protocols.

Note

Port numbers used

The specified port numbers are the standard port numbers used by the S7-200 SMART CPU. Many communication protocols and implementations enable you to use other port numbers.

The following tables show the different layers and protocols used in the S7-200 SMART CPUs and in the S7-200 SMART communications modules.

Layers and protocols of the S7-200 SMART CPUs and software controllers (via PROFINET interface of the CPU)

The following table shows the protocols supported by S7-200 SMART CPUs. The S7-200 SMART software controllers also support the protocols listed in the following table for the Ethernet interfaces that are assigned to the software controller.

Table 3-1 Layers and protocols of the S7-200 SMART CPUs and Software Controllers (via PROFINET interface of the CPU)

Protocol / Role	Port number	(2) Link layer (4) Transport layer	Description / function	Default setting / notes
PROFINET protocols				
DCP	Not relevant	(2) Ethertype 0x8892 (PROFINET)	PROFINET Discovery and Basic Configuration Protocol. DCP determines PROFINET devices and enables the basic settings.	Default: Up to FW version V3.0 enabled.
LLDP	Not relevant	(2) Ethertype 0x88CC (LLDP)	PROFINET Link Layer Discovery Protocol. LLDP determines and manages neighborhood relations between PROFINET devices.	Default: Activated. Send function can be deactivated by Boundary "End of topology discovery" in the CPU properties; readiness to receive remains active. LLDP uses the special multicast MAC address: 01-80-C2-00-00-0E.
PROFINET IO data	Not relevant	(2) Ethertype 0x8892 (PROFINET)	PROFINET Cyclic IO Data Transfer With PROFINET IO telegrams, IO data is transferred cyclically between the PROFINET IO controller and IO devices via Ethernet.	Default: Deactivated. The protocol is only activated for PROFINET IO data traffic.
PROFINET Context Manager	34964	(4) UDP	PROFINET connection without RPC. Management of application and communication relationships between IO controller and IO devices.	Default: Enabled (UDP port open). This function cannot be deactivated.
Connection-oriented communications protocols				
HTTP Server	80	(4) TCP	Hypertext Transfer Protocol. HTTP is used for communication with the CPU-internal web server.	Default: activated.

Protocol / Role	Port number	(2) Link layer (4) Transport layer	Description / function	Default setting / notes
HTTPS Server	443	(4) TCP	Hypertext Transfer Protocol Secure. HTTPS is used for communication with the CPU-internal Web server via Secure Socket Layer (SSL).	Default: Deactivated. Can be enabled in the CPU properties. Requirement: Web server in the CPU properties is enabled.
MODBUS TCP Server / Client	502	(4) TCP	MODBUS Transmission Control Protocol. MODBUS/TCP is used by MB_CLIENT/ MB_SERVER instructions in the user program.	Default: Deactivated. Can be activated via Modbus instructions in the user program.
OUC[1]	1 ... 1999 can be used to limited extent[2]	(4) TCP (4) UDP (4) ISO-on-TCP (Port: 102)	Open User Communication (TCP/ UDP). OUC instructions enable connection establishment, connection termination and data transfer via the user program.	Default: Deactivated. You activate the respective protocol with the corresponding Open User Communication instruction in the user program or with the configuration of connections in the network view.
	2000 ... 5000 Recommended			
	5001 ... 49151 can be used to limited extent[2]			
SNMP Agent	161 162 (trap)	(4) UDP	Simple Network Management Protocol. SNMP is used for reading and setting of network management data (SNMP-managed objects) by the SNMP Manager.	Default: Deactivated. Can be activated by the configuration in the system block.
ISO-on-TCP Server	102	4) TCP	S7 protocol (according to RFC 1006). The S7 protocol uses ISO-on-TCP according to RFC 1006 for PG/HMI communication with STEP 7-Micro/WIN SMART V3.	Default: Activated. This function cannot be deactivated.
Put/Get Server	102	4) TCP	Put/Get Server, use S7 protocol to communicate with others Put/Get Client.	Default: Deactivated. You can enable Put/Get in the Communicaiton setting (Page 58). Once enabled, the new configuration must be downloaded to the PLC for the port to take effect.

[1] Note: OUC (open communication) provides direct access to the UDP and TCP protocols. You must take into consideration the IANA (Internet Assigned Numbers Authority) port restrictions and definitions.

[2] Do not use ports for OUC, which are already used by other protocols.

3.6 CPU access protection

Siemens strongly recommends you use password protection to prevent unauthorized operating of your CPU.

For more information about assigning and changing a CPU password, refer to Section "Configuring PLC security (Page 49)".

3.7 Configuring PLC security

To enhance the security and manage the password for your PLC, STEP 7-Micro/WIN SMART V3 provides the PLC Security Setting panel.

Opening the PLC Security Setting panel

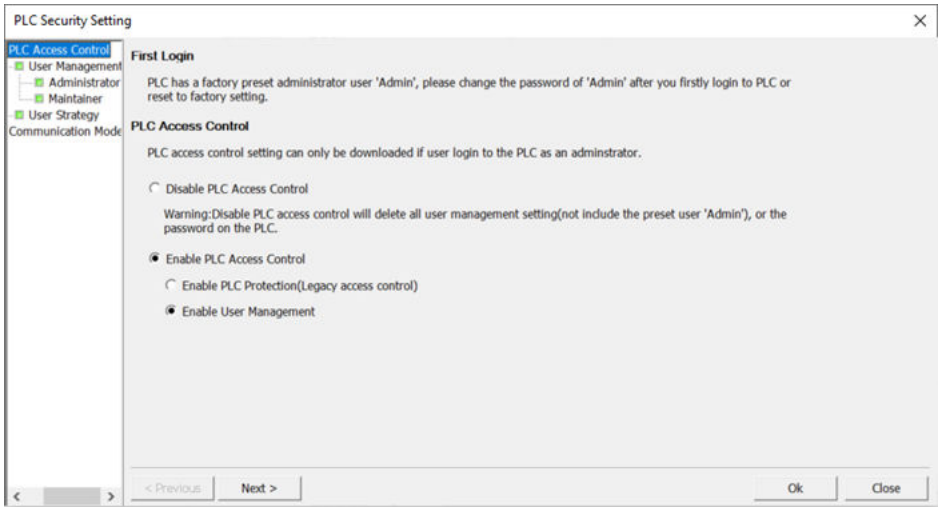
To open the PLC Security Setting panel, use one of the following methods:

- Click the "PLC Security Setting" button from the Security area of the Tools menu ribbon strip.

- Open the Security folder in the project tree (Page 149), select the "PLC Security Setting" node and press Enter; or double-click the "PLC Security Setting" node.

Setting the PLC Access Control

The PLC Security Setting panel offers below options for the user access control and the password management:



- Disable PLC Access Control
- Enable PLC Access Control
  - Enable PLC Protection
  - Enable User management

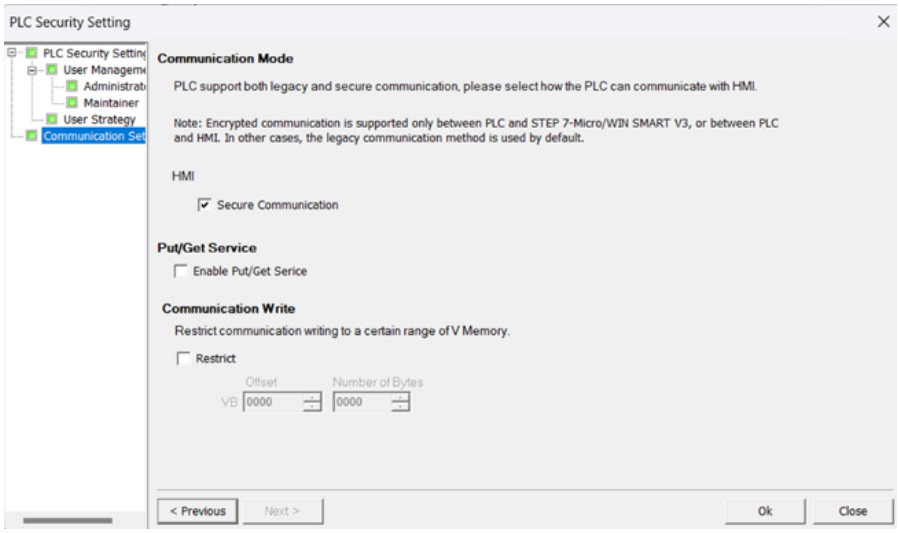
The default and recommended condition for the S7-200 SMART is "Enable User Management".

Setting communication mode

You can enable secure communication when the S7-200 SMART is communicating with STEP 7-Micro/WIN SMART V3 and HMI.

Note

The encrypted communication only works between PLC and STEP 7-Micro/WIN SMART V3, or between PLC and HMI.



3.7.1 PLC protection (Legacy)

Password-protected privilege levels

The CPU offers four levels of password protection, with "Disable PLC Access Control" (Level 1) providing unrestricted access and "Disallow Upload" (Level 4) providing the most restricted access.

A CPU password authorizes access to CPU functions and memory. With no CPU password downloaded ("Disable PLC Access Control" (Level 1)), the S7-200 SMART CPU allows

unrestricted access. If you have configured higher than "Disable PLC Access Control" (Level 1) access and downloaded a CPU password, the S7-200 SMART CPU requires password entry for access to CPU operations as defined in the table below.

The "Disallow Upload" (Level 4) password restriction protects the user program (your intellectual property) even if the password becomes known. You can never upload in Level 4 and can only change the privilege level if there is no user program present in the CPU. As a result, you can always protect your user program, even if someone discovers your password.

Table 3-2 S7-200 SMART CPU password-protected privilege levels

Description of operation	Disable PLC Access Control (Level 1)	Read privileges (Level 2)	Minimum privileges (Level 3)	Disallow upload (Level 4)
Read and write user data	Permitted	Permitted	Permitted	Permitted
Run, stop, and warm restart of the CPU	Permitted	Restricted	Restricted	Restricted
Read the time-of-day clock	Permitted	Permitted	Permitted	Permitted
Write the time-of-day clock	Permitted	Restricted	Restricted	Restricted
Upload the user program, data, and the CPU configuration	Permitted	Permitted	Restricted	Never Permitted
Download project	Permitted	Restricted	Restricted	Restricted
Reset to factory defaults	Permitted	Restricted	Restricted	Restricted
Delete project	Permitted	Restricted	Restricted	Restricted
Copy of program block, data block, or system data block to a memory card	Permitted	Restricted	Restricted	Restricted
Force and unforce	Permitted	Restricted	Restricted	Restricted
Execute single or multiple scan operations.	Permitted	Restricted	Restricted	Restricted
Writing of output in STOP mode.	Permitted	Restricted	Restricted	Restricted
Reset of scan rates in PLC information	Permitted	Restricted	Restricted	Restricted
Program status	Permitted	Permitted	Restricted	Never Permitted
Project compare	Permitted	Permitted	Restricted	Never Permitted

Serial ports mode changes and Time-of-Day (TOD) writes

You can allow CPU mode changes (go-to-RUN, go-to-STOP) and TOD writes through the serial ports (both the RS485 built-in and RS485/RS232 signal board if your CPU model supports it) without a password. To do so, select the "Allow" checkbox in the "Serial Ports" section.

This checkbox provides backward compatibility with older HMIs that do not prompt for a password for these functions. The following selections are available:

- If this box is checked and the CPU is password protected, then you can change operating modes and make TOD writes with these older HMIs.
- If this box is unchecked and the CPU is password protected, you cannot change operating modes or make TOD writes with these older HMIs.
- If the CPU is not password protected, you can change operating modes and make TOD writes with these older HMIs, regardless of the setting of the checkbox.

Accessing a password-protected CPU

Note

Always exit STEP 7-Micro/WIN SMART before disconnecting the cable to prevent another user from unauthorized access.

Entering the password over a network does not compromise the password protection for the S7-200 SMART CPU. If one authorized user is accessing restricted functions across a network, that does not authorize other users to access those functions. Only one user is allowed unrestricted access to the S7-200 SMART CPU at a time.

Disabling a password

You can disable the password by selecting "Disable PLC Access Control", since it allows all unrestricted CPU access.

Note

If the privilege level is at "Disallow Upload", you cannot download a new system block with a new password level if a valid user program exists. You must delete the user program first, and then you can download an updated system block.

What to do if you forget the password

If you forget your password, you must reset your PLC to factory defaults. (Refer to clear PLC memory for more information.)

3.7.1.1 Disable PLC Access Control

You can allow "Full Privileges" (Level 1) to S7-200 SMART from STEP 7-Micro/WIN SMART V3.

To disable the access control, follow these steps:

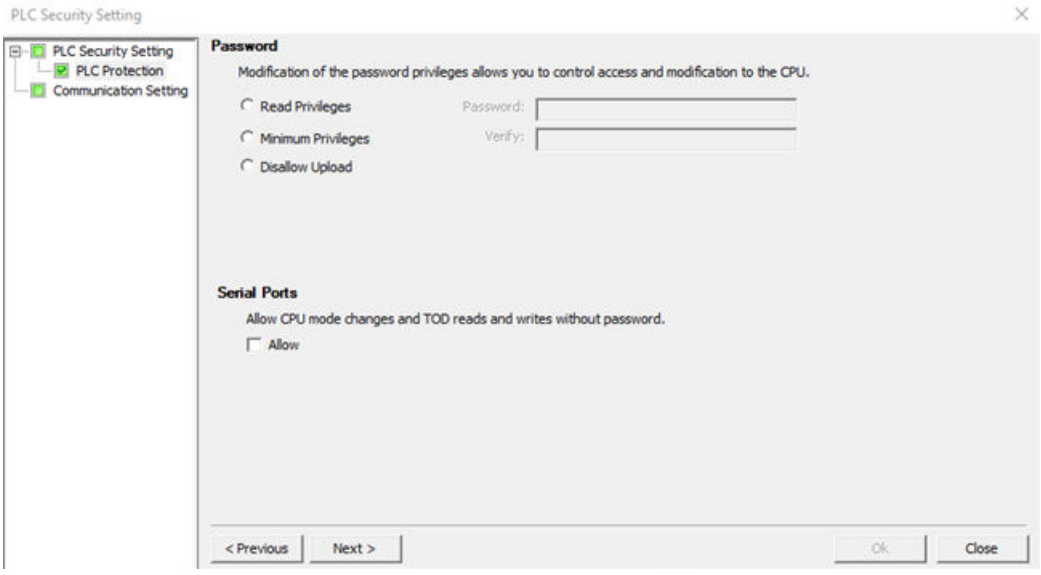
1. Select "Disable PLC Access Control" in the PLC Security Setting panel.
2. Click **Ok**.

3.7.1.2 Enable PLC Protection

You can configure the security settings and password for S7-200 SMART.

To enable the PLC protection, follow these steps:

1. Select "Enable PLC Access Control - Enable PLC Protection (legacy access control)" in the PLC Security Setting panel. And click **Next**.
2. Set PLC protection:
   - Select the Password-protected privilege levels and set the password.
   - Select "Allow" to allow the serial ports communication without a password. And click **Next**.



3. You can allow CPU mode changes (go-to-RUN, go-to-STOP) and TOD writes through the serial ports (both the RS485 built-in and RS485/RS232 signal board if your CPU model supports it) without a password. To do so, select the "Allow" checkbox in the "Serial Ports" section. This checkbox provides backward compatibility with older HMIs that do not prompt for a password for these functions. The following selections are available:
   - If this box is checked and the CPU is password protected, then you can change operating modes and make TOD writes with these older HMIs.
   - If this box is unchecked and the CPU is password protected, you cannot change operating modes or make TOD writes with these older HMIs.
   - If the CPU is not password protected, you can change operating modes and make TOD writes with these older HMIs, regardless of the setting of the checkbox.
4. Setting the communication mode (Page 58).

See also

Clearing PLC memory (Page 208)

3.7.2 Enable User Management

You can configure the user access settings for S7-200 SMART.

To enable the user management, follow these steps:

1. Select "Enable PLC Access Control - Enable User Management" in the PLC Security Setting panel. And click **Next**.
2. Check the "User Roles Description". And click **Next**.
3. Click **Add** or **Delete** to configure the administrator users. And click **Next**.
   Note: There is a default user called Admin in the table. The default password for this Admin is `Changeme_123`.
4. Click **Add** or **Delete** to configure the maintainer users. And click **Next**.
5. Configure the user strategy for both the administrator and the maintainer. And click **Next**.
6. Configure "Secure Communication" to secure the communication with HMI. And click **Save**.

User role privileges

User can login as administrator, maintainer or visitor. Each user role is granted with different privileges for operations.

Login as different user roles authorizes the access to CPU functions and memory. The S7-200 SMART requires you to login for access to the operations as defined in the table below.

Description of operation	Administrator	Maintainer	Visitor
Read user data	Permitted	Permitted	Permitted
Read PLC Information(PLC info and PLC log)	Permitted	Permitted	Permitted
Run, stop, and warm restart of the CPU	Permitted	Permitted	Restricted
Read the time-of-day clock	Permitted	Permitted	Permitted
Write the time-of-day clock	Permitted	Permitted	Restricted
Upload the user program, data, data log, and the CPU configuration	Permitted	Permitted	Restricted
Upload User Management (Security Configuration)	Permitted	Restricted	Restricted
Download project	Permitted	Restricted	Restricted
Download/upload user management block	Permitted	Restricted	Restricted
Download Transport Layer Security (TLS) certificate block	Permitted	Restricted	Restricted
Reset to factory defaults	Permitted	Restricted	Restricted
Delete of project	Permitted	Restricted	Restricted
Upgrade firmware, user-defined web resource	Permitted	Restricted	Restricted
Motion axis control	Permitted	Permitted	Restricted
Memory card	Permitted	Restricted	Restricted
Force and unforce	Permitted	Restricted	Restricted
Execute single or multiple scan operations.	Permitted	Permitted	Restricted
Reset of scan rates in PLC information	Permitted	Permitted	Restricted
Program status	Permitted	Permitted	Restricted
Project compare	Permitted	Permitted	Restricted
Read and write Web API & Status Chart	Permitted	Permitted	Restricted
Write user data(V.....)	Permitted	Permitted	Restricted

Description of operation	Administrator	Maintainer	Visitor
Change current user password	Permitted	Permitted	Restricted
Flash LED	Permitted	Permitted	Permitted

Administrator table

You can manage and configure the administrator access in the Administrator Table. There is a default user called Admin in the table.

- **Name:** double-click to set the name for this user role. User name must be unique.
- **Password:** double-click to set the password for this user role.
- **Access Terminal:** set as "ALL" because administrator users have the highest privileges.
- **Comments:** you can add comments for the user role. Not required.

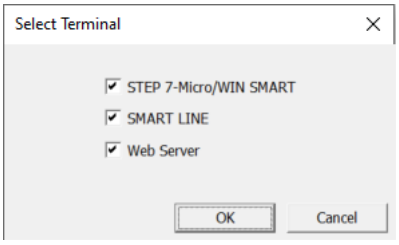
Note

- It is highly recommended that you change the password for the default user "Admin" after you receive the device.
- The user name must start with a letter, and can only contain letters, numbers and underscores. The length must be between 5 and 20 characters.
- The password can only contain digit, lowercase and uppercase character and special character at the same time. The length must be between 10 and 63 characters.

Maintainer table

You can manage and configure the maintainer access in the Maintainer Table. No maintainer is added by default.

- **Name:** double-click to set the name for this user role.
- **Password:** double-click to set the password for this user role.
- **Access Terminal:** select or deselect the access rights of the maintainer.



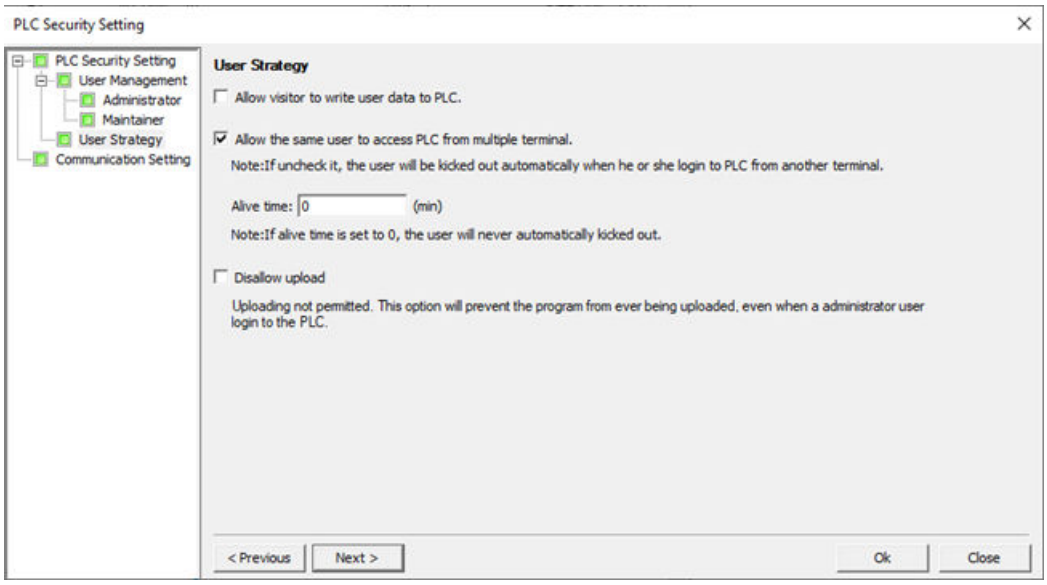
- **Comments:** you can add comments for the user role. Not required.

Note

- The user name must start with a letter, and can only contain letters, numbers and underscores. The length must be between 5 and 20 characters.
- The password can only contain digit, lowercase and uppercase character and special character at the same time. The length must be between 10 and 63 characters.

User strategy

You can configure the user access strategy and apply for both the administrator and the maintainer.



- **Allow visitor to write user data to PLC**: user can write data to PLC and modify system time even if the user is not logged in (visitor mode).
- **Allow the same user to access PLC from multiple terminal**: user can login to PLC from multiple terminals, such as STEP 7-Micro/WIN SMART V3, HMI or Web server.
  - If this option is not selected, the same user is only allowed to log in to one terminal. For example, if a user logs in to the STEP 7-Micro/WIN SMART V3 and then logs in to the Web server, the STEP 7-Micro/WIN SMART V3 will be automatically kicked off.
  - If this option is selected and if the Access item is set to ALL, the same user is allowed to connect to up to eight terminals.
  - "Active Time" is used to set how long the logged-in user can access the PLC, and when the timeout occurs, the logged-in user on the terminal will be kicked out of the PLC. When the "Alive time" mechanism is running independently on each terminal, the timer will only be reset when operating the PLC (the timer will not be affected by the program's online status).
- **Disallow upload**: prevent any user (administrator or maintainer) from uploading the data/program.

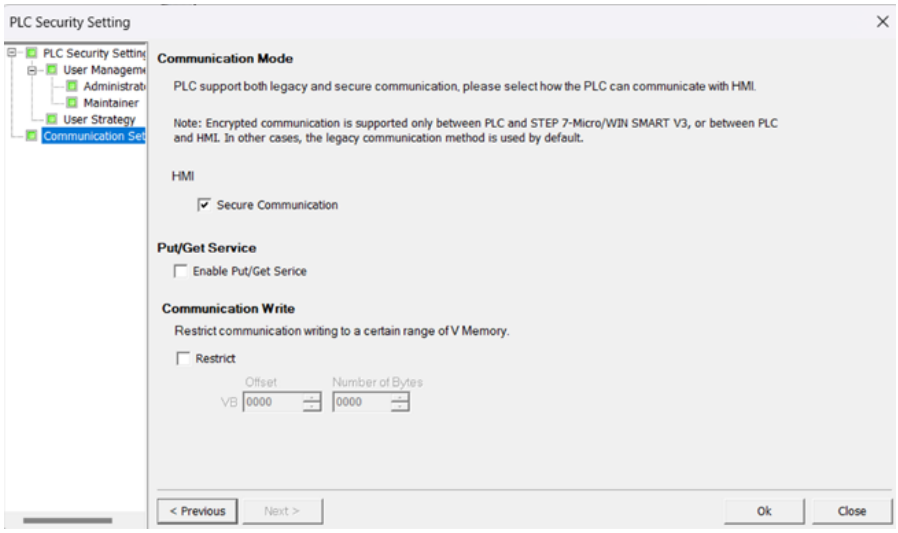
Communication setting

Setting the communication setting (Page 58)

3.7.3 Communicaiton setting

This page allows you to configure the security settings for the PLC to communicate with other devices.

- Select "Secure Communication" to secure the communication with STEP 7-Micro/Win SMART V3 and HMI.
- Select "Enable Put/Get Server" to enable the communication between S7-200 SMART CPUs through the Ethernet connection.
- Select "Restrict" to restrict communication write. And click **Save**.



Put/Get Server

You can enable or disable the PLC communication between S7-200 SMART CPUs through the Ethernet connection.

- When this PLC acts as a server, if this communication is enabled, it can be accessed by other clients; if this communication is disabled, other clients will not be able to access it.
- When this PLC acts as a client, this setting does not affect its operation.

The Put/Get Server is disabled by default.

Communication write restrictions

You can restrict communication writes to a specific range of V memory and disallow communication writes to other memory areas (I, Q, AQ, and M). To restrict communication writes to a specific range of V memory, select the "Restrict" checkbox, and configure the range in bytes of V memory.

This area can be as small as no bytes to as large as all V memory.

With this functionality, the user program can validate the data written into this subset of memory before using it in your application for even better security. Note that these restrictions pertain only to communication writes (for example, writes from HMIs,

STEP 7-Micro/WIN SMART, or PC Access and PUTs from other CPUs,), not writes from the user program.

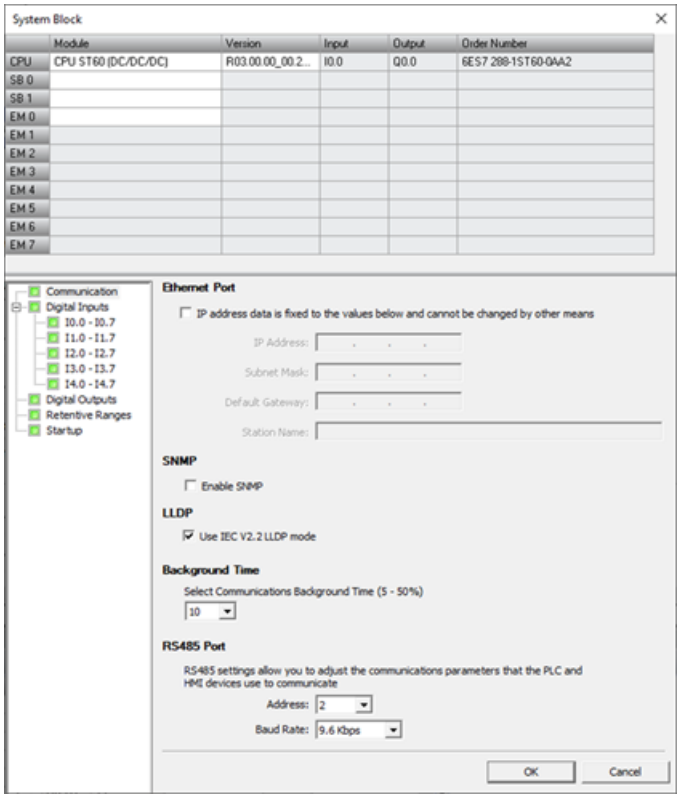
Note

If you restrict write access to a specific range of V memory, be sure that Text Display modules or HMIs only write within the writable range of V memory. Also, if you use the PID wizard, PID control panel, motion wizard, or motion control panel be sure that the V memory that the wizards or panels use are within the writable range of V memory.

With this restriction disabled, you can write to the full ranges of memory areas, including I, Q, M, V, and AQ.

3.8 Configuring communication

Click the Communication node of the system block dialog to configure the Ethernet port, SNMP, background time, and RS485 port.



Ethernet port

If you want your CPU to obtain its Ethernet network port information from the project, click the "IP address data is fixed to the values below and cannot be changed by other means" checkbox. You can then enter the following Ethernet network information:

- IP address: Each device must have an Internet Protocol (IP) address. The device uses this address to deliver data on a more complex, routed network.
- Subnet mask: A subnet is a logical grouping of connected network devices. Nodes on a subnet are usually located in close physical proximity to each other on a Local Area Network (LAN). The subnet mask defines the boundaries of an IP subnet. A subnet mask of 255.255.255.0 is generally suitable for a local network.
- Default gateway: Gateways (or IP routers) are the link between LANs. Using a gateway, a computer in a LAN can send messages to other networks, which might have other LANs behind them. If the destination of the data is not within the LAN, the gateway forwards the data to another network or group of networks where it can be delivered to its destination. Gateways rely on IP addresses to deliver and receive data packets.
- Station name: The station name is the name by which this CPU is identified on the network. Use a name that helps you identify the CPU on the Communications dialog.

Note

The station name follows the standard DNS (Domain Name System) naming conventions. The S7-200 SMART CPUs limit the station name to a maximum of 63 characters, which can consist of the lower case letters a through z, the digits 0 through 9, the hyphen character (minus sign) and the period character.

The CPU prohibits certain names:

- The station name must not have the format n.n.n.n where n is a value of 0 through 999.
- You cannot begin the station name with the string port-nnn or the string port-nnn-nnnnn where n is a digit 0 through 9. For example, port-123 and port-123-45678 are illegal station names. A station name cannot start or end with a hyphen or period.

SNMP

If you want to use SNMP to monitor and manage network devices, enable SNMP by ticking the box in front of 'Enable SNMP'.

Use IEC V2.2 LLDP mode

Default for devices which support IEC V2.3 mode:

- The option "Use IEC V2.2 LLDP" mode is disabled (no checkmark). The PROFINET interface is set to IECV2.3 mode (IEC 61158 V2.3).
- Select this option to set the PROFINET interface to IEC V2 2 mode. This may be necessary when you have split a system into several dependent projects (e.g., in case of an I-device configuration).

Default for devices which do not support IEC V2.3 mode:

- The option "Use IEC V2.2 LLDP mode" is selected (checkmark is set) and cannot be changed.
- If all devices with PROFINET interfaces are completely configured and the parameters assigned in the current project, keep the default setting.

Background time

You can configure the percentage of the scan cycle time that is dedicated to processing the communication requests. As you increase the percentage of time that is dedicated to processing communication requests, you are increasing scan time, which makes your control process run more slowly. The scan time only increases if there are communication requests to process.

The default percentage of the scan time dedicated to processing communication requests is set to 10%. This setting provides a reasonable compromise for processing compilation/ status operations, while minimizing the impact to your control process. You can adjust this value by 5% increments up to a maximum of 50%.

As you add more communication partners to the S7-200 SMART CPU, additional background time is required to handle the requests from those partners. GET and PUT instructions need additional resources to create and maintain connections to other devices. The EM DP01 PROFIBUS DP module requires additional communication background time if you have HMIs or other CPUs communicating with the S7-200 SMART CPU through the EM DP01. Open User Communication (OUC) also adds an additional load to the CPU and may require additional background time.

RS485 port

Use these settings to adjust the communication parameters for system protocols for the onboard RS485 port. The system protocols are used when connecting to a programming device or HMI devices:

- RS485 port address: Click the scroll buttons to enter the desired CPU address (1-126). The default port address is 2.
- Baud Rate: Choose the desired data baud rate from the dropdown list (9.6 Kbps, 19.2 Kbps, or 187.5 Kbps).

Note

You can make the following RS485 communication connections for S7-200 SMART CPUs:

- Use a USB-PPI cable to program all CPU models through any serial port, including the RS485 port, the signal board port, and the DP01 PROFIBUS port.
- Use the RS485 and RS232 ports for HMI access (Data read/write) and Freeport communications.

3.9 Protect the project

User programs in the CPU are encrypted and stored in the file system. They can be exported and verified using Import/Upload/Runtime after logging in with authorization. It is recommended that you store these files securely and be aware of security risks.

3.9.1 Set Project Password

To password protect the entire project, follow these steps:

1. Click the Project button from the Protection section of the File menu.

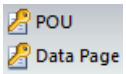
2. Select the "Password-protect this project" checkbox.
3. Enter and verify a password.

Setting password for the project will prevent unauthorized user from reading or writting any part of your STEP 7-Micro/WIN SMART V3 project.

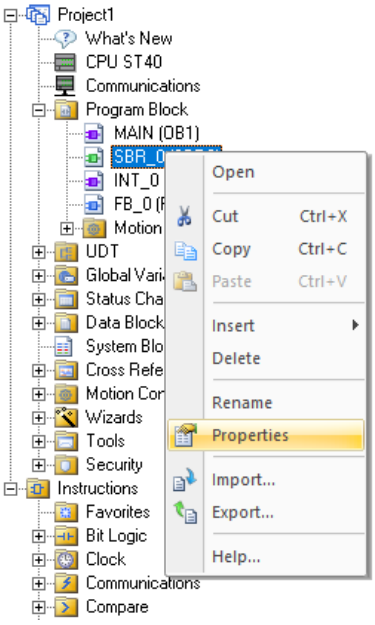
3.9.2 POU protection

To access the properties of a POU or a data block, follow one of these steps:

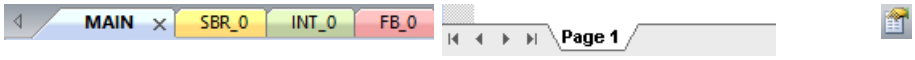
- Click the POU or the Data Page button from the Protection area of the View menu ribbon strip to display the properties of the POU currently in the program editor, or the page in the data block.



- Right-click a POU or data block in the project tree and select **Properties** from the context menu.

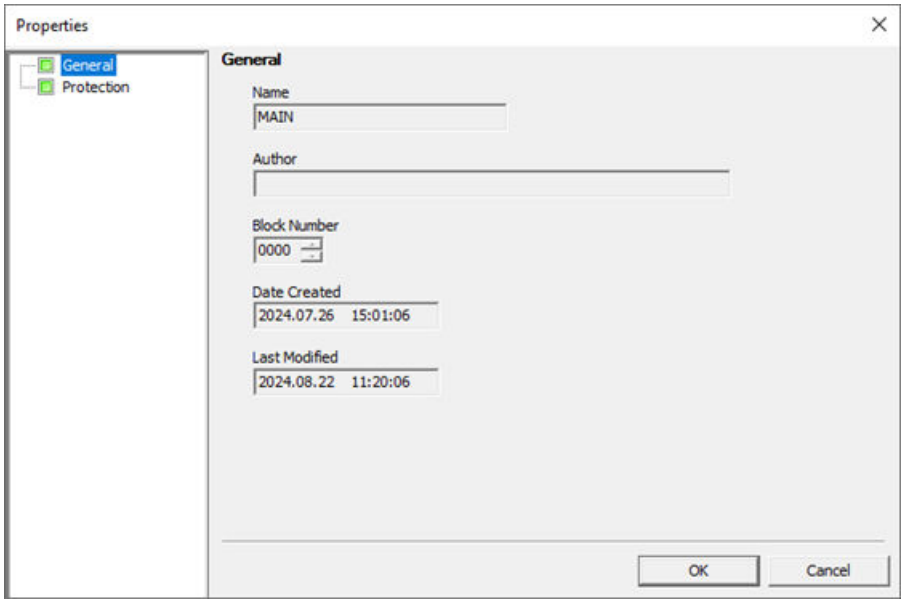


- Right-click a POU or data block page tab in the editor window and click the "Properties" button.



The POU Properties dialog has two pages:

- General
- Protection



POU properties general page

The General page allows you to renumber or rename the main program block, subroutines, interrupt routines, function block and data block pages. You can also assign an author to the project.

If you assign global variable names to the POUs or the data block, the global variable name is displayed in your program code even when you do not have Symbolic Addressing view enabled. The global variable table displays a "POU Variables" tab that lists the all of the global variable name assignments. You can only view this tab; you cannot edit the entries from the global variable table. To change an assignment, you must edit the Properties for the appropriate POU.

Note

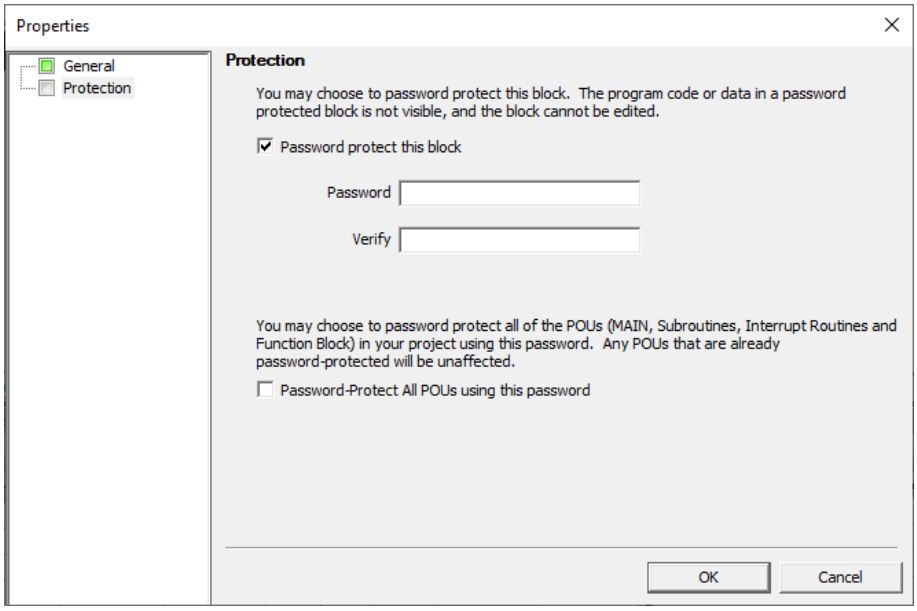
You cannot use the default name (the POU address assigned by STEP 7-Micro/WIN SMART V3, such as SBR_1 for a subroutine , a INT_1 for an interrupt routine or FB_1 for a function block) as the global variable name for another POU because that would constitute a duplicate assignment. If you violate the guidelines for making a global variable name assignment, STEP 7-Micro/WIN SMART V3 reports an error when you try to compile your program.

POU properties protection page

The Protection page allows you to password-protect a POU or data block, so that it is invisible to other users and is encrypted on download. You must provide a password for a protected POU to open the program block for normal editing, edit in RUN mode, and program status operations.

To password-protect the POU, follow these steps:

1. Right click the MAIN POU and select Properties.
2. Click the Protection node.
3. Select the "Password-protect this POU" checkbox. Enter and verify a password.



4. You can also choose to protect all of the POUs (MAIN, Subroutines, Function block and Interrupt Routines) in your project by selecting the Password-protect All POUs checkbox.

Note

A strong password provides greater security. A strong password length must meet the following requirements:

- Between 10-63 characters
- Must include numbers, uppercase and lowercase letters, and special characters !@#$ %^&*()-+~_={}[]|\:;,.<>
- Not words that can be found in a dictionary,
- Not names or identifiers that can be derived from personal information.

Keep the password secret and change it frequently.

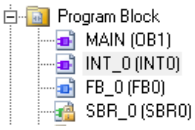
Note

Increased security for POU and data block (data page) passwords

STEP 7-Micro/WIN SMART V2.2 increased the security for passwords from previous versions. If you are working with a project that you created with a previous version of STEP 7-Micro/WIN SMART, re-enter your passwords to activate the increased security.

POU block encryption

When a POU or data block is protected, it appears in the project tree with a lock icon as follows:

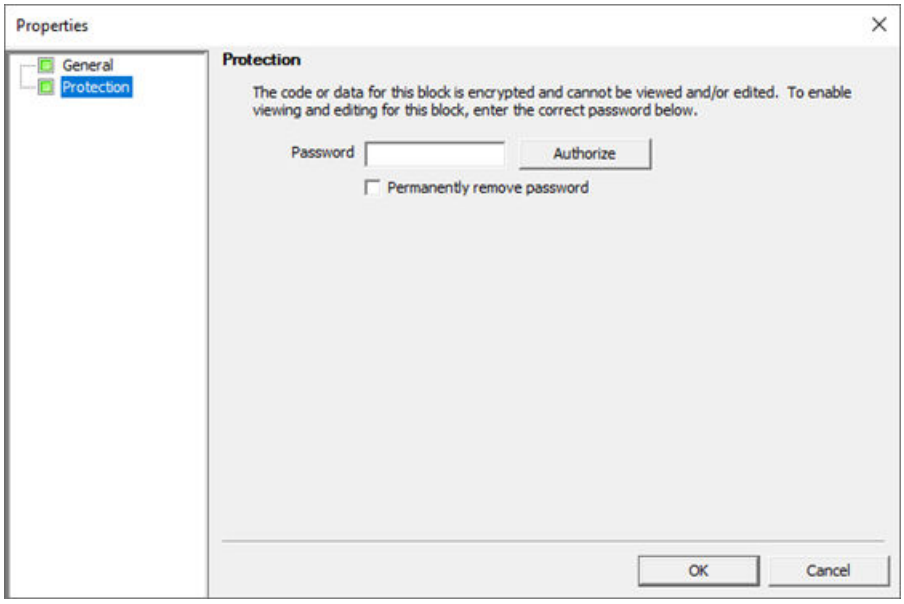


Unlocking a POU

You must provide a password for a protected POU to open the block (for normal editing, edit in RUN mode, and program status operations).

1. Open the POU properties and click the Protection node.
2. Enter your password.
3. By default the "**Permanently remove password**" check box is unchecked
   - **Unchecked:** If you "Authorize" a POU for editing with this box unchecked, then you get a temporary authorization for the current editing session and password protection remains active for future editing sessions.
   - **Checked:** If you check the "Permanently remove password" box: and "authorize" the POU, then the password protection is removed.
4. Click "Authorize".

You must repeat this procedure for each protected POU you chose to unlock.



See also

Trend view details

Project password protection

PLC password protection

Application user reference

Getting started

3.10 Secure decommissioning

In this section, you will find information on how to properly decommission individual components of your automation system. Decommissioning is necessary when the component has reached the end of its service life.

Decommissioning includes environmentally sound disposal and secure removal of all digital data of electronic components with storage medium.

3.10.1 Securely removing data

Before disposing of components of your automation system, you should securely delete all data from the storage media of these components. How to securely delete data from the devices so that it cannot be recovered is described below.

NOTICE
Data misuse resulting from non-secure deletion of data Incomplete or non-secure deletion of data from data memories can result in data misuse by third parties. For this reason, ensure secure deletion of data from all storage media used before disposing of the product.

To delete all data from the data memories of S7-200 SMART CPU, you can follow the step in the following links to clear PLC. The function deletes all information that was saved internally on the module.

- Clearing PLC memory (Page 208)
- Creating a reset-to-factory-defaults memory card (Page 210)

3.10.2 Recycling and disposal

You can fully recycle S7-200 SMART CPU due to their low-pollutant equipment. For environmentally friendly recycling and disposal of your old equipment, contact a certified electronic waste disposal company and dispose of the equipment according to the applicable regulations in your country.

Getting started

4

STEP 7-Micro/WIN SMART V3 makes it easy for you to program your CPU. In just a few short steps using a simple example, you can learn how to create a user program that you can download and run on your CPU.

All you need for this example is an Ethernet or USB-PPI communication cable, a CPU, and a programming device running the STEP 7-Micro/WIN SMART V3 programming software.

4.1 Connecting to the CPU

Connecting your CPU is easy. For this example, you only need to connect power to your CPU and then connect the Ethernet or USB-PPI communication cable between your programming device and the CPU.

Connecting power to the CPU

WARNING

Ensure power is off prior to installing, wiring or removing devices

Before you install or remove any electrical device, ensure that the power to that equipment has been turned off.

Attempts to install or connect the wiring for the CPU or related equipment with power applied could cause electric shock or faulty operation of equipment. Failure to disable all power to the CPU and related equipment during installation or removal procedures could result in death or serious injury to personnel, and/or damage to equipment.

Always follow appropriate safety precautions and ensure that power to the CPU is disabled before attempting to install or remove the CPU or related equipment.

Connect the CPU to a power source. The following figure shows the wiring connections for either a DC or an AC model of the CPU.

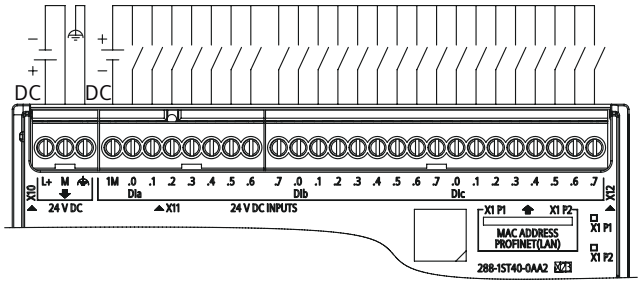


Figure 4-1 DC installation

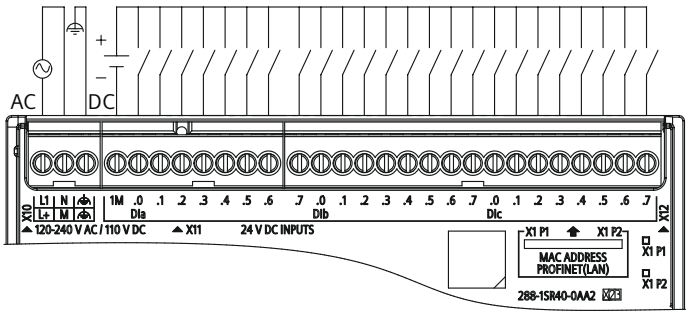
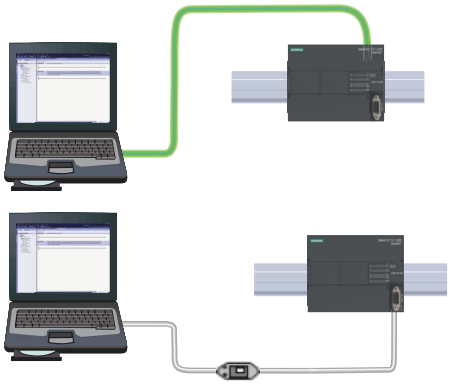


Figure 4-2 AC installation

4.1.1 Configuring the CPU for communication

4.1.1.1 Overview

A CPU can communicate with a STEP 7-Micro/WIN SMART programming device on two types of communications networks:



A CPU can communicate with a STEP 7-Micro/WIN SMART programming device on an Ethernet network.

A CPU can communicate with a STEP 7-Micro/WIN SMART programming device on an RS485 network.

Note

- Support installing STEP 7-Micro/WIN SMART V2.8 and STEP 7-Micro/WIN SMART V3.0 on the same computer.
- STEP 7-Micro/WIN SMART V2.8 can connect to CPU of Version 2.8 or eariler, STEP 7-Micro/WIN SMART V3 can only connect to CPU of version 3.0.
- Only one STEP 7-Micro/WIN SMART can connect to one CPU at the same time.

Consider the following when setting up Ethernet communications between a CPU and a programming device:

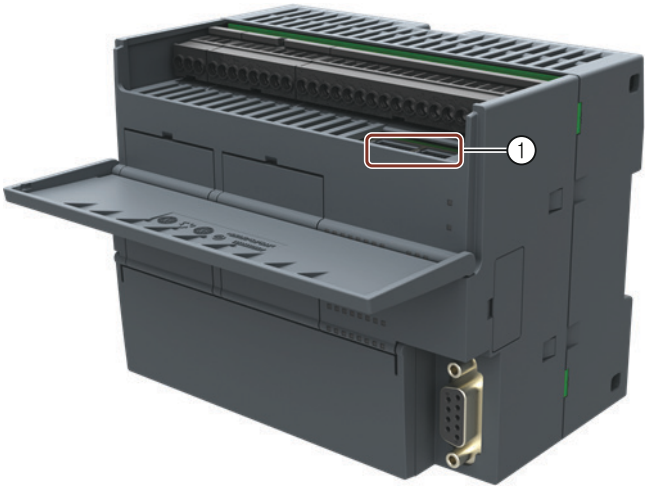
- Configuration/Setup: No hardware configuration is required for a single CPU. If you want multiple CPU's on the same network, then you must change the default IP addresses to new, unique IP addresses.
- Dual network ports, multiple can be daisy-chained

4.1.1.2 Establishing the Ethernet hardware communication connection

The Ethernet interfaces establish the physical connections between a programming device and a CPU. Since Auto-Cross-Over functionality is built into the CPU, either a standard or crossover Ethernet cable can be used for the interface. An Ethernet switch is not required to connect a programming device directly to a CPU.

Follow the steps below to create the hardware connection between a programming device and a CPU:

1. Install the CPU.
2. Remove the RJ45 connection cover from the Ethernet port. Retain the cover for reuse.
3. Plug the Ethernet cable into the Ethernet port on the top right of the CPU as shown below.
4. Connect the Ethernet cable to the programming device.



① PROFINET (LAN) ports

4.1.1.3 Setting up Ethernet communication with the CPU

From STEP 7-Micro/WIN SMART V3, use one of the following methods to display the Ethernet "Communications" dialog for configuring communication to the CPU.

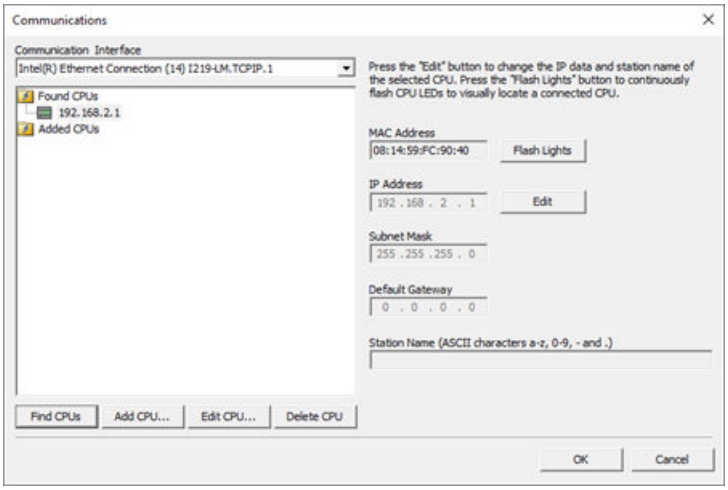
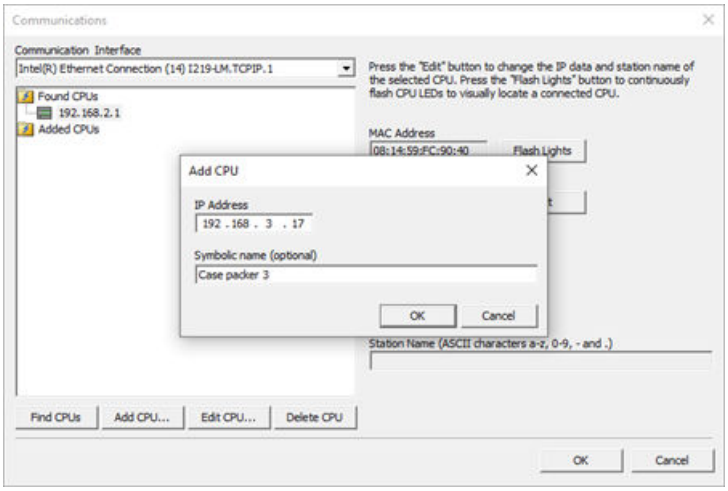
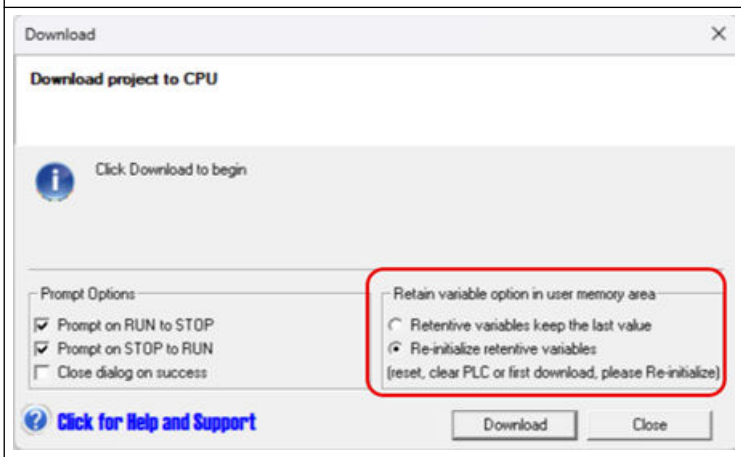
- From the project tree, double-click the "Communications" node.
- Click the "Communications" button from the navigation bar.
- Select "Communications" from the "Component" drop-down list in the Windows area of the "View" menu ribbon strip.

The "Communications" dialog provides two methods of selecting the CPU to be accessed:

- Click the "Find CPUs" button to have STEP 7-Micro/WIN SMART V3 search your local network for CPUs. The IP address of each CPU found on the network is listed under "Found CPUs".
- Click the "Add CPU" button to manually enter the access information (IP address and so forth) for a CPU that you wish to access. The IP address for each CPU, manually added with this method, is listed under "Added CPUs" and is retained.

Note

STEP 7-Micro/WIN SMART V3 can only search for the CPU of version 3.0.

	For "Found CPUs" (CPUs located on your local network), use the "Communications" dialog to connect with your CPU: • Select TCP/IP for your Communication Interface. • Click the "Find CPUs" button to display all operational CPUs ("Found CPUs") on the local Ethernet network. All CPUs have a default IP address. See the Note below. • Highlight a CPU, and then click "OK".
	For "Added CPUs" (CPUs on the local or remote networks), use the "Communications" dialog to connect with your CPU: • Select TCP/IP for your Communication Interface. • Click the "Add CPU" button to do one of the following: – Enter the IP address of a CPU that is accessible from the programming device, but is not on the local network. – Enter the IP address of a CPU directly that is on the local network. All CPUs have a default IP address. See the Note below. • Highlight a CPU, and then click "OK".
	After you have established communication with the CPU, you are ready to create and download the example program. To download all project components, click the Download button from the Transfer area of the File or PLC menu ribbon strip, or alternatively press the shortcut key combination CTRL+D. If STEP 7-Micro/WIN SMART does not find your CPU, check the settings for the communications parameters and repeat these steps.

Note

The CPU list will show all of the CPUs regardless of Ethernet network class and subnet.

To make a connection to your CPU, your Communication Interface (for Ethernet, a network interface card (NIC)) and the CPU must be on the same class of network and on the same subnet. You can either set up your network interface card to match the default IP address of the CPU, or you can change the IP address of the CPU to match the network class and subnet of your network interface card.

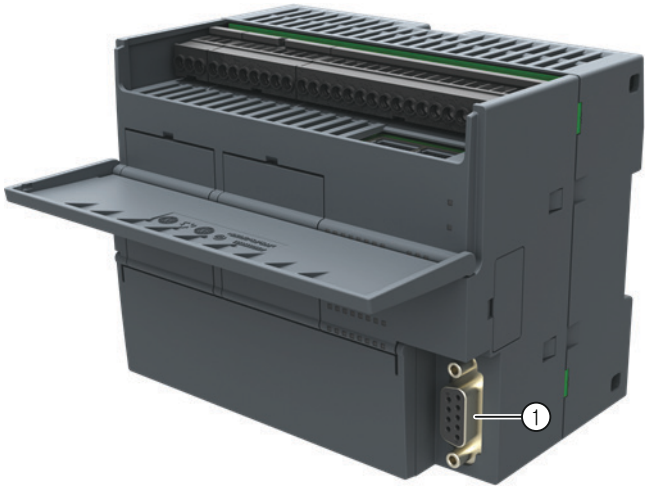
See the "Configuring or changing an IP address for a CPU or device in your project" (Page 482) for information about how to accomplish this.

4.1.1.4 Establishing the RS485 hardware communication connection

The RS485 interfaces establish the physical connections between a programming device and a CPU.

Follow the steps below to create the hardware connection between a programming device and a CPU:

1. Install the CPU.
2. Plug the USB/PPI cable into the RS485 port on the bottom right of the CPU as shown below.
3. Connect the USB/PPI cable to the programming device.



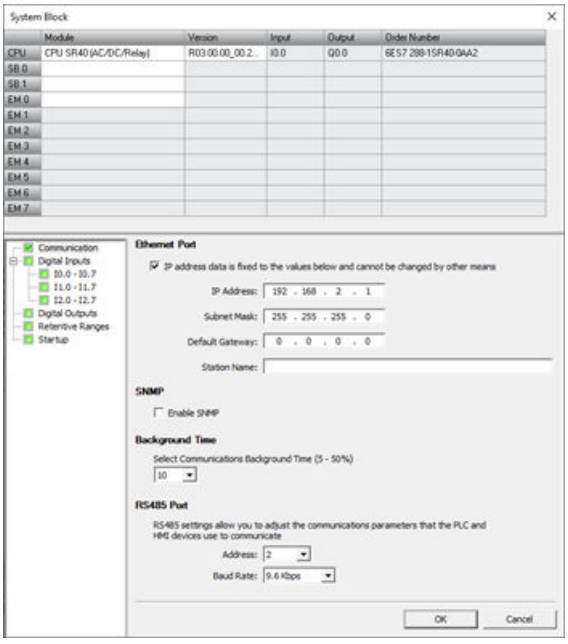
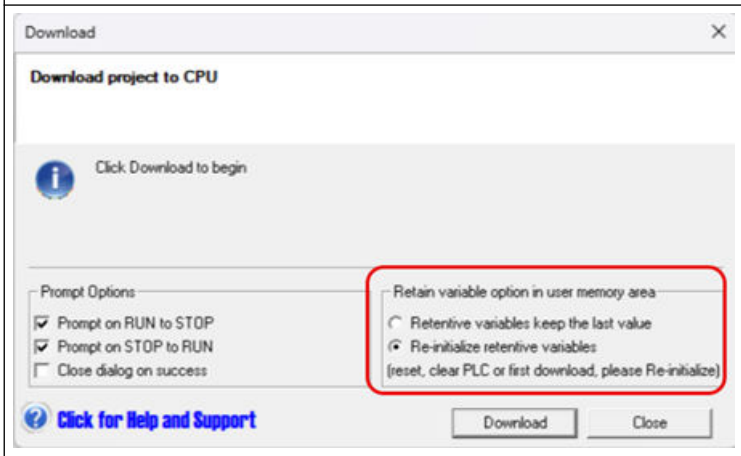
① RS485 port

4.1.1.5 Setting up RS485 communication with the CPU

RS485 network information configuration or changes done in the system block are part of the project and do not become active until you download your project to the CPU.

To access this dialog, perform one of the following:

- In the "Navigation" bar, click the "System Block" button.
- In the Project tree, select the "System Block" node, then press Enter; or double-click the "System Block" node.

	Enter or change the following access information: • RS485 port address • RS485 port baud rate
	After you have established communication with the CPU, you are ready to create and download the example program. To download all project components, click the Download button from the Transfer area of the File or PLC menu ribbon strip, or alternatively press the shortcut key combination CTRL+D. If STEP 7-Micro/WIN SMART does not find your CPU, check the settings for the communications parameters and repeat these steps.

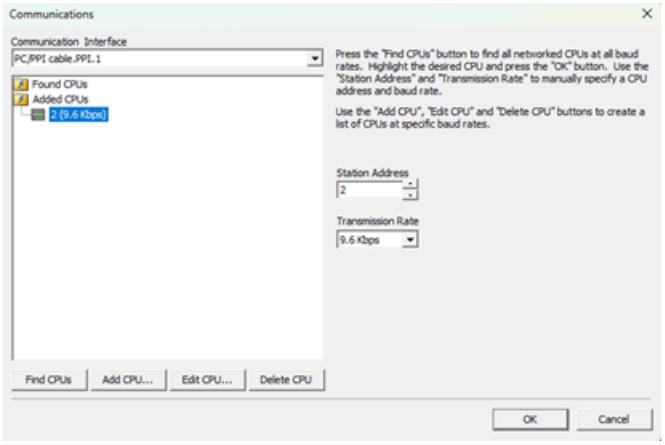
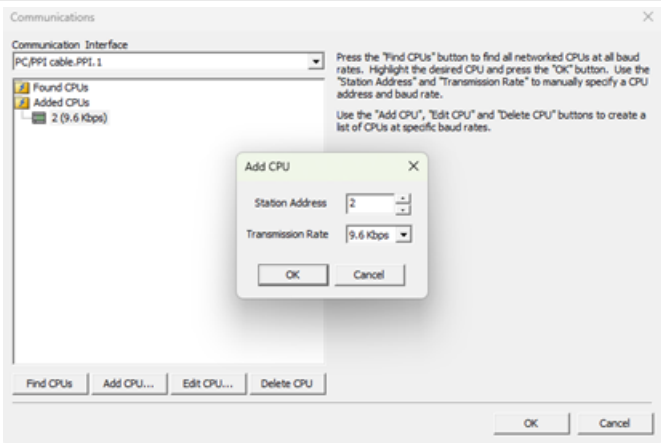
All CPUs and devices that have valid RS485 port addresses are displayed in the "Communications" dialog.

In STEP 7-Micro/WIN SMART, you can access CPUs in one of two ways:

- From the project tree, double-click the "Communications" node.
- Click the "Communications" button from the navigation bar.
- Select "Communications" from the "Component" drop-down list in the Windows area of the "View" menu ribbon strip.

The "Communications" dialog provides two methods of selecting the CPU to be accessed:

- Click the "Find CPUs" button to have STEP 7-Micro/WIN SMART V3 search your local network for CPUs. The RS485 network address of each CPU found on the network is listed under "Found CPUs".
- Click the "Add CPU" button to manually enter the access information (RS485 network address and baud rate) for a CPU that you wish to access. The RS485 network address for each CPU, manually added with this method, is listed under "Added CPUs" and is retained.

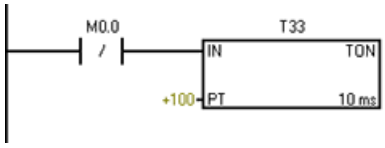
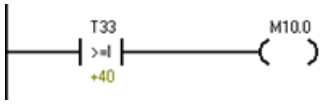
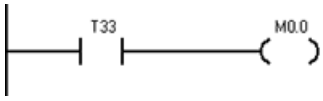
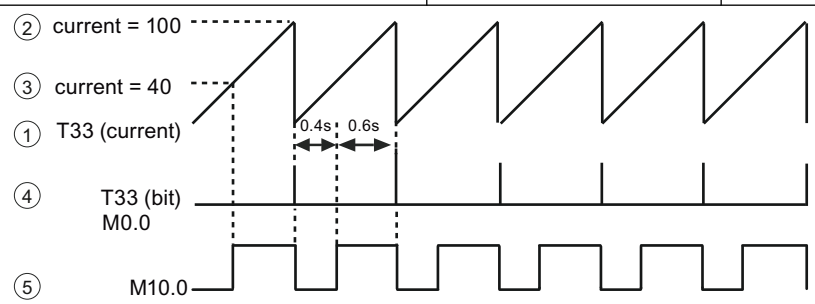
	For "Found CPUs" (CPUs located on the RS485 network), use the "Communications" dialog to connect with your CPU: • Select "PC/PPI cable.PPI.1" for your Communication Interface. • Click the "Find CPUs" button to display all operational CPUs ("Found CPUs") on the RS485 network. All CPUs default their RS485 network settings to address 2 and 9.6 Kbps. • Highlight a CPU, and then click "OK". Note: You can open multiple copies of STEP 7-Micro/WIN SMART on a computer. Be aware that when you open a second copy of STEP 7-Micro/WIN SMART or use the "Find CPUs" button in either copy, the communication connection to the CPU in your first/other copy of STEP 7-Micro/WIN SMART might be disconnected.
	For "Added CPUs" (CPUs on the RS485 network), use the "Communications" dialog to connect with your CPU: • Select "PC/PPI cable.PPI.1" for your Communication Interface. • Click the "Add CPU" button. • Enter the RS485 network address and baud rate of a CPU that you wish to access directly on the RS485 network. You can add multiple CPUs on the RS485 network. As always, STEP 7-Micro/WIN SMART communicates with one CPU at a time. All CPUs default their RS485 network settings to address 2 and 9.6 Kbps. • Highlight a CPU, and then click "OK".

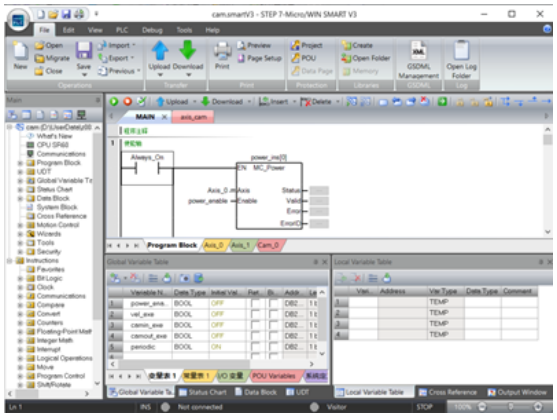
4.2 Creating the sample program

Entering this example of a control program will help you understand how easy it is to use STEP 7-Micro/WIN SMART V3. This program uses six instructions in three networks to create a very simple, self-starting timer that resets itself.

For this example, you use the Ladder (LAD) editor to enter the instructions for the program. The following example shows the complete program in both LAD and Statement List (STL). The description column explains the logic for each network. The timing diagram shows the operation of the program. There are no network comments in the STL program.

Table 4-1 Sample program for getting started with STEP 7-Micro/WIN SMART V3

LAD/FBD	STL	Description
	**Network 1** **LDN M0.0** **TON T33, +100**	10 ms timer T33 times out after (100 x 10 ms = 1 s) M0.0 pulse is too fast to monitor with Status view.
	**Network 2** **LDW>= T33, +40** **= M10.0**	Comparison becomes true at a rate that is visible with Status view. Turn on M10.0 after (40 x 10 ms = 0.4 s) for a 40% OFF / 60% ON waveform.
	**Network 3** **LD T33** **= M0.0**	T33 (bit) pulse is too fast to monitor with Status view. Reset the timer through M0.0 after the (100 x 10 ms = 1 s) period.
		**Timing diagram:** • ① T33 (current) • ② Current = 100 • ③ Current = 40 • ④ T33 (bit) and M0.0 • ⑤ M10.0



Notice the project tree and the program editor. You use the project tree to insert instructions into the networks of the program editor by dragging and dropping the instructions from the "Instructions" portion of the Project tree to the networks.

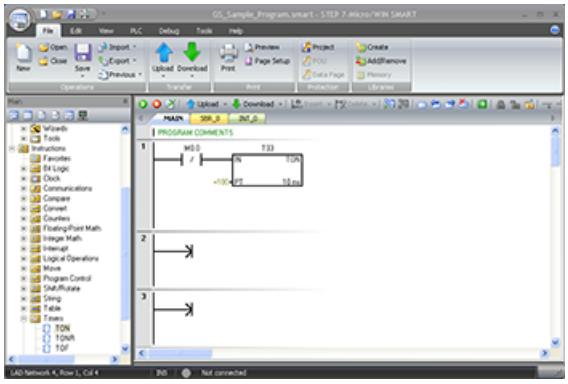
The Program Block folder in the project tree contains all of the blocks of your program.

The program editor toolbar icons provide shortcuts to PLC commands and programming operation.

After you enter and save the program, you can download the program to the CPU.

4.2.1 Network 1: Starting the timer

Network 1: Starting the timer



When M0.0 is off (0), this contact turns on and provides power flow to start the timer.

To enter the contact for M0.0:

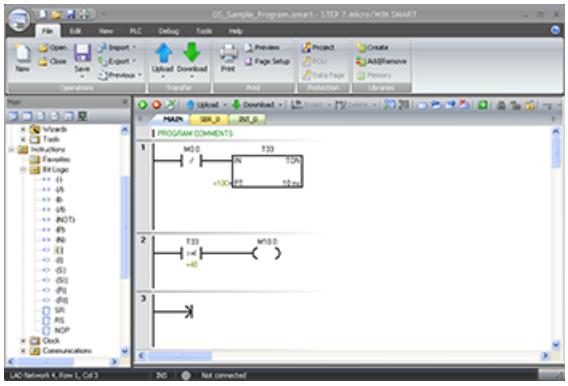
1. Either double-click the "Bit Logic" icon or click the plus sign (+) to display the bit logic instructions.
2. Select the "Normally Closed" contact.
3. Hold down the left mouse button and drag the contact onto the first network.
4. Enter the following address for the contact: M0.0
5. Press the Return key to enter the address for the contact.

To enter the timer instruction for T33:

1. Double-click the "Timers" icon to display the timer instructions.
2. Select the "TON" (on-delay timer) instruction.
3. Hold down the left mouse button and drag the timer onto the first network.
4. Enter the following timer number for the timer: T33
5. Press the Return key to enter the timer number and to move the focus to the preset time (PT) parameter.
6. Enter the following value for the preset time: +100.
7. Press the Return key to enter the value.

4.2.2 Network 2: Turning the output on

Network 2: Turning the output on



When the timer value for T33 is greater than or equal to 40 (40 times 10 milliseconds, or 0.4 seconds), the contact provides power flow to turn on output M10.0 of the CPU.

To enter the Compare instruction:

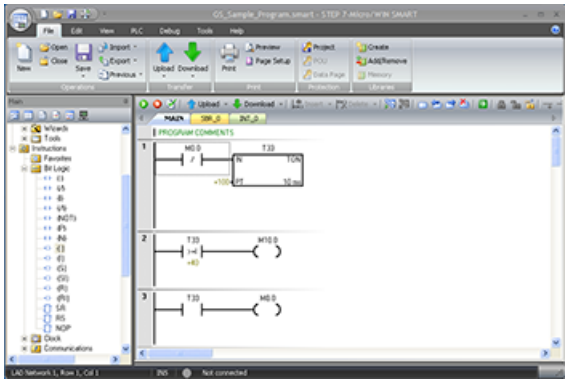
1. Double-click the Compare icon to display the compare instructions. Select the ">=I" instruction (greater-than-or-equal-to-integer).
2. Hold down the left mouse button and drag the compare instruction onto the second network.
3. Click "???" above the contact and enter the address for the timer value: T33
4. Press the Return key to enter the timer number and to move the focus to the other value to be compared with the timer value.
5. Enter the following value to be compared with the timer value: +40
6. Press the Return key to enter the value.

To enter the instruction for turning on output M10.0:

1. Double-click the Bit Logic icon to display the bit logic instructions and select the output coil.
2. Hold down the left mouse button and drag the coil onto the second network.
3. Click "???" above the coil and enter the following address: M10.0
4. Press the Return key to enter the address for the coil.

4.2.3 Network 3: Resetting the timer

Network 3: Resetting the timer



When the timer reaches the preset value (100) and turns the timer bit on, the contact for T33 turns on. Power flow from this contact turns on the M0.0 memory location. Because the timer is enabled by a Normally Closed contact for M0.0, changing the state of M0.0 from off (0) to on (1) resets the timer.

To enter the contact for the timer bit of T33:

1. Select the "Normally Open" contact from the bit logic instructions.
2. Hold down the left mouse button and drag the contact onto the third network.
3. Click "???" above the contact and enter the address of the timer bit: T33
4. Press the Return key to enter the address for the contact.

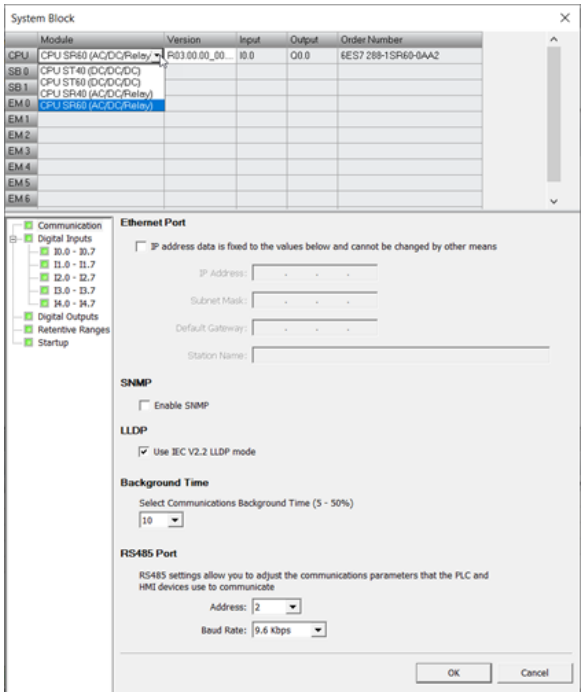
To enter the coil for turning on M0.0:

1. Select the output coil from the bit logic instructions.
2. Hold down the left mouse button and drag the output coil onto the third network.
3. Click "???" above the coil and enter the following address: M0.0
4. Press the Return key to enter the address for the coil.

4.2.4 Setting the CPU type and version for your project

Configure your project for the CPU and version matching your physical CPU. If the project is not configured for the correct CPU and CPU version, then the download could fail or the program may not run.

To select your CPU, click the "CPU" field under the "Module" column to display the dropdown list button, and select your CPU from the dropdown list. Using the same procedure, select your CPU version in the "Version" column.

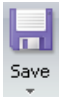


4.2.5 Saving the sample project

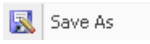
Saving the sample project

After entering the three networks of instructions, you have finished entering the program. When you save the program, you create a project that includes the CPU type and other parameters. To save the project in a file name and location that you specify:

1. Click the down arrow under the Save button from the Operations area of the File menu ribbon strip to display the Save As button.



2. Click the Save As button and provide a filename for saving your project.



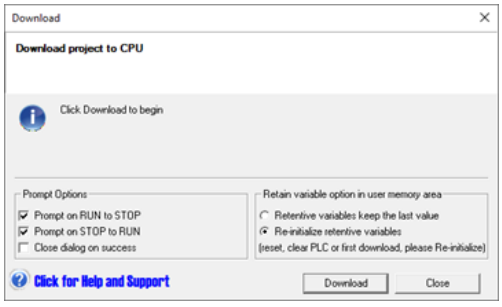
3. Enter a name for the project in the "Save As" dialog.
4. Browse to a location where you want to save your project.
5. Click "Save" to save the project.

After saving the project, you can download the program to the CPU.

4.3 Downloading the sample program

First, ensure that your network hardware and PLC connector cable for either Ethernet (Page 71) (standard CPUs only) or RS485 (Page 74) communications is working, and that PLC communication is operating properly.

To download all project components, click the "Download" button from the "Transfer" area of the File or PLC menu ribbon strip, or alternatively press the shortcut key combination "CTRL+D".



Click the Download dialog "Download" button.

STEP 7-Micro/WIN SMART V3 copies the complete program or program components that you selected to the CPU.

If your CPU is in RUN mode, a dialog prompts you to place the CPU in STOP mode. Clicking "Yes" sets the CPU to STOP mode.

Note

Each project is associated with a CPU type. If the project type does not match the CPU to which you are connected, STEP 7-Micro/WIN SMART V3 indicates a mismatch and prompts you to take an action.

4.4 Changing the operating mode of the CPU

The CPU has two modes of operation: STOP mode and RUN mode. The status LEDs on the front of the CPU indicates the current mode of operation. In STOP mode, the CPU is not executing the program, and you can download program. In RUN mode, the CPU is executing the program; however, you can download program.

Placing the CPU in RUN mode

1. Click the "RUN" button on either the PLC menu ribbon strip or on the program editor toolbar:
2. When prompted, click "OK" to change the operating mode of the CPU.

You can monitor the program in STEP 7-Micro/WIN SMART V3 by clicking the "Program Status" button from the "Debug" menu ribbon strip, or from the program editor toolbar. STEP 7-Micro/WIN SMART V3 displays the values for the instructions.

Placing the CPU in STOP mode

To stop the program, click the "STOP" button and acknowledge the prompt to place the CPU in STOP mode. You can also place a STOP instruction (Page 425) in your program logic to put the CPU in STOP mode.

Installation 5

5.1 Guidelines for installing S7-200 SMART devices

The S7-200 SMART equipment is designed to be easy to install. You can install the S7-200 SMART either on a panel or on a standard DIN rail, and you can orient the S7-200 SMART either horizontally or vertically. The small size of the S7-200 SMART allows you to make efficient use of space.

WARNING

Safety requirements for installing S7-200 SMART PLCs

S7-200 SMART PLCs are Open Type Controllers. You must install the PLC in a housing, cabinet, or electric control room. Open Type Controllers must be installed within an enclosure which protects you from hazards, including mechanical hazards, electrical shock and spread of fire. Limit entry to the housing, cabinet, or electric control room to authorized personnel.

Failure to follow these installation requirements could result in death or serious injury to personnel, and/or damage to equipment.

Always follow these requirements when installing the PLC.

The installation should provide a dry environment for the S7-200 SMART. SELV/PELV circuits are considered to provide protection against electric shock in dry locations.

The installation should provide the appropriate mechanical strength, flammability protection, and stability protection that is approved for open equipment in your particular location category according to applicable electrical and building codes.

Conductive contamination due to dust, moisture, and airborne pollution can cause operational and electrical faults in the PLC.

If you locate the PLC in an area where conductive contamination may be present, the PLC must be protected by an enclosure with appropriate protection rating. IP54 is one rating that is generally used for electronic equipment enclosures in dirty environments and may be appropriate for your application.

WARNING

Improper installation of the S7-200 SMART can result in electrical faults or unexpected operation of machinery.

Electrical faults or unexpected machine operation can result in death, severe personal injury, and/or property damage.

All instructions for installation and maintenance of a proper operating environment must be followed to ensure the equipment operates safely.

Separate the devices from heat, high voltage, and electrical noise

As a general rule for laying out the devices of your system, always separate the devices that generate high voltage and high electrical noise from the low-voltage, logic-type devices such as the PLC.

When configuring the layout of the PLC inside your panel, consider the heat-generating devices and locate the electronic-type devices in the cooler areas of your cabinet. Reducing the exposure to a high-temperature environment will extend the operating life of any electronic device.

Consider also the routing of the wiring for the devices in the panel. Avoid placing low-voltage signal wires and communications cables in the same tray with AC power wiring and high-energy, rapidly-switched DC wiring.

Provide adequate clearance for cooling and wiring

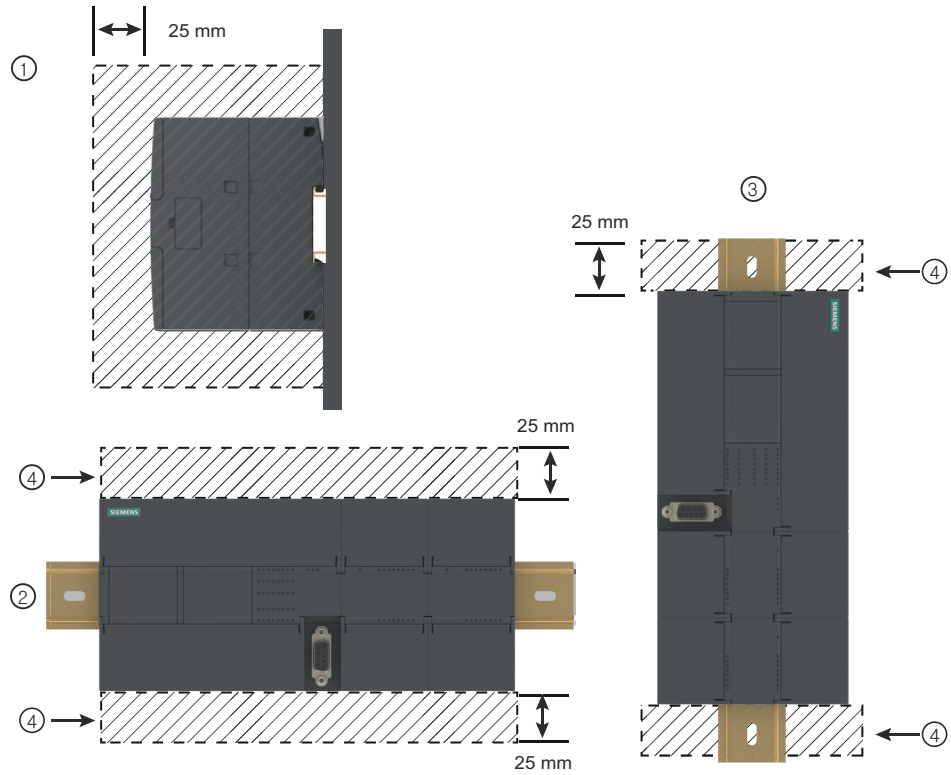
S7-200 SMART devices are designed for natural convection cooling. For proper cooling, you must provide a clearance of at least 25 mm above and below the devices. Also, allow at least 25 mm of depth between the front of the modules and the inside of the enclosure.

CAUTION

Temperature considerations

Vertical mounting reduces the maximum allowable ambient temperature by 10 degrees C. Operating outside the maximum temperature range could result in erratic process operation and could result in minor personal injury.

If your installation includes expansion modules, mount the CPU below them as shown in the following figure. Follow the prescribed guidelines for mounting modules to ensure proper cooling.



①	Side view	③	Vertical installation
②	Horizontal installation	④	Clearance area

When planning your layout for the PLC, allow enough clearance for the wiring and communications cable connections.

5.2 Power budget

Your CPU has an internal power supply that provides power for the CPU, the expansion modules, signal boards, and other 24 V DC user power requirements. Use the following information as a guide for determining how much power (or current) the CPU can provide for your configuration.

Refer to the technical specifications for your particular CPU (Page 23) to determine the 24 V DC sensor supply power budget, the 5 V DC logic budget supplied by your CPU and the 5 V DC power requirements of the expansion modules and signal boards. Refer to the Calculating a power budget (Page 1057) to determine how much power (or current) the CPU can provide for your configuration.

The standard CPU provides the 5 V DC logic power needed for any expansion in your system. Pay careful attention to your system configuration to ensure that the CPU can supply the 5 V

DC power required by your selected expansion modules. If your configuration requires more power than the CPU can supply, you must remove a module.

Note

If the CPU power budget is exceeded, you may not be able to connect the maximum number of modules allowed for your CPU.

The standard CPU also provides a 24 V DC sensor supply that can supply 24 V DC for input points, for relay coil power on the expansion modules, or for other requirements. If your power requirements exceed the budget of the sensor supply, then you must add an external 24 V DC power supply to your system. You must manually connect the 24 V DC supply to the input points or relay coils.

If you require an external 24 V DC power supply, ensure that the power supply is not connected in parallel with the sensor supply of the CPU. For improved electrical noise protection, it is recommended that the commons (M) of the different power supplies be connected.

WARNING

Connecting power supplies safely

Connecting an external 24 V DC power supply in parallel with the 24 V DC sensor supply of the CPU can result in a conflict between the two supplies as each seeks to establish its own preferred output voltage level.

The result of this conflict can be shortened lifetime or immediate failure of one or both power supplies, with consequent unpredictable operation of the PLC system. Unpredictable operation could result in death or serious injury to personnel, and/or damage to equipment.

The DC sensor supply of the CPU and any external power supply should provide power to different points. A single connection of the commons is allowed.

Some of the 24 V DC power input ports in the S7-200 SMART system are interconnected, with a common logic circuit connecting multiple M terminals. For example, the following circuits are interconnected when designated as "not isolated" in the data sheets: the 24 V DC power supply of the CPU, the power input for the relay coil of an EM, or the power supply for a non-isolated analog input. All non-isolated M terminals must connect to the same external reference potential.

WARNING

Avoiding unwanted current flow

Connecting non-isolated M terminals to different reference potentials will cause unintended current flows that may cause damage or unpredictable operation in the PLC and any connected equipment.

Failure to comply with these guidelines could cause damage or unpredictable operation which could result in death or severe personal injury and/or property damage.

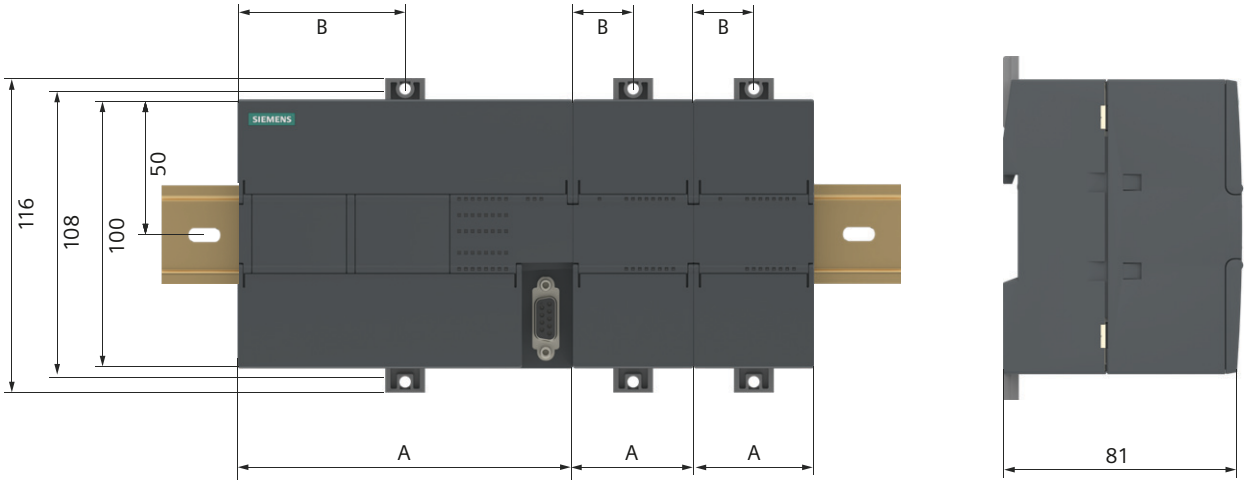
Always ensure that all non-isolated M terminals in an S7-200 SMART system are connected to the same reference potential.

Refer to the technical specifications for your particular CPU (Page 23) to determine the 24 V DC sensor supply power budget, the 5 V DC logic budget supplied by your CPU and the 5 V DC power requirements of the expansion modules and signal boards.

5.3 Installation and removal procedures

5.3.1 Mounting dimensions for the S7-200 SMART devices

The CPU and expansion modules include mounting holes to facilitate installation on panels.

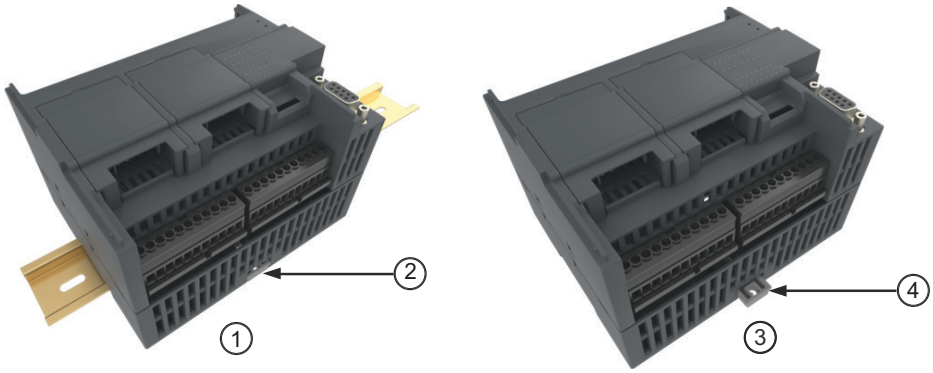


S7-200 SMART module		Width A (mm)	Width B (mm)
CPU SR40, CPU ST40		125	62.5
CPU SR60, CPU ST60 [1]		175	37.5[1]
Expansion modules:	EM 4AI, EM 8AI, EM 2AQ, EM 4AQ, EM 8DI, EM 16DI, EM 8DQ, and EM 8DQ RLY, EM 16DQ RLY, and EM 16DQ Transistor	45	22.5
	EM 8DI/8DQ and EM 8DI/8DQ RLY	45	22.5
	EM 16DI/16DQ and EM 16DI/16DQRLY	70	35
	EM 2AI/1AQ and EM 4AI/2AQ	45	22.5
	EM 2RTD, EM 4RTD	45	22.5
	EM 4TC	45	22.5
	EM DP01	70	35

[1]The CPU xx60 models have two sets of mounting holes. The width "B" dimension is measured from the center of each mounting hole to the corresponding edge of the housing.

5.3.2 Installing and removing the CPU

The CPU can be easily installed on a standard DIN rail or on a panel. DIN rail clips are provided to secure the device on the DIN rail. The clips also snap into an extended position to provide a screw mounting position for panel-mounting the unit.



① DIN rail installation
② DIN rail clip in latched position
③ Panel installation
④ Clip in extended position

Figure 5-1 Installation on a DIN rail or on a panel

Before you install or remove any electrical device, ensure that the power to that equipment has been turned off. Also, ensure that the power to any related equipment has been turned off.

WARNING

Remove power to PLC before installing or removing equipment

Attempts to install or remove the PLC or related equipment with the power applied could cause electric shock or faulty operation of equipment.

Failure to disable all power to the PLC and related equipment during installation or removal procedures could result in death or serious injury to personnel, and/or damage to equipment.

Always follow appropriate safety precautions and ensure that power to the PLC is disabled before attempting to install or remove the CPU or related equipment.

Always ensure that whenever you replace or install a device, you use the correct module or equivalent device.

WARNING

Module replacement

If you install an incorrect module, the program in the CPU could function unpredictably.

Failure to replace a device with the same model, orientation, or order could result in death or serious injury to personnel, and/or damage to equipment.

Replace the device with the same model, and be sure to orient and position it correctly.

Note

Install expansion modules separately after the CPU has been installed.

Consider the following when installing the units on the DIN rail or on a panel:

- For DIN rail mounting, make sure the upper DIN rail clip is in the latched (inner) position and that the lower DIN rail clip is in the extended position for the CPU.
- After installing the devices on the DIN rail, move the lower DIN rail clips to the latched position to lock the devices on the DIN rail.
- For panel mounting, make sure the DIN rail clips are pushed to the extended position.

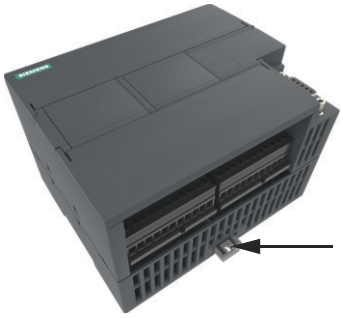
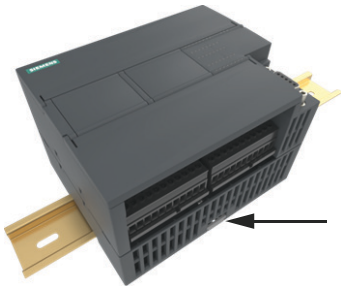
To install the CPU on a panel, follow these steps:

1. Locate, drill, and tap the mounting holes (M4 or American Standard number 8), using the dimensions in the table, Mounting dimensions (mm) (Page 89).
2. Ensure that the CPU and S7-200 SMART equipment are disconnected from electrical power.
3. Secure the module(s) to the panel, using a Pan Head M4 screw with spring and flat washer. Do not use a flat head screw.
4. If you are using an expansion module, put it next to the CPU and slide together until the connectors join securely.

   Note

   The type of screw will be determined by the material upon which it is mounted. You should apply appropriate torque until the spring washer becomes flat. Avoid applying excessive torque to the mounting screws. Do not use a flat head screw.

Table 5-1 Installing a CPU on a DIN rail

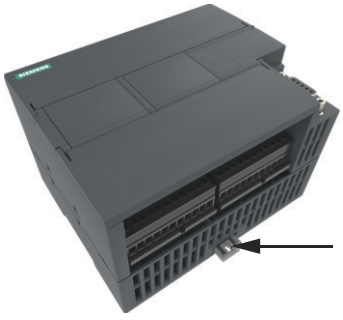
Task	Procedure
	Follow the steps below to install a CPU on a DIN rail. 1. Secure the rail to the mounting panel every 75 mm. 2. Snap open the DIN clip (located on the bottom of the module) and hook the back of the module onto the DIN rail. 3. Rotate the module down to the DIN rail and snap the clip closed. Carefully check that the clip has fastened the module securely onto the rail. To avoid damage to the module, press on the tab of the mounting hole instead of pressing directly on the front of the module.
	

Note

Using DIN rail stops could be helpful if your CPU is in an environment with high vibration potential or if the CPU has been installed vertically. Use an end bracket (8WA1 808 or 8WA1 805) on the DIN rail to ensure that the modules remain connected.

If your system is in a high-vibration environment, then panel-mounting the CPU will provide a greater level of vibration protection.

Table 5-2 Removing a CPU from a DIN rail

Task	Procedure
	Follow the steps below to remove a CPU from a DIN rail. 1. Remove power from the CPU and any attached I/O modules. 2. Disconnect all the wiring and cabling that is attached to the CPU. The CPU and most expansion modules have removable connectors to make this job easier. 3. Unscrew the mounting screws or snap open the DIN clip. 4. If you have expansion modules connected, slide the CPU to the left to disengage it from the expansion module connector. Note: unscrewing or unsnapping the DIN clips of the expansion modules can make it easier to disengage the CPU. 5. Remove the CPU.

5.3.3 Installing and removing a signal board or battery board

Table 5-3 Installing a signal board on a CPU

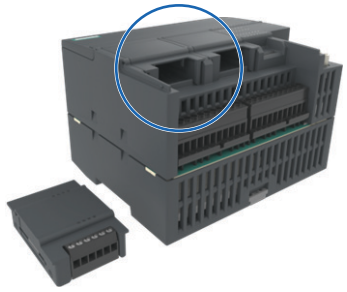
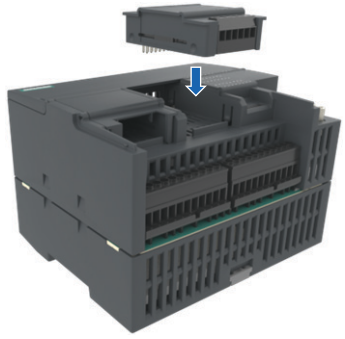
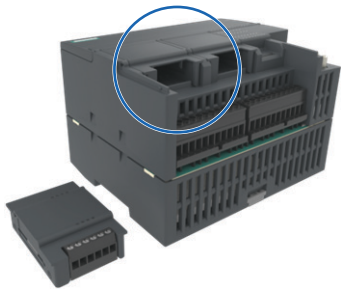
Task	Procedure
	Follow the steps below to install a signal board or battery board 1. Ensure that the CPU and all S7-200 SMART equipment are disconnected from electrical power. 2. Remove the top and bottom terminal block covers from the CPU. 3. Place a screwdriver into the slot on top of the CPU at the rear of the cover. 4. Gently pry the cover up and remove it from the CPU. 5. Place the signal board or battery board straight down into its mounting position in the top of the CPU. 6. Firmly press the module into position until it snaps into place. 7. Replace the terminal block covers.
	

Table 5-4 Removing a signal board or battery board on a CPU

Task	Procedure
	Follow the steps below to remove a signal board or battery board 1. Ensure that the CPU and all S7-200 SMART equipment are disconnected from electrical power. 2. Remove the top and bottom terminal block covers from the CPU. 3. Place a screwdriver into the slot on top of the module. 4. Gently pry the module up to disengage it from the CPU. 5. Remove the module straight up from its mounting position in the top of the CPU. 6. Replace the cover onto the CPU. 7. Replace the terminal block covers.
	

Installing or replacing the battery in the SB BA01 battery board

The SB BA01 battery board requires battery type CR1025. The battery is not included with the SB BA01 and must be purchased.

To install the battery, follow these steps:

1. In the SB BA01, install the new battery with the positive side of the battery on top, and the negative side next to the printed wiring board.
2. The SB BA01 is now ready to be installed in the CPU. Follow the installation directions above.

To replace the battery, follow these steps:

1. Remove the SB BA01 from the CPU following the removal directions above.
2. Carefully remove the old battery using a small screwdriver. Push the battery out from under the clip.
3. Install a new CR1025 replacement battery with the positive side of the battery on top and the negative side next to the printed wiring board.
4. Re-install the SB BA01 battery board following the installation directions above.

5.3.4 Removing and reinstalling the terminal block connector

The S7-200 SMART modules have removable connectors to make connecting the wiring easy.

Table 5-5 Removing the connector

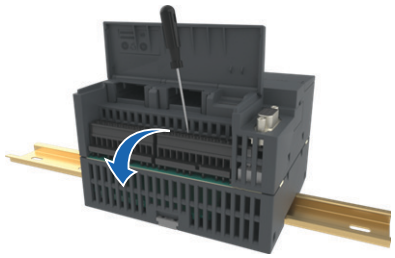
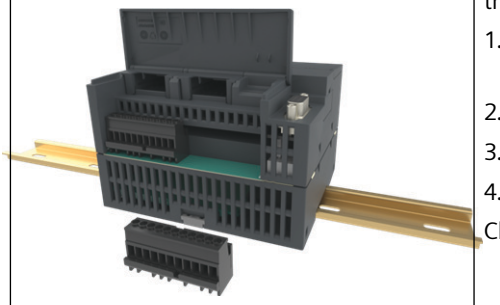
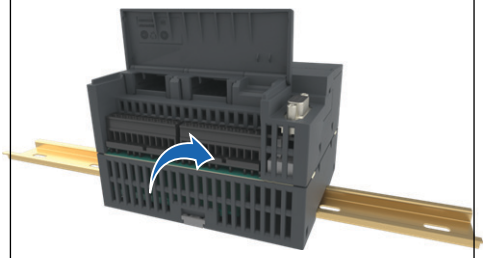
Task	Procedure
	Prepare the system for terminal block removal by removing the power from the CPU and opening the cover above the connector. 1. Ensure that the CPU and all S7-200 SMART equipment are disconnected from electrical power. 2. Inspect the top of the connector and locate the slot for the tip of the screwdriver. 3. Insert a small screwdriver into the slot. 4. Gently pry the top of the connector away from the CPU. The connector will release with a snap. 5. Grasp the connector and remove it from the CPU.
	

Table 5-6 Installing the connector

Task	Procedure
	Prepare the components for terminal block installation by removing power from the CPU and opening the cover above the connector. 1. Ensure that the CPU and all S7-200 SMART equipment are disconnected from electrical power. 2. Align the connector with the pins on the unit. 3. Align the wiring edge of the connector inside the rim of the connector base. 4. Press firmly down and rotate the connector until it snaps into place. Check carefully to ensure that the connector is properly aligned and fully engaged.
	

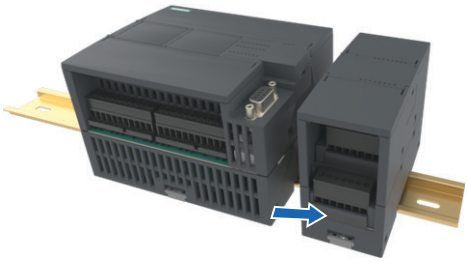
5.3.5 Installing and removing an expansion module

Install expansion modules separately after the CPU has been installed.

Table 5-7 Installing an expansion module

Task	Procedure
	Follow the steps below to install an expansion module: 1. Ensure that the CPU and all S7-200 SMART equipment are disconnected from electrical power. 2. Remove the cover for the I/O bus connector from the right side of the CPU. 3. Insert a screwdriver into the slot above the cover. 4. Gently pry the cover out at its top and remove the cover. Retain the cover for reuse.
	Connect the expansion module to the CPU. 1. Pull out the bottom DIN rail clip to allow the expansion module to fit over the rail. 2. Position the expansion module to the right of the CPU. 3. Hook the expansion module over the top of the DIN rail. 4. Slide the expansion module to the left until the I/O connector fully engages the connector on the right of the CPU and push the bottom clip in to latch the expansion module onto the rail.

Table 5-8 Removing an expansion module

Task	Procedure
	Follow the steps below to remove an expansion module: 1. Ensure that the CPU and all S7-200 SMART equipment are disconnected from electrical power. 2. Remove the I/O connectors and wiring from the expansion module. Loosen the DIN rail clips of all the S7-200 SMART devices. 3. Physically slide the expansion module to the right.
	

5.3.6 Installing and removing the expansion cable

The S7-200 SMART expansion cable provides additional flexibility in configuring the layout of your S7-200 SMART system. Only one expansion cable is allowed per CPU system. You install the expansion cable either between the CPU and the first EM, or between any two EMs.

Table 5-9 Installing and removing the male connector of the expansion cable

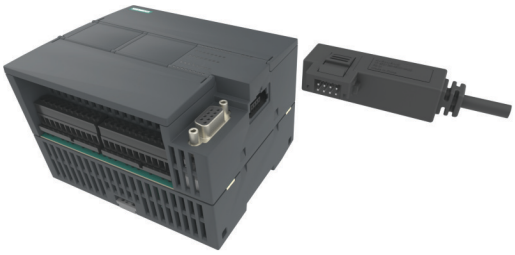
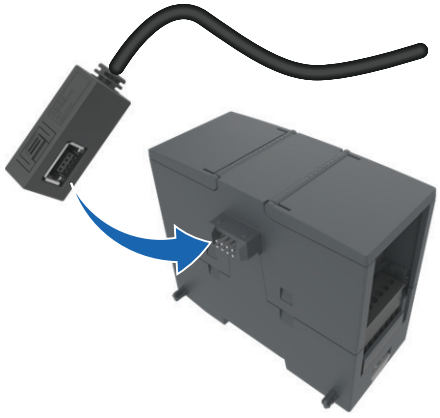
Task	Procedure
	To install the male connector: 1. Ensure that the CPU and all S7-200 SMART equipment are disconnected from electrical power. 2. Push the male connector into the bus connector on the right side of the expansion module or CPU. 3. The male connector is locked in place when it is fully seeded. To remove the male connector: 1. Ensure that the CPU and all S7-200 SMART equipment are disconnected from electrical power. 2. Use your thumb to press down the latch on the top of the male connector to release it from the expansion module or CPU. 3. Remove the male connector from the expansion module or CPU by pulling it straight out.

Table 5-10 Installing and removing the female connector of the expansion cable

Task	Procedure
	To install the female connector: 1. Ensure that the CPU and all S7-200 SMART equipment are disconnected from electrical power. 2. Push the female connector into the bus connector on the left side of the expansion module. 3. The female connector is locked in place when it is fully seeded. To remove the female connector: 1. Ensure that the CPU and all S7-200 SMART equipment are disconnected from electrical power. 2. Use your thumb to press down the latch on the top of the female connector to release it from the expansion module. 3. Remove the female connector from the expansion module by pulling it straight out.

Note

Installing the expansion cable in a vibration environment

If the expansion cable is connected to modules that move or are not firmly fixed, the connection on the cable ends can gradually become loose.

Use a cable tie to fix the cable ends on the DIN-rail (or other place) to provide extra strain relief.

Avoid using excessive force when you pull the cable during installation. Ensure the cable-module connection is in the correct position once installation is complete.

5.4 Wiring guidelines

Proper grounding and wiring of all electrical equipment is important to help ensure the optimum operation of your system and to provide additional electrical noise protection for your application and the PLC. Refer to the technical specifications (Page 981) for the wiring diagrams.

Prerequisites

Before you ground or install wiring to any electrical device, ensure that the power to that equipment has been turned off. Also, ensure that the power to any related equipment has been turned off.

Ensure that you follow all applicable electrical codes when wiring the PLC and related equipment. Install and operate all equipment according to all applicable national and local standards. Contact your local authorities to determine which codes and standards apply to your specific case.

WARNING

Attempts to install or wire the PLC or related equipment with power applied could cause electric shock or faulty operation of equipment. Failure to disable all power to the PLC and related equipment during installation or removal procedures could result in death or serious injury to personnel, and/or damage to equipment.

Always follow appropriate safety precautions and ensure that power to the PLC is disabled before attempting to install or remove the PLC or related equipment.

Always take safety into consideration as you design the grounding and wiring of your PLC system. Electronic control devices, such as the PLC, can fail and can cause unexpected operation of the equipment that is being controlled or monitored. For this reason, you should implement safeguards that are independent of the PLC to protect against possible personal injury or equipment damage.

WARNING

Control devices can fail in an unsafe condition, resulting in unexpected operation of controlled equipment. Such unexpected operations could result in death or serious injury to personnel, and/or damage to equipment.

Use an emergency stop function, electromechanical overrides, or other redundant safeguards that are independent of the PLC.

Isolation guidelines

The AC power supply boundaries and I/O boundaries to AC circuits have been designed and approved to provide safe separation between AC line voltages and low voltage circuits. These boundaries include double or reinforced insulation, or basic plus supplementary insulation, according to various standards. Components which cross these boundaries such as optical couplers, capacitors, transformers, and relays have been approved as providing safe separation. Only circuits rated for AC line voltage include safety isolation to other circuits. Isolation boundaries between 24 V DC circuits are functional only, and you should not depend on these boundaries for safety.

The sensor supply output, communications circuits, and internal logic circuits of an S7-200 SMART with included AC power supply are sourced as SELV (safety extra-low voltage) according to EN 61131-2.

To maintain the safe character of the S7-200 SMART low voltage circuits, external connections to communications ports, analog circuits, and all 24 V DC nominal power supply

and I/O circuits must be powered from approved sources that meet the requirements of SELV, PELV, Class 2, Limited Voltage, or Limited Power according to various standards.

WARNING

Safe use of power converters

Use of non-isolated or single insulation supplies to supply low voltage circuits from an AC line can result in hazardous voltages appearing on circuits that are expected to be touch safe, such as communications circuits and low voltage sensor wiring.

Such unexpected high voltages could result in death or serious injury to personnel, and/or damage to equipment.

Use only high-voltage-to-low-voltage power converters that are approved as sources of touch-safe, limited-voltage circuits.

Grounding guidelines

The best way to ground your application is to ensure that all the common and ground connections of your PLC and related equipment are grounded to a single point. This single point should be connected directly to the earth ground for your system.

All ground wires should be as short as possible and should use a large wire size, such as 2 mm^2 (14 AWG).

When locating grounds, remember to consider safety grounding requirements and the proper operation of protective interrupting devices.

Wiring guidelines

When designing the wiring for your S7-200 SMART CPU, provide a single disconnect switch that simultaneously removes power from the CPU power supply, from all input circuits, and from all output circuits. Provide over-current protection, such as a fuse or circuit breaker, to limit fault currents on supply wiring. Consider providing additional protection by placing a fuse or other current limit in each output circuit.

Install appropriate surge suppression devices for any wiring that could be subject to lightning surges.

Avoid placing low-voltage signal wires and communications cables in the same wire tray with AC wires and high-energy, rapidly switched DC wires. Always route wires in pairs, with the neutral or common wire paired with the hot or signal-carrying wire.

Use the shortest wire possible and ensure that the wire is sized properly to carry the required current.

Use wire and cable with a temperature rating 30 °C higher than the ambient temperature around the S7-200 SMART CPU (for example, a minimum of 85 °C-rated conductors for 55 °C ambient temperature). You should determine other wiring type and material requirements from the specific electrical circuit ratings and your installation environment.

Use shielded wires for optimum protection against electrical noise. Typically, grounding the shield at the S7-200 SMART CPU gives the best results. You should ground communication cable shields to S7-200 SMART CPU communication connector shells using connectors that

engage the cable shield, or by bonding the communication cable shields to a separate ground. You should ground other cable shields using clamps or copper tape around the shield to provide a high surface area connection to the grounding point.

When wiring input circuits that are powered by an external power supply, include an overcurrent protection device in that circuit. External protection is not necessary for circuits that are powered by the 24 V DC sensor supply from the S7-200 SMART CPU because the sensor supply is already current-limited.

All S7-200 SMART CPU modules have removable connectors for user wiring. To prevent loose connections, ensure that the connector is seated securely and that the wire is installed securely into the connector.

To help prevent unwanted current flows in your installation, the S7-200 SMART CPU provides isolation boundaries at certain points. When you plan the wiring for your system, you should consider these isolation boundaries. Refer to the technical specifications (Page 981) for the amount of isolation provided and the location of the isolation boundaries. Circuits rated for AC line voltage include safety isolation to other circuits. Isolation boundaries between 24 V DC circuits are functional only, and you should not depend on these boundaries for safety.

A summary of wiring rules for the S7-200 SMART CPUs, EMs, and SBs is shown below:

Table 5-11 Wiring rules for S7-200 SMART CPUs, EMs, and SBs

Wiring rules for...	CPU and EM connector	SB connector
Connectible conductor cross-sections for standard wires	2 mm^2 to 0.3 mm^2 (14 AWG to 22 AWG)	1.3 mm^2 to 0.3 mm^2 (16 AWG to 22 AWG)
Number of wires per connection	1 or combination of 2 wires up to 2 mm^2 (total)	1 or combination of 2 wires up to 1.3 mm^2 (total)
Wire strip length	6.4 mm	6.3 to 7 mm
Tightening torque* (maximum)	0.56 N-m (5 inch-pounds)	0.33 N-m (3 inch-pounds)
Tool	2.5 to 3.0 mm flathead screwdriver	2.0 to 2.5 mm flathead screwdriver

* To avoid damaging the connector, be careful that you do not over-tighten the screws.

Note

Ferrules or end sleeves on stranded conductors reduce the risk of stray strands causing short circuits. Ferrules longer than the recommended strip length should include an insulating collar to prevent shorts due to side movement of conductors. Cross-sectional area limits for bare conductors also apply to ferrules.

Guidelines for lamp loads

Lamp loads are damaging to relay contacts because of the high turn-on surge current. This surge current will nominally be 10 to 15 times the steady state current for a tungsten lamp. A replaceable interposing relay or surge limiter is recommended for lamp loads that will be switched a large number of times during the lifetime of the application.

Guidelines for inductive loads

Use suppressor circuits with inductive loads to limit the voltage rise when a control output turns off. Suppressor circuits protect your outputs from premature failure caused by the high voltage transient that occurs when current flow through an inductive load is interrupted.

In addition, suppressor circuits limit the electrical noise generated when switching inductive loads. High frequency noise from poorly suppressed inductive loads can disrupt the operation of the PLC. Placing an external suppressor circuit so that it is electrically across the load and physically located near the load is the most effective way to reduce electrical noise.

S7-200 SMART DC outputs include internal suppressor circuits that are adequate for inductive loads in most applications. Since S7-200 SMART relay output contacts can be used to switch either a DC or an AC load, internal protection is not provided.

A good suppressor solution is to use contactors and other inductive loads for which the manufacturer provides suppressor circuits integrated in the load device, or as an optional accessory. However, some manufacturer provided suppressor circuits may be inadequate for your application. An additional suppressor circuit may be necessary for optimal noise reduction and contact life.

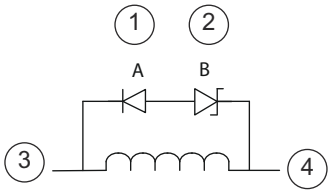
For AC loads, a metal oxide varistor (MOV) or other voltage clamping device may be used with a parallel RC circuit, but is not as effective when used alone. An MOV suppressor with no parallel RC circuit often results in significant high frequency noise up to the clamp voltage.

A well-controlled turn-off transient will have a ring frequency of no more than 10 kHz, with less than 1 kHz preferred. Peak voltage for AC lines should be within +/- 1200 V of ground. Negative peak voltage for DC loads using the PLC internal suppression will be ~40 V below the 24 V DC supply voltage. External suppression should limit the transient to within 36 V of the supply to unload the internal suppression.

Note

The effectiveness of a suppressor circuit depends on the application and must be verified for your particular usage. Ensure that all components are correctly rated and use an oscilloscope to observe the turn-off transient.

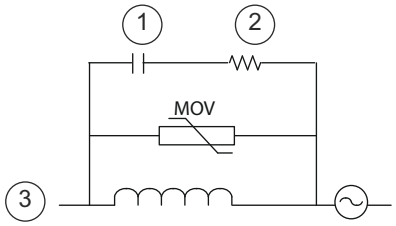
Typical suppressor circuit for DC or relay outputs that switch DC inductive loads



① 1N4001 diode or equivalent
② 8.2 V Zener (DC outputs), 36 V Zener (Relay outputs)
③ Output point
④ M, 24 V reference

In most applications, the addition of a diode (A) across a DC inductive load is suitable, but if your application requires faster turn-off times, then the addition of a zener diode (B) is recommended. Be sure to size your zener diode properly for the amount of current in your output circuit.

Typical suppressor circuit for relay outputs that switch AC inductive loads



① See table for C value
② See table for R value
③ Output point

Ensure that the working voltage of the metal oxide varistor (MOV) is at least 20% greater than the nominal line voltage.

Choose pulse-rated, non-inductive resistors, and capacitors recommended for pulse applications (typically metal film). Verify the components meet average power, peak power, and peak voltage requirements.

If you design your own suppressor circuit, the following table suggests resistor and capacitor values for a range of AC loads. These values are based on calculations with ideal component parameters. I rms in the table refers to the steady-state current of the load when fully ON.

Table 5-12 AC suppressor circuit resistor and capacitor values

Inductive load			Suppressor values		
I rms	230 V AC	120 V AC	Resistor		Capacitor
Amps	VA	VA	Ω	W (power rating)	nF
0.02	4.6	2.4	15000	0.1	15
0.05	11.5	6	5600	0.25	470
0.1	23	12	2700	0.5	100
0.2	46	24	1500	1	150
0.5	115	60	560	2.5	470
1	230	120	270	5	1000
2	460	240	150	10	1500

Conditions satisfied by the table values:
Maximum turn-off transition step < 500 V
Resistor peak voltage < 500 V
Capacitor peak voltage < 1250 V
Suppressor current < 8% of load current (50 Hz)
Suppressor current < 11% of load current (60 Hz)
Capacitor dV/dt < 2 V/µs
Capacitor pulse dissipation : $\int(dv/dt)^2dt < 10000\ V^2/\mu s$
Resonant frequency < 300 Hz
Resistor power for 2 Hz max switching frequency
Power factor of 0.3 assumed for typical inductive load

WARNING

Correct placement of external resistor/capacitor noise suppression circuit

When you use relay expansion modules to switch AC inductive loads, you must place the external resistor/capacitor noise suppression circuit across the AC load to prevent unexpected machine or process operation. Unexpected machine or process operation could result in death or severe personal injury.

Always be sure to follow these guidelines in placing the external resistor/capacitor noise suppression circuit.

PLC concept

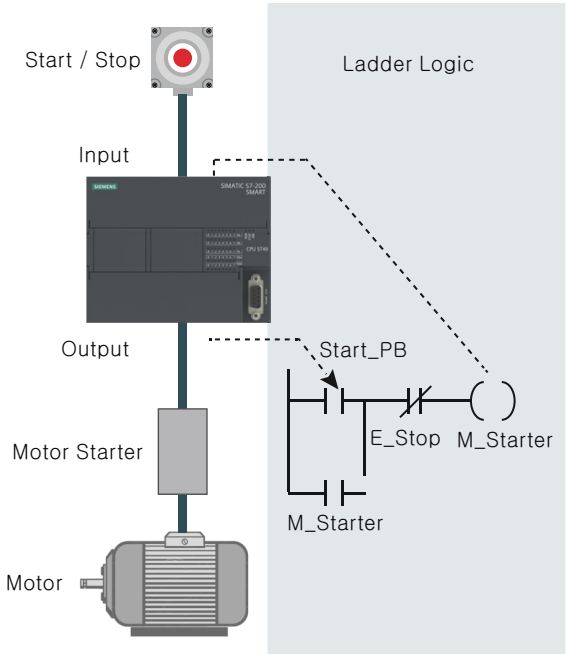
6

The basic function of the CPU is to monitor field inputs and, based on your control logic, turn on or off field output devices. This chapter explains the concepts used to execute your program, the various types of memory used, and how that memory is retained.

6.1 Execution of the control logic

The CPU continuously cycles through the control logic in your program, reading and writing data. The basic operation is very simple:

- The CPU reads the status of the inputs.
- The program that is stored in the CPU uses these inputs to evaluate the control logic.
- As the program runs, the CPU updates the data.
- The CPU writes the data to the outputs.

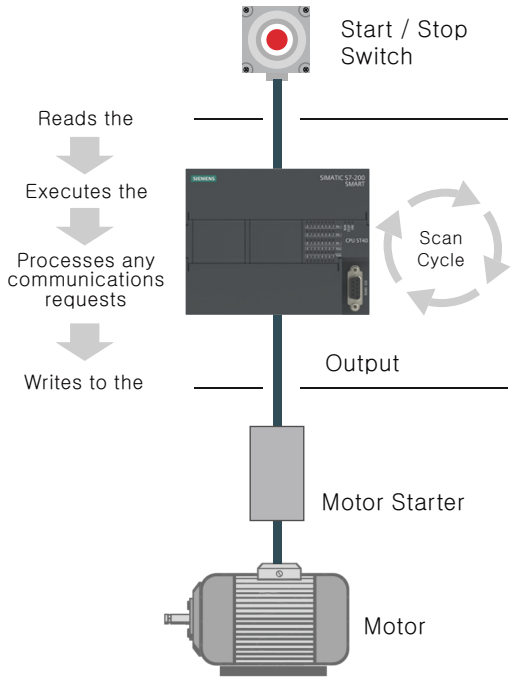


The figure shows a simple diagram of how an electrical relay diagram relates to the CPU. In this example, the state of the switch for starting the motor is combined with the states of other inputs. The calculations of these states then determine the state for the output that goes to the actuator which starts the motor.

Tasks in a scan cycle

The CPU executes a series of tasks repetitively. This cyclical execution of tasks is called the scan cycle. The execution of the user program is dependent upon whether the CPU is in STOP mode or in RUN mode. In RUN mode, your program is executed; in STOP mode, your program is not executed.

Table 6-1 Tasks performed by the CPU in a scan cycle

Scan cycle	Description
	Reading the inputs: The CPU copies the state of the physical inputs to the process image input register.
	Executing the control logic in the program: The CPU executes the instructions of the program and stores the values in the various memory areas.
	Processing any communications requests: The CPU performs any tasks required for communications.
	Executing the CPU self-test diagnostics: The CPU ensures that the firmware, the program memory, and any expansion modules are working properly.
	Writing to the outputs: The values stored in the process image output register are written to the physical outputs.

6.1.1 Reading the inputs and writing to the outputs

Reading the inputs

Digital inputs: Each scan cycle begins by reading the current value of the digital inputs and then writing these values to the process image input register.

Analog inputs: The CPU does not read the analog input values as part of the normal scan cycle. Instead, an analog value is read immediately from the device when your program accesses the analog input.

Writing to the outputs

Digital outputs: At the end of every scan cycle, the CPU writes the values stored in the process-image output register to the digital outputs.

Analog outputs: The CPU does not write analog output values as part of the normal scan cycle. Instead, the analog outputs are written immediately when your program accesses the analog output.

6.1.2 Executing the user program

During the execution phase of the scan cycle, the CPU executes your main program, starting with the first instruction and proceeding to the last instruction. The immediate I/O instructions give you immediate access to inputs and outputs during the execution of either the main program or an interrupt routine.

If you use subroutines in your program, the subroutines are stored as part of the program. The subroutines are executed when they are called by the main program, by another subroutine, by a function block, or by an interrupt routine. Subroutine or fucntion block nesting depth is 8 levels deep from the main and 4 levels deep from an interrupt routine.

If you use interrupts in your program, the interrupt routines that are associated with the interrupt events are stored as part of the program. The interrupt routines are not executed as part of the normal scan cycle, but are executed when the interrupt event occurs (which could be at any point in the scan cycle).

Local memory is reserved for each of 14 entities: the main program, eight subroutine or fucntion block nesting levels when initiated from the main program, one interrupt routine, and four subroutine nesting levels when initiated from an interrupt routine. Local memory has a local scope in that it is available only within its associated program entity, and cannot be accessed by the other program entities. For more information about Local memory, refer to Local Memory Area: L in this chapter.

The following figure depicts the flow of a typical scan including the Local memory usage and two interrupt events, one during the program-execution phase and another during the communications phase of the scan cycle. Subroutines are called by the next higher level, and are executed when called. Interrupt routines are not called; they are a result of an occurrence of the associated interrupt event.

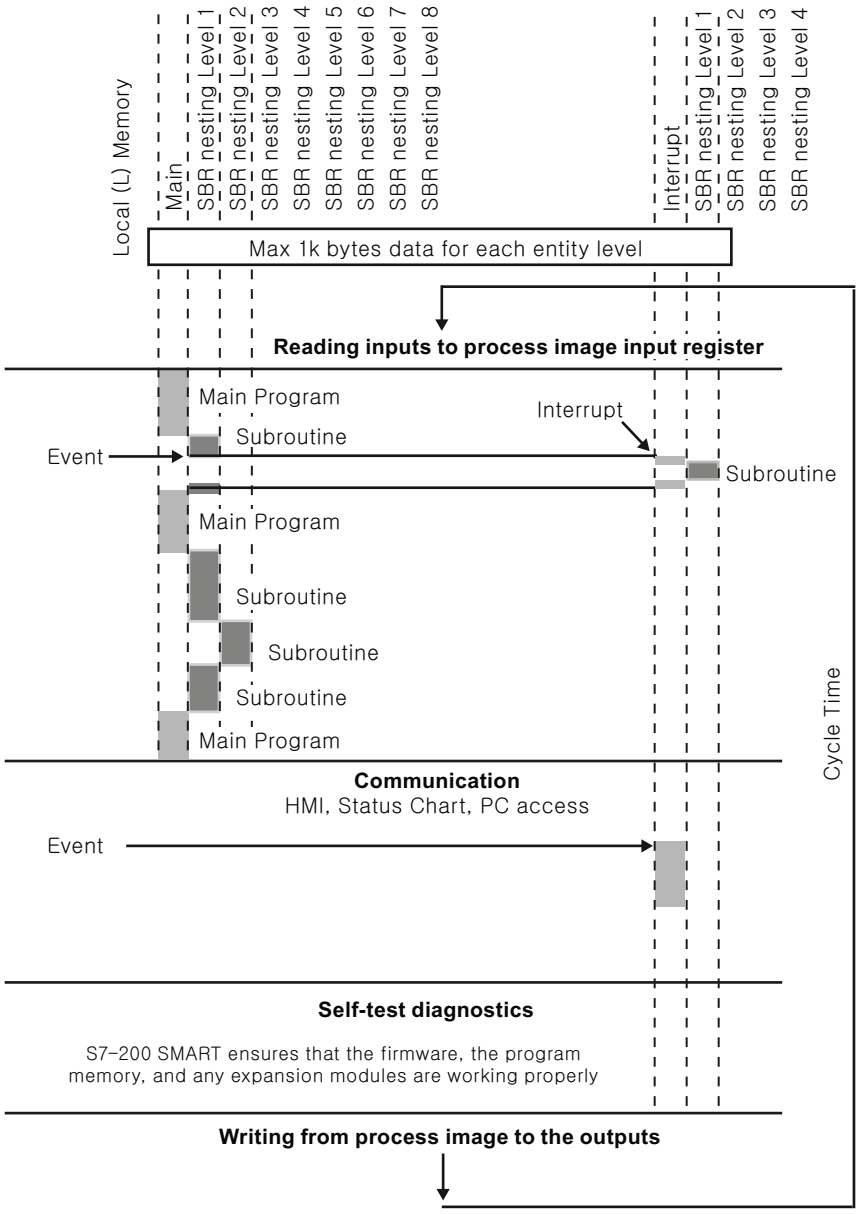


Figure 6-1 Typical_Scan_Flow

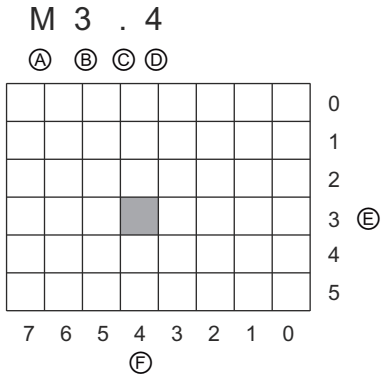
Note

The FB is called in the similar way as SBR.

6.2 Accessing data

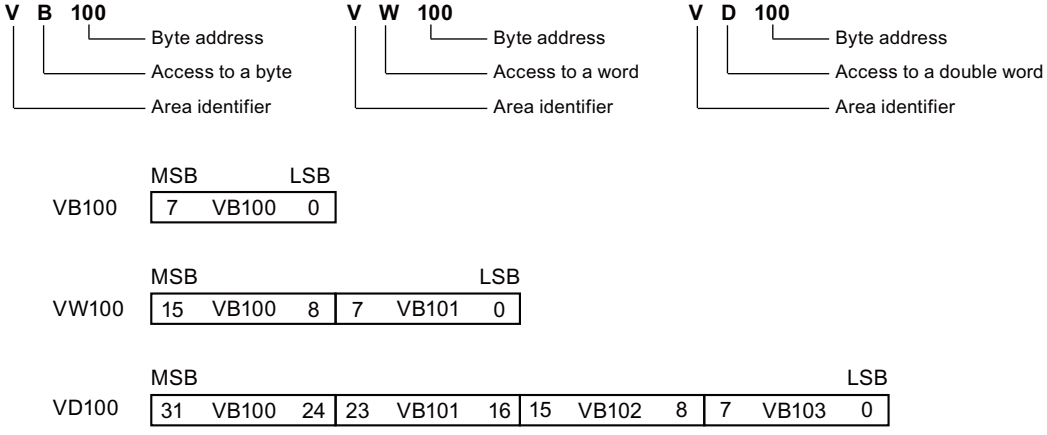
The CPU stores information in different memory locations that have unique addresses. You can explicitly identify the memory address that you want to access. This allows your program to have direct access to the information. To access a bit in a memory area, you specify the address, which includes the memory area identifier, the byte address, and the bit number (which is also called "byte.bit" addressing).

Table 6-2 Bit addressing

Elements of a bit address	Description	
	A	Memory area identifier
	B	Byte address: byte 3
	C	Separator ("byte.bit")
	D	Bit location of the byte (bit 4 of 8, bits numbered 7 to 0)
	E	Bytes of the memory area
	F	Bits of the selected byte

In this example, the memory area and byte address ("M3") designates byte 3 of M memory, with a period (".") to separate the bit address (bit 4).

You can access data in most memory areas (V, I, Q, M, S, L, and SM) as bytes, words, or double words by using the byte-address format. To access a byte, word, or double word of data in the memory, you must specify the address in a way similar to specifying the address for a bit. This includes an area identifier, data size designation, and the starting byte address of the byte, word, or double-word value, as shown in the following figure.



The following table shows the range of integer values that can be represented by the different sizes of data.

Table 6-3 Decimal and hexadecimal ranges for the different sizes of data

Representation	Byte (B)	Word (W)	Double Word (D)
Unsigned Integer	0 to 255 16#00 to 16#FF	0 to 65,535 16#0000 to 16#FFFF	0 to 4,294,967,295 16#00000000 to 16#FFFFFFFF
Signed Integer	-128 to +127 16#80 to 16#7F	-32,768 to +32,767 16#8000 to 16#7FFF	-2,147,483,648 to +2,147,483,647 16#8000 0000 to 16#7FFF FFFF
Real (IEEE 32-bit Floating Point)	*Not applicable*	*Not applicable*	+1.175495E-38 to +3.402823E+38 (positive) -1.175495E-38 to -3.402823E+38 (negative)

Data in other memory areas (such as T, C, HC, and the accumulators) are accessed by using an address format that includes an area identifier and a device number.

6.2.1 Accessing memory areas

I (process-image input)

The CPU samples the physical input points at the beginning of each scan cycle and writes these values to the process image input register. You can access the process image input register in bits, bytes, words, or double words:

Table 6-4 Absolute addressing for I memory

Bit:	I*[byte address].[bit address]*	I0.1
Byte, Word, or Double Word:	I*[size][starting byte address]*	IB4, IW7, ID20

Q (process-image output)

At the end of the scan cycle, the CPU copies the values stored in the process image output register to the physical output points. You can access the process image output register in bits, bytes, words, or double words:

Table 6-5 Absolute addressing for Q memory

Bit:	Q*[byte address].[bit address]*	Q1.1
Byte, Word, or Double Word:	Q*[size][starting byte address]*	QB5, QW14, QD28

You can use variable memory to store intermediate results of operations being performed by the control logic in your program.

V (variable memory)

You can also use variable memory to store other data pertaining to your process or task. You can access the variable memory area in bits, bytes, words, or double words:

Table 6-6 Absolute addressing for V memory

Bit:	V*[byte address].[bit address]*	V10.2
Byte, Word, or Double Word:	V*[size][starting byte address]*	VB16 VW100 VD2136

DB2 (variable memory)

In PLC programming, DB2 can only be accessed using Symbolic Addressing. But when communicating with HMI, the DB2 area must be accessed using absolute addressing.

Table 6-7 Absolute addressing for DB2 memory

Bit:	DB2.DBX[byte address].[bit address]	DB2.DBX10.2
Byte, Word, or Double Word:	DB2.DB[size][starting byte address]	DB2.DBB2 DB2.DBW4 DB2.DBD6

M (flag memory)

You can use the flag memory area (M memory) as internal control relays to store the intermediate status of an operation or other control information. You can access the flag memory area in bits, bytes, words, or double words:

Table 6-8 Absolute addressing for M memory

Bit:	M*[byte address].[bit address]*	M26.7
Byte, Word, or Double Word:	M*[size][starting byte address]*	MB0, MW11, MD20

T (timer memory)

The CPU provides timers that count increments of time in resolutions (time-base increments) of 1 ms, 10 ms, or 100 ms. Two variables are associated with a timer:

- Current value: this 16-bit signed integer stores the amount of time counted by the timer.
- Timer bit: this bit is set or cleared as a result of comparing the current and the preset value. The preset value is entered as part of the timer instruction.

You access both of these variables by using the timer address (T + timer number). Access to either the timer bit or the current value is dependent on the instruction used: instructions with bit operands access the timer bit, while instructions with word operands access the current value. As shown in the following figure, the Normally Open Contact instruction

accesses the timer bit, while the Move Word instruction accesses the current value of the timer.

Table 6-9 Absolute addressing for T memory

Timer:	T*[timer number]*	T24

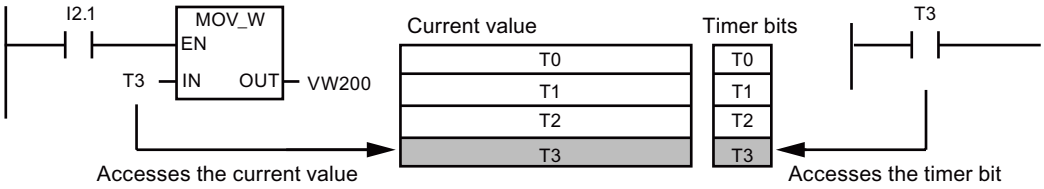


Figure 6-2 Accessing the timer bit or the current value of a timer

C (counter memory)

The CPU provides three types of counters that count each low-to-high transition event on the counter input(s): one type counts up only, one type counts down only, and one type counts both up and down. Two variables are associated with a counter:

- Current value: this 16-bit signed integer stores the accumulated count.
- Counter bit: this bit is set or cleared as a result of comparing the current and the preset value. The preset value is entered as part of the counter instruction.

You access both of these variables by using the counter address (C + counter number). Access to either the counter bit or the current value is dependent on the instruction used: instructions with bit operands access the counter bit, while instructions with word operands access the current value. As shown in the following figure, the Normally Open Contact instruction accesses the counter bit, while the Move Word instruction accesses the current value of the counter.

Table 6-10 Absolute addressing of C memory

Counter	C*[counter number]*	C24

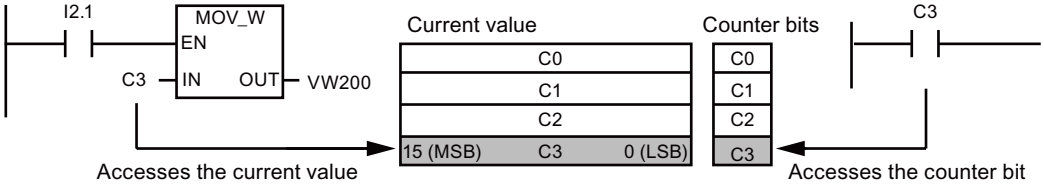


Figure 6-3 Accessing the counter bit or the current value of a counter

HC (high-speed counter)

The high-speed counters count high-speed events independent of the CPU scan. High-speed counters have a signed, 32-bit integer counting value (or current value). To access the count value for the high-speed counter, you specify the address of the high-speed counter, using the memory type (HC) and the counter number. The current value of the high-speed counter is a read-only value and can be addressed only as a double word (32 bits).

Table 6-11 Absolute addressing of HC memory

High-speed counter	HC*[high-speed counter number]*	HC1

AC (accumulators)

The accumulators are read/write devices that can be used like memory. For example, you can use accumulators to pass parameters to and from subroutines and to store intermediate values used in a calculation. The CPU provides four 32-bit accumulators (AC0, AC1, AC2, and AC3). You can access the data in the accumulators as bytes, words, or double words.

The size of the data being accessed is determined by the instruction that is used to access the accumulator. As shown in the following figure, you use the least significant 8 or 16 bits of the value that is stored in the accumulator to access the accumulator as bytes or words. To access the accumulator as a double word, you use all 32 bits.

For information about how to use the accumulators within interrupt subroutines, refer to the Interrupt instructions (Page 393).

Table 6-12 Absolute addressing of AC memory

Accumulator	AC*[accumulator number]*	AC0

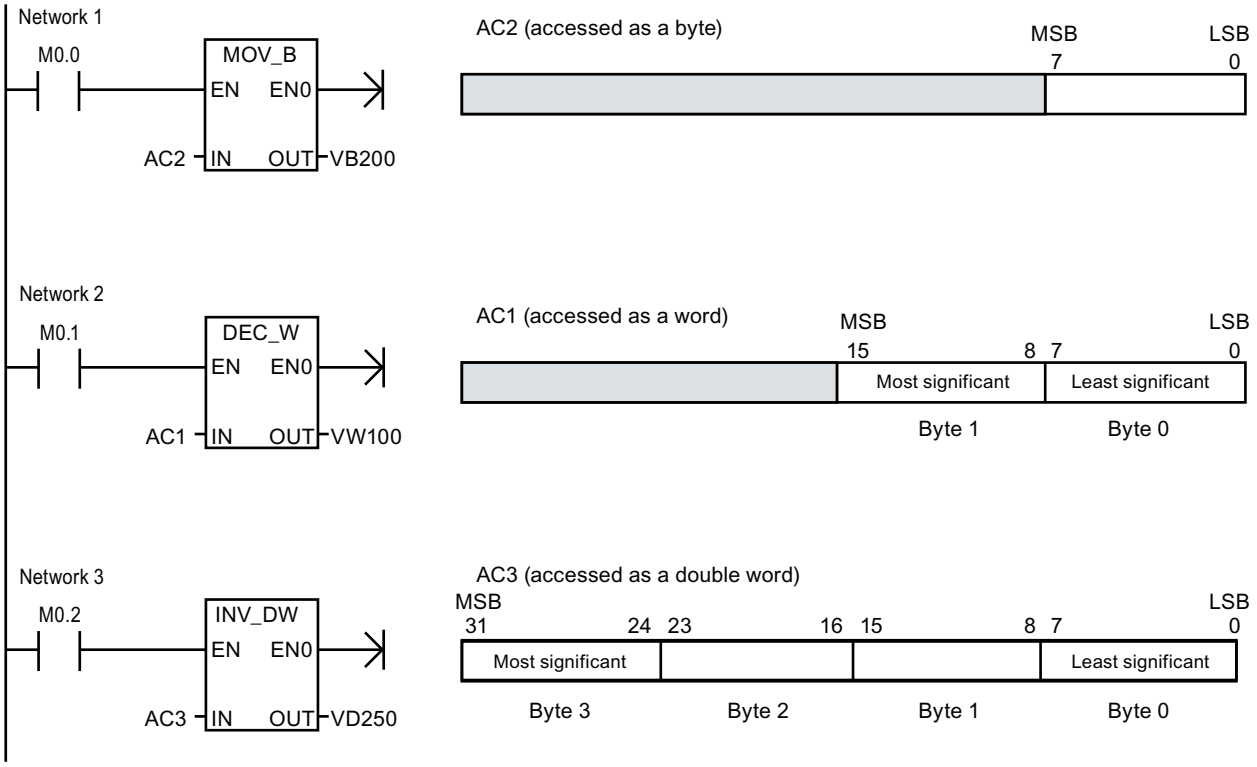


Figure 6-4 Accessing the accumulators

SM (special memory)

The SM bits provide a means for communicating information between the CPU and your user program. You can use these bits to select and control some of the special functions of the CPU, such as: a bit that turns on for the first scan cycle, a bit that toggles at a fixed rate, or a bit that shows the status of math or operational instructions. You can access the SM bits as bits, bytes, words, or double words:

Table 6-13 Absolute addressing of SM memory

Bit:	SM*[byte address].[bit address]*	SM0.1
Byte, Word, or Double Word:	SM*[size][starting byte address]*	SMB86, SMW300, SMD1000

For more information, see the descriptions of the SM bits (Page 1067).

L (local memory area)

The CPU provides 64 L memory bytes for each POU (program organizational unit) in a local memory stack. A POU's associated L memory addresses are accessible only by the currently executing POU (main, subroutine, or interrupt routine). When you use interrupt routines and subroutines, the L memory stack is used to preserve L memory values of a POU that temporarily suspends execution, so another POU can execute. The suspended POU can then resume execution with the L memory values that existed prior to giving execution control to another POU.

L memory stack maximum nesting limits:

- Eight subroutine nesting levels when initiated from the main program
- Four subroutine nesting levels when initiated from an interrupt routine

The nesting limits allow a 14 level execution stack in your program. For example, the main program (level 1) has eight nested subroutines (levels 2 to 9). During execution of the 9th level subroutine, an interrupt occurs (level 10). The interrupt routine contains four nested subroutines (levels 11 to 14).

L memory rules:

- You can use L memory for local scratchpad "TEMP" variables in all POU types (main, subroutine, and interrupt routines)
- Only subroutines can use L memory for "IN" IN_OUT", and "OUT" variable types that are passed to or from subroutines.
- If you are programming a subroutine in either LAD or FBD, only 60 bytes are allowed for TEMP, IN, IN_OUT, and OUT variables. STEP 7-Micro/WIN SMART V3 uses the last four bytes of local memory

Local memory symbols, variable types, and data types are assigned in the Variable table that is available when the associated POU is opened in the program editor. Absolute L memory addresses are automatically assigned when a POU is successfully compiled.

In most cases, use L memory symbol name references in your program logic, because you cannot know all the absolute L memory addresses until after the complete POU is successfully compiled. However, you can use absolute L memory addresses as shown in the following table.

Table 6-14 Absolute addressing of L memory

Bit:	L*[byte address].[bit address]*	L0.0
Byte, Word, or Double Word:	L*[size] [starting byte address]*	LB33, LW5, LD20

Local memory and to global V memory use a similar address syntax, but V memory has a global scope while L memory has a local scope. Global scope means that the same memory address can be accessed from any POU. Local scope means that the L memory allocation is associated with a particular POU and cannot be accessed by another program unit.

The local scope of L memory also affects symbol usage, when a global symbol and a local symbol use the same name. If your program logic references that symbol name, the CPU ignores the global symbol and processes the address assigned to the local memory symbol.

Note

Local memory value assignments are not always preserved for successive executions of a POU

L memory addresses are reused for the next execution sequence, after the current nested sequence is completed. Depending on a POU's level in the execution stack and L memory assignments made since a POU's last execution, a POU's L memory assignments made in a previous execution may be overwritten with unexpected values.

Remember to reassign the correct values to L memory variables, in your program logic. Reinitialize all TEMP values before processing them and ensure that any output values (OUT and IN_OUT) are correct.

AI (analog input)

The CPU converts an analog value (such as temperature or voltage) into a word-length (16-bit) digital value. You access these values by the area identifier (AI), size of the data (W), and the starting byte address. Since analog inputs are words and always start on even-number bytes (such as 0, 2, or 4), you access them with even-number byte addresses (such as AIW0, AIW2, or AIW4). Analog input values are read-only values.

Table 6-15 Absolute addressing of AI memory

Analog input	AIW*[starting byte address]*	AIW4

AQ (analog output)

The CPU converts a word-length (16-bit) digital value into a current or voltage, proportional to the digital value (such as for a current or voltage). You write these values by the area identifier (AQ), size of the data (W), and the starting byte address. Since analog outputs are words and always start on even-number bytes (such as 0, 2, or 4), you write them with even-number byte addresses (such as AQW0, AQW2, or AQW4). Analog output values are write-only values.

Table 6-16 Absolute addressing of AQ memory

Analog output	AQW*[starting byte address]*	AQW4

S (sequence control relay)

S bits are associated with SCRs, which you can use to organize machine or steps into equivalent program segments. SCRs allow logical segmentation of the control program. You can access the S memory as bits, bytes, words, or double words.

Table 6-17 Absolute addressing of S memory

Bit:	S*[byte address].[bit address]*	S3.1
Byte, Word, or Double Word:	S*[size][starting byte address]*	SB4, SW7, SD14

6.2.2 Format for Real numbers

Real (or floating-point) numbers are represented as 32-bit, single-precision numbers, whose format is described in the ANSI/IEEE 754-1985 standard. Real numbers are accessed in double-word lengths.

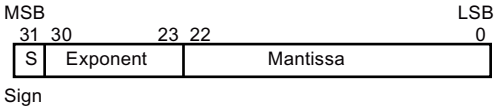


Figure 6-5 Format of a Real number

Note

Floating-point numbers are accurate up to 6 decimal places. Therefore, you can specify a maximum of 6 decimal places when entering a floating-point constant.

Calculations that involve a long series of values including very large and very small numbers can produce inaccurate results. This can occur if the numbers differ by 10 to the power of *x*, where *x* > 6. For example: 100 000 000 + 1 = 100 000 000

6.2.3 Format for strings

A string is a sequence of characters, with each character being stored as a byte. The first byte of the string defines the length of the string, which is the number of characters. The following figure shows the format for a string. A string can have a length of 0 to 254 characters, plus the length byte, so the maximum length for a string is 255 bytes. A string constant is limited to 126 bytes.

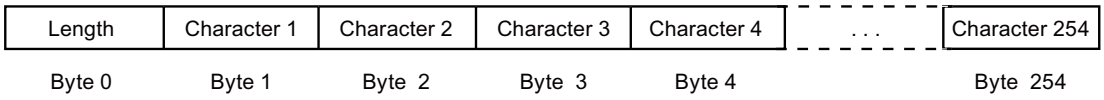


Figure 6-6 Format for strings

6.2.4 Assigning a constant value for instructions

You can use a constant value in many of the programming instructions. Constants can be bytes, words, or double words. The CPU stores all constants as binary numbers, which can then be represented in decimal, hexadecimal, ASCII, or real number (floating point) formats.

Table 6-18 Representation of constant values

Representation	Format	Sample
Decimal	[decimal value]	20047
Hexadecimal	16#[hexadecimal value]	16#4E4F
Binary	2#[binary number]	2#1010_0101_1010_0101
ASCII	'[ASCII text]'	'ABCD'
Real	ANSI/IEEE 754-1985	+1.175495E-38 (positive) -1.175495E-38 (negative)
String	"[stringtext]"	"ABCDE"

Note

The CPU does not support "data typing" or data checking (such as specifying that the constant is stored as an integer, a signed integer, or a double integer). For example, an Add instruction can use the value in VW100 as a signed integer value, while an Exclusive Or instruction can use the same value in VW100 as an unsigned binary value.

6.2.5 Addressing the local and expansion I/O

The local I/O provided by the CPU provides a fixed set of I/O addresses. You can add I/O points by connecting expansion I/O modules to the right side of the CPU or by installing a signal board. The addresses of the points of the module are determined by the type of I/O and the position of the module in the chain. For example, an output module does not affect the addresses of the points on an input module, and vice versa. Likewise, analog modules do not affect the addressing of digital modules, and vice versa.

Note

Process image register space for digital I/O is always reserved in increments of eight bits (one byte). If a module does not provide a physical point for each bit of each reserved byte, these unused bits cannot be assigned to subsequent modules in the I/O chain. For input modules, the unused bits are set to zero with each input update cycle.

Analog I/O points are always allocated in increments of two points. If a module does not provide physical I/O for each of these points, these I/O points are lost and are not available for assignment to subsequent modules in the I/O chain.

The following table provides an example of the fixed mapping convention (established by STEP 7 Micro/WIN SMART and downloaded as part of the I/O configuration, in the system block).

Table 6-19 CPU mapping convention

	CPU	Signal board	Expansion module 0	Expansion module 1	Expansion module 2	Expansion module 3	Expansion module 4	Expansion module 5
Starting address	I0.0 Q0.0	I7.0 Q7.0 AI12 AQ12	I8.0 Q8.0 AI16 AQ16	I12.0 Q12.0 AI32 AQ32	I16.0 Q16.0 AI48 AQ48	I20.0 Q20.0 AI64 AQ64	I24.0 Q24.0 AI80 AQ80	I28.0 Q28.0 AI96 AQ96

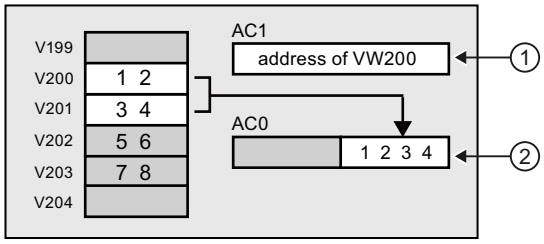
6.2.6 Using pointers for indirect addressing

Indirect addressing uses a pointer to access data in memory. Pointers are double word memory locations that contain the address of another memory location. You can only use V memory locations, L memory locations, or accumulator registers (AC1, AC2, AC3) as pointers. To create a pointer, you must use the Move Double Word instruction to move the address of the indirectly addressed memory location to the pointer location. Pointers can also be passed to a subroutine as a parameter.

An S7-200 SMART CPU allows pointers to access the following memory areas: I, Q, V, M, S, AI, AQ, SM, T (current value only), and C (current value only). You cannot use indirect addressing to access an individual bit or to access HC, L or accumulator memory areas.

To indirectly access the data in a memory address, you create a pointer to that location by entering an ampersand character (&) and the first byte of the memory location to be addressed. The input operand of the instruction must be preceded with an ampersand (&) to signify that the address of a memory location, instead of its contents, is to be moved into the location identified in the output operand of the instruction (the pointer).

Entering an asterisk (*) in front of an operand for an instruction specifies that the operand is a pointer. As shown in the following figure, entering *AC1 means that AC1 stores a pointer to the word-length value being referenced by the Move Word (MOVW) instruction. In this example, the values stored in both VB200 and VB201 are moved to accumulator AC0.



① MOVD &VB200, AC1

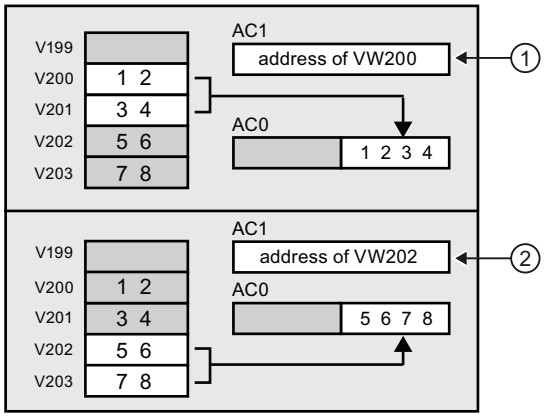
Creates the pointer by moving the address of VB200 (initial byte of VW200) to AC1

② MOVW *AC1, AC0

Moves the word value referenced by the pointer in AC1

Figure 6-7 Creating and using a pointer

As shown in the following figure, you can change the value of a pointer. Since pointers are 32-bit values, use double-word instructions to modify pointer values. Simple mathematical operations, such as adding or incrementing, can be used to modify pointer values.



① MOVD &VB200, AC1

Creates the pointer by moving the address of VB200 (initial byte of VW200) to AC1

MOVW *AC1, AC0

Moves the word value referenced by the pointer in AC1

② +D +2, AC1

Adds 2 to the accumulator to point to the next word location

MOVW *AC1, AC0

Moves the word value referenced by the pointer in AC1

Figure 6-8 Modifying a pointer

Note

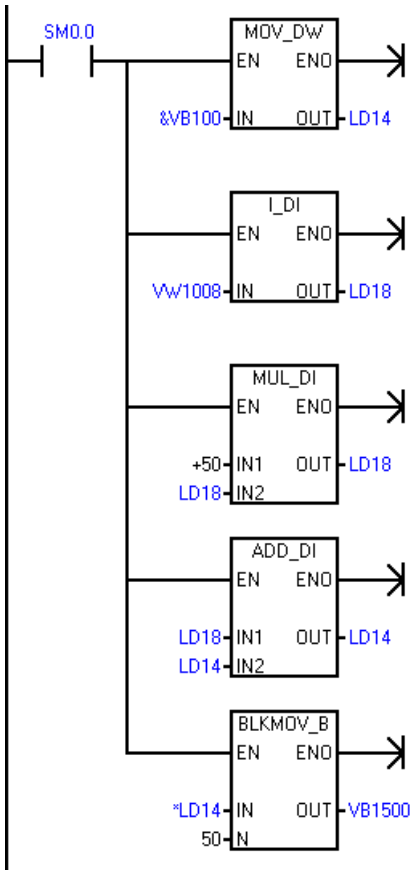
When modifying the value of a pointer, remember to adjust for the size of the data that you are accessing: to access a byte, increment the pointer value by 1; to access a word or a current value for a timer or counter, add or increment the pointer value by 2; and to access a double word, add or increment the pointer value by 4.

6.2.7 Pointer examples

Using a pointer to access data in a table

This example uses LD14 as a pointer to a recipe stored in a table of recipes that begins at VB100. In this example, VW1008 stores the index to a specific recipe in the table. If each recipe in the table is 50 bytes long, you multiply the index by 50 to obtain the offset for the starting address of a specific recipe. By adding the offset to the pointer, you can access the individual recipe from the table. In this example, the recipe is copied to the 50 bytes that start at VB1500.

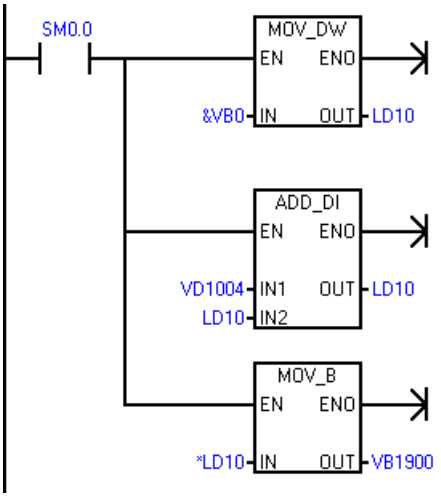
Table 6-20 Example: Using a pointer to access data in a table

LAD		STL
	To transfer a recipe from a table of recipes: • Each recipe is 50 bytes long. • The index parameter (VW1008) identifies the recipe to be loaded. Create a pointer to the starting address of the recipe table. Convert the index of the recipe to a double-word value. Multiply the offset to accommodate the size of each recipe. Add the adjusted offset to the pointer. Transfer the selected recipe to VB1500 through VB1549	**Network 1** **LD SM0.0** **MOVD &VB100, LD14** **ITD VW1008, LD18** ***D +50, LD18** **+D LD18, LD14** **BMB *LD14, VB1500, 50**

Using an offset to access data

This example uses LD10 as a pointer to the address VB0. You then increment the pointer by an offset stored in VD1004. LD10 then points to another address in V memory (VB0 + offset). The value stored in the V memory address pointed to by LD10 is then copied to VB1900. By changing the value in VD1004, you can access any V memory location.

Table 6-21 Example: Using an offset to read the value of any V memory location

LAD		STL
	Load the starting address of the V memory to a pointer. Add the offset value to the pointer. Copy the value from the V memory location (offset) to VB1900	**Network 1** **LD SM0.0** **MOVD &VB0, LD10** **+D VD1004, LD10** **MOVB *LD10, VB1900**

6.3 Saving and restoring data

6.3.1 Downloading project components

Note

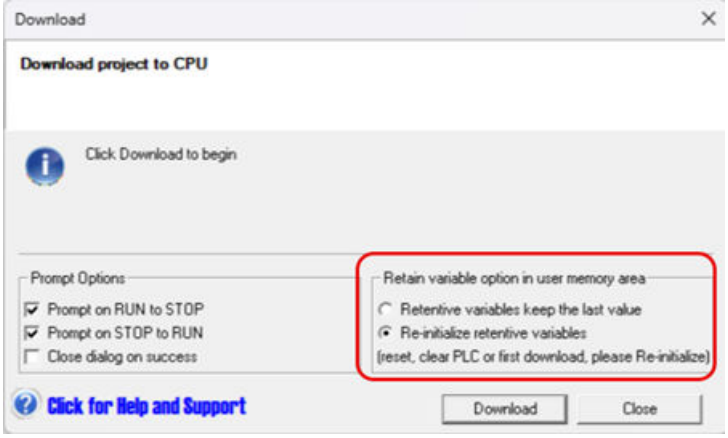
Downloading a program to the CPU completely overwrites any pre-existing program in the CPU. Be sure that you want to overwrite the program before performing a download.

To download project components from STEP 7-Micro/WIN SMART V3 to the CPU, follow these steps:

1. Ensure that your Communication Interface and PLC connector cable for either Ethernet (Page 71) or RS485 (Page 74) communications is working, and that PLC communication is operating properly.
2. Login to the PLC whose project you want to download.
   If you're not logged in, you'll be notified when you click the download button.
3. Place the CPU in STOP mode (Page 82).

4. To download all project components, click the Download button from the Transfer area of the File or PLC menu ribbon strip, or alternatively press the shortcut key combination **CTRL+D**.

5. To download selected project components, click the down arrow under the Download button, and then select the specific project component you want to download (Project or User Defined Web (Page 236)) from the drop-down list.
6. After clicking the Download button, if you see a Communications dialog, select the Communication Interface and the Ethernet IP address or RS485 network address for the PLC to which you want to download.
7. From the Download dialog, set the download options for the blocks, and whether you want to be prompted on CPU transitions from RUN to STOP mode (Page 82) and STOP to RUN mode (Page 82).
8. Optionally click the "Close dialog on success" check box if you want the dialog to automatically close after a successful download.
9. Click the Download button.



Note

The feature marked in the figure is only effective for the V and DB2 retention ranges.

STEP 7-Micro/WIN SMART V3 copies the complete program and the following program components to the CPU. The status icon indicates informational messages, or whether

potential problems or errors occurred with the download. The status message provides specific results of the operation.

Note

You can only download a program created for use in STEP 7 Micro/WIN V3 to an S7-200 SMART 3.0 version CPU.

You can only download a program created for use in STEP 7 Micro/WIN V2.8 or earlier to a S7-200 SMART V2.8 version or earlier CPUs.

You can download a maximum of 4 MB of project information to an S7-200 SMART 3.0 version CPU and if this is exceeded, you will be warned that the project is too large and the download is prevented.

STEP 7-Micro/WIN SMART V3 also supports program edit and download in RUN mode.

What happens when you download

STEP 7-Micro/WIN SMART V3 and the CPU perform the following tasks in sequence on your project components when you download:

Step	Action	Related topics, additional description
1.	Project components in the program editors serve as input for the download operation, based on the download objects you selected. The program editors can include new program data that you've entered, a saved and opened .smartV3 project, or an uploaded ASCII import file.	File open Range checking Project file I/O errors Program editor errors
2.	STEP 7-Micro/WIN SMART V3 compile A compile or download command starts the compiler. If the compile passes, control passes to the next step; if not, the compile or download operation exits.	All STEP 7-Micro/WIN SMART V3 compiler errors are listed in the Output Window. Double-click the error and the editor scrolls to the error location. A successful compile shows the resulting block size of the program and data block.
3.	Send blocks to CPU across communication network for PLC compile.	Communication Errors To download (Editor to PLC) or upload (PLC to Editor), PLC communication must be operating properly. Make sure your network hardware and PLC connector cable are working.
4.	PLC compile If PLC compile succeeds, control passes to the next step; if not, download exits with error(s).	The PLC Compiler verifies that the PLC hardware supports all program instructions, ranges, and structure. Click the PLC button from the Information area of the PLC menu to view the first compile error found.
5.	Program is in CPU permanent memory and ready to be executed in RUN mode.	Fatal Errors (Page 1065) and non-fatal run-time errors (Page 1062) are accessible from the Information area of the PLC menu.

If the download attempt produces compiler errors or download errors, correct the errors and reattempt the download.

See also

Uploading project components

See also

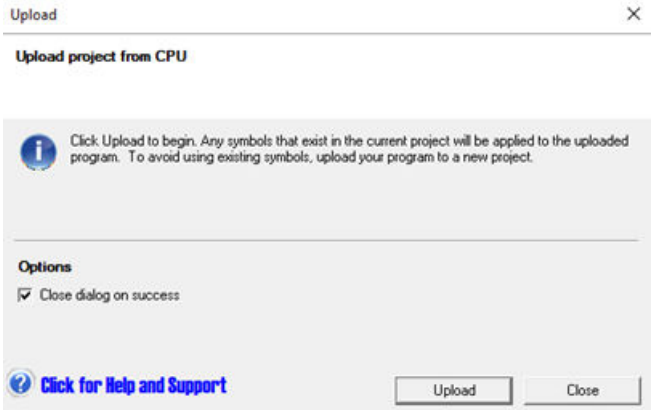
Establishing the RS485 hardware communication connection (Page 74)

6.3.2 Uploading project components

To upload project components from the PLC to a STEP 7-Micro/WIN SMART V3 program editor, follow these steps:

1. Ensure that your network hardware and PLC connector cable (Ethernet or RS485) are working, and that PLC communication is operating properly (Page 748).
2. Login to the PLC whose project you want to upload.
   If you're not logged in, you'll be notified when you click the Upload button.
3. To upload all project components, click the Upload button from the Transfer section of the File or PLC menu ribbon strip, or press the shortcut key combination **CTRL+U.**

4. To upload selected project components, click the down arrow under the Upload button, and then select the specific project component you want to upload (Project, Data Logs).
5. If you see a Communications dialog, select the Communication Interface and the Ethernet IP address or RS485 network address of the PLC from which you want to upload.
6. Optionally click the "Close dialog on success" check box if you want the dialog to automatically close after a successful upload.
7. Click the "Upload" button to start the upload.



STEP 7-Micro/WIN SMART V3 copies the complete program and the following program components from the PLC to the currently open project. The status icon indicates informational messages, or whether potential problems or errors occurred with the upload. The status message provides specific results of the operation.

If the upload is successful, you can save the uploaded program, or make further changes.

Note

You cannot use an earlier version of STEP 7-Micro/WIN SMART to upload a project to a version 3.0 CPU.

Note

Uploading into a new project is a risk-free way to capture the program block, system block, and/or data block information. Since the project is empty, you cannot inadvertently destroy data.

Uploading into an existing project is useful if you want to overwrite all modifications that have been made to the program since it was downloaded (Page 124) to the PLC. Uploading into an existing project does, however, overwrite any additions or modifications you have made to the project. Use this option, only if you want to completely overwrite your STEP 7-Micro/WIN SMART V3 project with the project stored in the PLC.

STEP 7-Micro/WIN SMART V3 does not upload comments, but if you currently have a program with comments open in the program editor, the comments are retained. Take care if uploading over an existing project and use this method only if the projects are similar.

6.3.3 Types of storage

The CPU provides a variety of features to ensure that your user program and data are properly retained.

- Retentive memory: selectable areas of memory that remain unchanged over a power cycle. Retentive memory can be configured in the system data block. V, M, and current values to timers and counters are the only memory areas that can be configured to be retentive.
- Permanent memory: memory used to store the program block, data block, system block, forced values, as well as values configured to be retentive.
- Memory card: removable microSDHC card for standard CPUs that you can use for the following purposes:
  - To store the projects blocks as a program transfer card (Page 131)
  - To completely erase the PLC as a restore-to-factory-defaults card (Page 210)
  - To update the PLC and expansion module firmware as a firmware update card (Page 129)

6.3.4 Using a memory card

Using a memory card

The standard S7-200 SMART CPUs support the use of a microSDHC card for:

- User program transfer (Page 131)
- Reset CPU to factory default condition (Page 210)
- Firmware update of the CPU and attached expansion modules as supported

You can use any standard, commercial microSD card with a capacity up to 128 GB.

The following CPU behaviors are common, regardless of the memory card usage:

1. Inserting a memory card into a CPU in RUN mode causes the CPU to automatically transition to STOP mode.
2. A CPU cannot advance to RUN mode if a memory card is inserted.
3. Memory card evaluation is performed only after a CPU power-up or warm restart. Therefore, program transfer and firmware update can only occur after a CPU power-up or warm restart.
4. The memory card can be used to store files and folders not related to program transfer and firmware update usage as long as their names do not conflict with the file and folder names used for program transfer and firmware update usage.

WARNING

Verify that the CPU is not actively running a process before installing the memory card.

Installing the memory card will cause the CPU to go to STOP mode, which could affect the operation of an online process or machine. Unexpected operation of a process or machine could result in death or injury to personnel and/or property damage.

Before inserting the memory card, always ensure that the CPU is offline and in a safe state.

Program transfer card

You can use a memory card to transfer user program content into the CPU permanent memory, completely or partially replacing content already in the load memory.

To be used for program transfer purposes, the memory card is organized as follows:

Table 6-22 Memory card used for program transfer card

At the root level of the card	
File: S7_JOB.S7S	A text file containing the word TO_ILM
Folder: SIMATIC.S7S	A folder containing user program files to be transferred to the CPU

Reset to factory defaults card

You can use a memory card to erase all retained data, putting the CPU back into a factory default condition.

To be used for reset to factory default (Page 210) purposes, the memory card is organized as follows:

Table 6-23 Memory card used to reset to factory defaults

At the root of the card	
File: S7_JOB.S7S	A text file containing the word RESET_TO_FACTORY

Firmware update card

You can use a memory card to update the firmware in a CPU and any connected expansion modules.

The file and folder organization of a firmware update memory card is as follows:

Table 6-24 Memory card used for firmware update purposes

At the root level of the card	
File: S7_JOB.S7S	A text file containing the word FWUPDATE
Folder: FWUPDATE.S7S	A folder containing update files (.upd) for each device to be updated

After power-up, if the CPU detects the presence of a memory card, it locates and opens the S7_JOB.SYS file on the card. If the CPU discovers the FWUPDATE string in that file, then the CPU enters a firmware update sequence.

The CPU examines each update file (.upd) in the FWUPDATE.S7S folder and if the order ID contained in the update file name matches the order ID (MLFB) of a connected device (CPU, expansion module or signal board), then the CPU updates the firmware of that device with the firmware content contained within the update file.

Note

Firmware update from STEP 7-Micro/WIN SMART V3

You can also perform a firmware update from STEP 7-Micro/WIN SMART V3 using the RS485 port. Especially for CPU models that do not have a memory card, this method is valuable. Refer to the PLC menu section of the STEP 7-Micro/WIN SMART V3 online help for instructions.

6.3.5 Inserting a memory card in a standard CPU

Table 6-25 Inserting and removing a memory card in a standard CPU

Task	Procedure
	Follow the steps below to insert the microSDHC memory card into the CPU. 1. Open the bottom terminal block connector cover. 2. Insert the microSDHC memory card in the memory card slot (marked X50) located above the terminal block connectors. 3. Replace the terminal block connector cover after inserting the card to ensure that the card is secure. Follow the steps below to remove the microSDHC memory card from the CPU. 1. Open the bottom terminal block connector cover. 2. Grasp the microSDHC memory card from the CPU and pull it out of the card slot (marked Micro-SD X50). 3. Replace the bottom terminal block cover.

6.3.6 Transferring your program with a memory card

The standard S7-200 SMART CPU models support standard, commercial microSD/microSDHC/microSDXC cards with a capacity ranging from 4 GB to 128 GB using the FAT32 file system format. You can use a microSD/microSDHC/microSDXC card as a program transfer card for portable storage for your program and project data.

Note

Card Requirements:

- Cards larger than 32GB must be formatted to FAT32 using third-party tools because the Windows operating system does not support formatting cards larger than 32GB to FAT32.
- Card speed class ≤ Class10/U1 (that is, 10MB/s)

WARNING

Verify that the CPU is not running a process before inserting the memory card.

Inserting a memory card into a CPU in RUN mode causes the CPU to automatically transition to STOP mode.

Inserting a memory card into a running CPU can cause disruption to process operation, possibly resulting in death or severe personal injury.

Always ensure that the CPU is in STOP mode (Page 82) prior to inserting a memory card.

Creating a program transfer memory card on the PLC

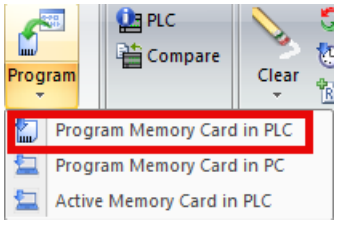
Note

STEP 7-Micro/WIN SMART V3 first erases any SIMATIC content on the card prior to transferring the program to the card. Any other data on the card that you've stored using a card reader and Windows Explorer is left undisturbed.

Note also that you cannot change the CPU to RUN mode if a memory card is inserted.

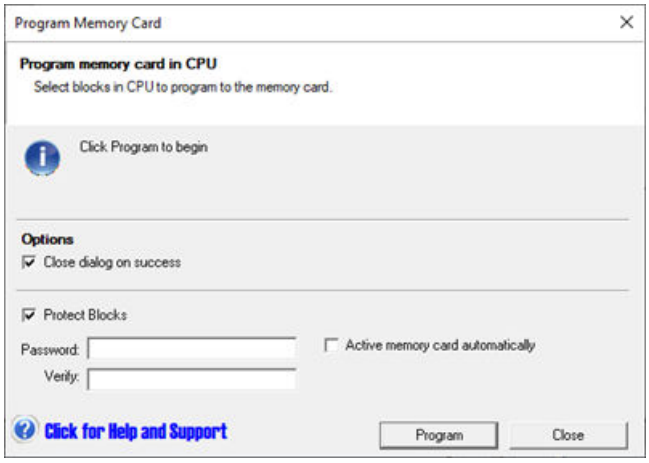
To program the memory card as a program transfer card, follow these steps:

1. Ensure that your network hardware and PLC connector cable are working, the CPU is powered on and in STOP mode, and that PLC communication is operating properly (Page 71).
2. If not already inserted, insert a microSDHC memory card in the CPU. You can insert or remove the memory card while the CPU is powered on.
3. Download (Page 82) the program to the PLC, if not already downloaded.
4. Click the down arrow under the Setting button in the PLC menu ribbon strip, and then select "Program Memory Card in PLC".



5. Select "Protect Data" if you want to set a password for activating the memory card.

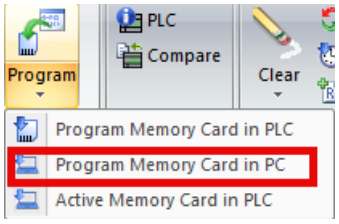
6. Click the Setting button.
7. Enter the password (Page 49) if one is required for programming the memory card.



Creating a program transfer memory card on the PC

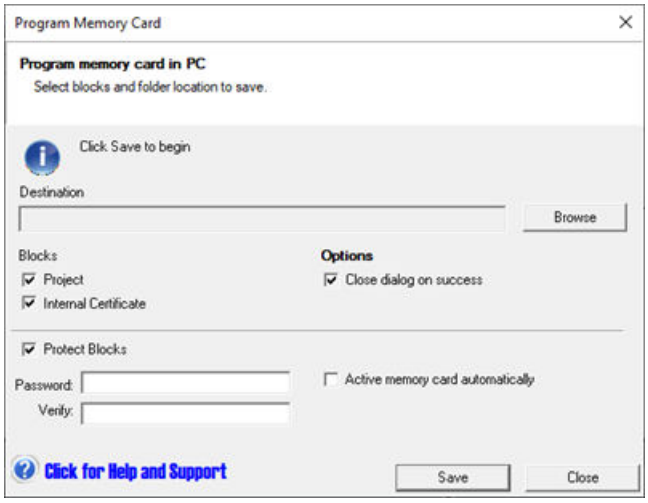
To save the program on the computer, follow the steps:

1. Click the down arrow under the Setting button and then select "Program Memory Card in PC".



2. Select which of the following blocks (or all) to store on the memory card:
   – Project
   – Internal Certificate

3. Click "Browse" button and navigate to the root folder of the SD card.

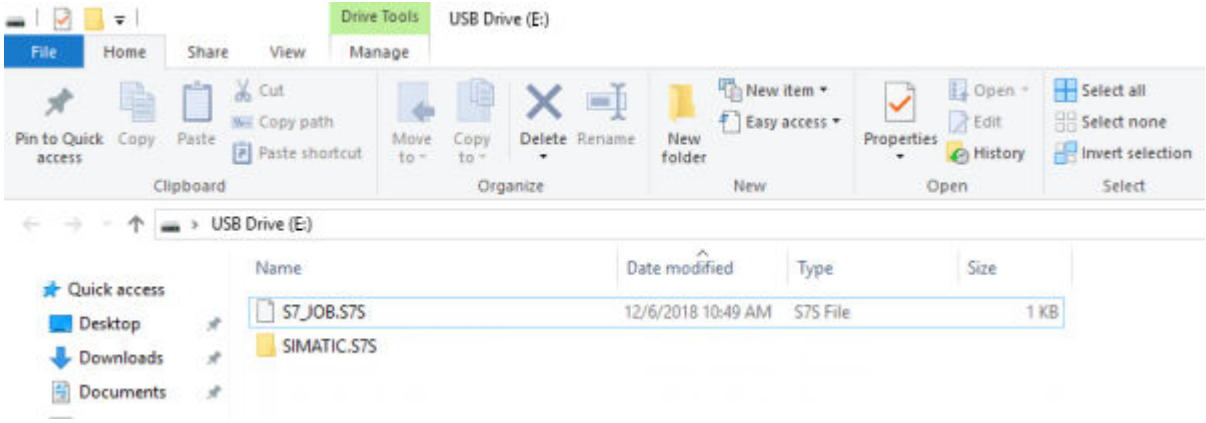


4. Select "Protect Data" if you want to set a password for activating the memory card.

Note

If you select "Protect Data", you need to activate the memory card later.

5. Click the Save button and two files are generated in your SD card as follows:



Note

The PLC might fail to compile the program restored from an SD card by "Program Memory Card in PC". To ensure your program/configuration is valid, Siemens recommends that you connect to a PLC and download the program at least once.

Note

The memory card created in PC doesn't contain any force configuration in the program.

Activating memory card in PLC

If you select the "Protect Data" when you create the card, activate the card as followt steps.

1. Insert the memory card into the PLC.
2. Click the down arrow under the Setting button and then select "Activate Memory card in PLC".
3. Enter the password you set when you create the card.

Restoring the program from a program transfer memory card

To copy the contents of the program transfer card to the PLC, you must cycle the power to the CPU with the program transfer card inserted. The CPU then performs the following tasks:

1. Clears RAM
2. Copies the user program, the system block (PLC configuration), and the data block from the memory card to CPU permanent memory.

While the copy operation is in progress, the STOP and RUN LEDs on the S7-200 SMART CPU alternately flash. When the S7-200 SMART CPU completes the copy operation, the LEDs stop flashing.

Note

Program transfer card compatibility

Restoring a program transfer card that you created on a different CPU model might fail due to the differences in model types. During the restore process, the CPU validates the following characteristics of the program content stored on the memory card:

- Size of program block
- Size of V memory specified in the data block
- Quantity of onboard digital I/O configured in the system block (Page 181)
- Each retentive range that is configured in the system block
- Expansion module and signal board configurations in the system block
- Axis of Motion configurations in the system block
- Forced memory locations

Note

When you transfer a program created by STEP 7-Micro/WIN SMART V3 or later, note the following:

- Web server and internal certificate configuration created in STEP 7-Micro/WIN SMART V3 cannot be transferred to CPU whose version is smaller than 3.0.
- If the internal certificate is selected, the internal certificate will be generated in target CPU. The validated period is one year since the card making.
- If web server is disabled, internal certificate is grey and can't be store.

Note

In addition to using a memory card as a program transfer card, you can also create a reset-to-factory-defaults memory card (Page 210).

6.3.7 Restoring data after power on

The CPU performs the following actions after a power cycle:

- Restores the program block and the system block from permanent memory
- Restores the retentive memory assignments
- Restores the non-retentive portions of V memory from the contents of the data block in permanent memory
- Clears the non-retentive portions of other memory areas

6.4 Changing the operating mode of the CPU

The CPU has three modes of operation: STOP mode, offline mode and RUN mode. The status LEDs on the front of the CPU indicates the current mode of operation. In STOP mode, the CPU is not executing the program, and you can download program blocks. In RUN mode, the CPU is executing the program; however, you can download program blocks.

Placing the CPU in RUN mode

1. Click the "RUN" button on either the PLC menu ribbon strip or on the program editor toolbar:
2. When prompted, click "OK" to change the operating mode of the CPU.

You can monitor the program in STEP 7-Micro/WIN SMART V3 by clicking the "Program Status" button from the "Debug" menu ribbon strip, or from the program editor toolbar. STEP 7-Micro/WIN SMART V3 displays the values for the instructions.

Placing the CPU in offline mode

1. Click the "offline" button on either the PLC menu ribbon strip or on the program editor toolbar:
2. When prompted, click "OK" to change the operating mode of the CPU.

You can monitor the program in STEP 7-Micro/WIN SMARTV3 by clicking the "Program Status" button from the "Debug" menu ribbon strip, or from the program editor toolbar. STEP 7-Micro/WIN SMART V3 displays the values for the instructions.

Placing the CPU in STOP mode

To stop the program, click the "STOP" button and acknowledge the prompt to place the CPU in STOP mode. You can also place a STOP instruction (Page 393) in your program logic to put the CPU in STOP mode.

6.5 Status LEDs

The CPU and EM use LEDs to provide information about operational status.

CPU status LEDs

The CPU provides the following LED status indicators:

State	LED Behavior	Description
STOP	STOP: on RUN, ERROR: off	Applicable when the CPU is in STOP Mode
STOP with Force	RUN: off STOP: blink at 1 Hz rate ERROR: off	Applicable when the CPU is in STOP mode and a value is forced
STOP with PROFINET	STOP: on RUN: off ERROR: blink at 1 Hz rate	Applicable when the CPU is in STOP mode and any configured PROFINET device loses connection or gets alarms.
RUN	RUN: on STOP, ERROR: off	Applicable when the CPU is in RUN Mode
RUN with Force	RUN: on STOP: blink at 1 Hz rate ERROR: off	Applicable when the CPU is in RUN mode and a value is forced
RUN with PROFINET	STOP: off RUN: on ERROR: blink at 1 Hz rate	Applicable when the CPU is in RUN mode and any configured PROFINET device loses connection or gets alarms.
Busy	STOP, RUN: blink out of phase at 2 Hz rate ERROR: off	Applicable when after card evaluation during power-up or restart, a memory card operation is in progress or when a restart operation is in progress
Memory Card Inserted	STOP: blinks at 2 Hz rate RUN, ERROR: off	Applicable when someone inserts a memory card into a powered-on CPU
Memory Card OK	STOP: blinks at 2 Hz rate RUN, ERROR: off	Applicable when after card evaluation during power-up or restart, a memory card operation completes successfully
Memory Card Error	STOP, ERROR: blink in phase at 2 Hz rate RUN: off	Applicable when after card evaluation during power-up or restart, a memory card operation terminates with an error
Defective	STOP, ERROR: on RUN: off	Applicable when the CPU is in Defective Mode
Ping	STOP, RUN: blink out of phase at 1 Hz rate ERROR: blink in phase with RUN	Applicable when the CPU receives a Signal DCP control request (Flash LEDs)

EM status LEDs

The expansion modules (EM) provide the following LED status indicators:

Each digital EM provides a DIAG LED that indicates the status of the module:

- Green indicates that the module is operational
- Red indicates that the module is defective or non-operational

Each analog EM provides an I/O Channel LED for each of the analog inputs and outputs.

- Green indicates that the channel has been configured and is active
- Red indicates an error condition of the individual analog input or output

In addition, each analog EM provides a DIAG LED that indicates the status of the module:

- Green indicates that the module is operational
- Red indicates that the module is defective or non-operational

The EM DP01 has a different set of LEDs. See LED status indicators for the EM DP01 PROFIBUS DP (Page 543).

The EM detects the presence or absence of power to the module (field-side power, if required).

Table 6-26 Status LEDs for a expansion module (EM)

Description	**DIAG (Red / Green)**	**I/O Channel (Red / Green)**
Field-side power is off *	Flashing red	Flashing red
Not configured or update in progress	Flashing green	Off
Module configured with no errors	On (green)	On (green)
Error condition	Flashing red	-
I/O error (with diagnostics enabled)	-	Flashing red
I/O error (with diagnostics disabled)	-	On (green)

* Status is only supported on analog signal modules.

Programming concepts

7

7.1 Guidelines for designing a PLC system

There are many methods for designing a PLC system. The following general guidelines can apply to many design projects. Of course, you must follow the directives of your own company's procedures and the accepted practices of your own training and location.

Partition your process or machine

Divide your process or machine into sections that have a level of independence from each other. These partitions determine the boundaries between controllers and influence the functional description specifications and the assignment of resources.

Create the functional specifications

Write the descriptions of operation for each section of the process or machine. Include the following topics: I/O points, functional description of the operation, states that must be achieved before allowing action for each actuator (such as solenoids, motors, and drives), description of the operator interface, and any interfaces with other sections of the process or machine.

Design the safety circuits

Identify equipment requiring hard-wired logic for safety. Control devices can fail in an unsafe manner, producing unexpected startup or change in the operation of machinery. Where unexpected or incorrect operation of the machinery could result in physical injury to people or significant property damage, consideration should be given to the use of electro-mechanical overrides which operate independently of the CPU to prevent unsafe operations.

The following tasks should be included in the design of safety circuits:

- Identify improper or unexpected operation of actuators that could be hazardous.
- Identify the conditions that would assure the operation is not hazardous, and determine how to detect these conditions independently of the CPU.
- Identify how the CPU and I/O affect the process when power is applied and removed, and when errors are detected. This information should only be used for designing for the normal and expected abnormal operation, and should not be relied on for safety purposes.
- Design manual or electro-mechanical safety overrides that block the hazardous operation independent of the CPU.
- Provide appropriate status information from the independent circuits to the CPU so that the program and any operator interfaces have necessary information.
- Identify any other safety-related requirements for safe operation of the process.

Specify the operator stations

Based on the requirements of the functional specifications, create drawings of the operator stations. Include the following items:

- Overview showing the location of each operator station in relation to the process or machine
- Mechanical layout of the devices, such as display, switches, and lights, for the operator station
- Electrical drawings with the associated I/O of the CPU or expansion module

Create the configuration drawings

Based on the requirements of the functional specification, create configuration drawings of the control equipment. Include the following items:

- Overview showing the location of each CPU in relation to the process or machine
- Mechanical layout of the CPU and expansion I/O modules (including cabinets and other equipment)
- Electrical drawings for each CPU and expansion I/O module (including the device model numbers, communications addresses, and I/O addresses)

Create a list of symbolic names (optional)

If you choose to use symbolic names for addressing, create a list of symbolic names for the absolute addresses. Include not only the physical I/O signals, but also the other elements to be used in your program.

7.2 Elements of the user program

A program organizational unit (POU) is composed of executable code and comments. The executable code consists of a main program, any function blocks and any subroutines or interrupt routines. The code is compiled and downloaded to the CPU. You can use the program organizational units (main program, subroutines, function blocks and interrupt routines) to structure your user program.

- The main body of the user program contains the instructions that control your application. The CPU executes these instructions sequentially, once per scan cycle.
- Subroutines are optional elements of your program which are executed only when called: by the main program, by an interrupt routine, or by another subroutine. Subroutines are useful in cases where you want to execute a function repeatedly. Rather than rewriting the logic for each place in the main program where you want the function to occur, you can write the logic once in a subroutine and call the subroutine as many times as needed during the main program. Subroutines provide several benefits:
  - Using subroutines reduces the overall size of your program.
  - Using subroutines decreases your scan time because you have moved the code out of the main program. The CPU evaluates the code in the main program every scan cycle, whether the code is executed or not, but the CPU evaluates the code in the subroutine only when you call the subroutine, and does not evaluate the code during the scans in which the subroutine is not called.
  - Using subroutines creates code that is portable. You can isolate the code for a function in a subroutine, and then copy that subroutine into other programs with little or no rework.

    Note

    Using V memory addresses can limit the portability of your subroutine, because it is possible for V memory address assignment from one program to conflict with an assignment in another program. Subroutines that use the local variable table (L memory) for all address assignments, by contrast, are highly portable because there is no concern about address conflicts between the subroutine and another part of the program when using local variables.

- Function blocks are reusable code blocks with associated memory. You typically use an FB to control the operation for tasks or devices that do not finish their operation within one scan cycle. To store the operating parameters so that they can be quickly accessed from one scan to the next, each FB in your user program has one or more FB instances. When you call an FB, you also specify an FB instance that contains the block parameters and the static local data for that call or "FB instance". The FB instance maintains these values after the FB finishes execution.By designing the FB for generic control tasks, you can reuse the FB for multiple devices by selecting different FB instances for different calls of the FB.An FB stores the Input, Output, and InOut, and Static parameters in an FB instance.

- Interrupt routines are optional elements of your program that react to specific interrupt events. You design an interrupt routine to handle a pre-defined interrupt event. Whenever the specified event occurs, the CPU executes the interrupt routine.
  The interrupt routines are not called by your main program. You associate an interrupt routine with an interrupt event, and the CPU executes the instructions in the interrupt routine only on each occurrence of the interrupt event.

  Note

  Because it is not possible to predict when the CPU might generate an interrupt, it is desirable to limit the number of variables that are used both by the interrupt routine and elsewhere in the program.

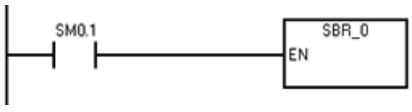
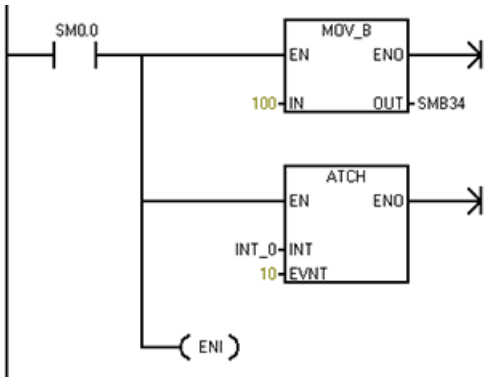
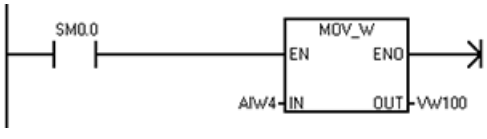
  Use the local variable table of the interrupt routine to ensure that your interrupt routine uses only the temporary memory and does not overwrite data used somewhere else in your program.

  There are a number of programming techniques you can use to ensure that data is correctly shared between your main program and the interrupt routines. Refer to the descriptions of the Interrupt instructions (Page 393).

- Other blocks contain information for the CPU.
  - System Block: The system block allows you to configure different hardware options for the CPU.
  - Data Block: The DB stores the initial values for different variables used by your program.

The following example shows a program that includes a subroutine and an interrupt routine. This sample program uses a timed interrupt for reading the value of an analog input every 100 ms.

Table 7-1 Sample program with a subroutine and an interrupt routine

Main		**Network 1** **LD SM0.1** **CALL SBR_0**	On first scan, call subroutine 0.
SBR 0		**Network 1** **LD SM0.0** **MOVB 100, SMB34** **ATCH INT_0, 10** **ENI**	Set the interval to 100 ms for the timed interrupt. Enable interrupt 0.
INT 0		**Network 1** **LD SM0.0** **MOVW AIW4,VW100**	Sample the value of analog input AI4.

7.3 Creating your user program

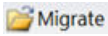
The STEP 7-MicroWIN SMART V3 user interface provides a convenient working space for creating your user project program. (STEP 7-MicroWIN SMART V3 projects are files with a .smartV3 file name extension.)

To open the user interface, double-click the STEP 7-MicroWIN SMART V3 icon, or select "STEP 7-MicroWIN SMART V3" from the "Siemens Automation" element on the "Start" menu.

7.3.1 Earlier versions of STEP 7-Micro/WIN projects

To work with a project that was created in STEP 7-Micro/WIN SMART V2.8, follow these steps:

- Click the Migrate button from the Operations section of the File menu ribbon strip, or alternatively press the Ctrl+M shortcut key combination.



Note

Only projects created in STEP 7-Micro/WIN SMART V2.8 can be migrated to V3.0. For a project created in a STEP 7-Micro/WIN SMART version earlier than V2.8, you need to open and save it in V2.8 and then migrate it to V3.0.

Note

STEP 7-Micro/WIN SMART V3 tries maintaining the normal functionality of the original program as much as possible after migration. Pay attention to the messages in the output window.

Note

Rename the symbol names of variable names that conflict with newly added reserved keywords

If the symbol names in the symbol table or the variable names in the variable table of the migrated project conflict with keywords introduced in version 3.0 (including newly added system POU names, new instruction names and newly reserved keywords), these variables turn red to indicate that there is an error after the migration process. Please make sure to rename them in time to avoid any problems.

CPU

Note

STEP 7-Micro/WIN SMART V3 does not support CR/CRs CPUs, 20 and 30 CPUs

STEP 7-Micro/WIN SMART V3 does not support CR/CRs CPUs, 20 and 30 CPUs. For V2.8 projects to be migrated that contain these CPUs, a dialog box will pop up to inform you that STEP 7-Micro/WIN SMART V3 does not currently support 20 and 30 series CPUs. If you want to migrate this project, use STEP 7-Micro/WIN SMART V2.8 to convert it into a CPU project with 40 and 60 series and then use STEP 7-Micro/WIN SMART V3 to migrate it.

After the migration is complete, the CPU version will be updated to 3.0.

POU

Name

- If the SBR/INT names in the migrated project conflict with the newly added system POU names in version 3.0, they will be renamed to "OriginalName_SBR", add _SBR to original name, after migration. If duplicates still exist after renaming, they will be renamed to "OriginalName_SBR_ID##".

Local memory

- LB 60 - LB 64 in the original project is mapped to LB 1020 - LB 1024 in the migrated project.

Instruction

In STEP 7-Micro/WIN SMART V3, the following instructions support three parameters in STL and the original instruction names are changed to "original name(V2)", add (V2) to original name.

- Add, subtract, multiply, and divide (Page 362)
- Multiply integers to double integer and divide integer with remainder (Page 366)
- AND, OR, XOR, NAND, NOR, and NXOR (Page 407)
- Shift and rotate (Page 428)

The legacy two-parameter instructions are located at the end of the corresponding folder in the project tree and are named “Legacy”.

Security setting

For security settings after migration, see the table below.-

STEP 7-Micro/WIN SMART V2.8		STEP 7-Micro/WIN SMART V3.0	
System block -> Security	• Full Privileges	PLC Security Setting	Disable PLC Access Control
	• Read Privileges • Minimum Privileges • Disallow Upload		Enable PLC Access Control • Read Privileges • Minimum Privileges • Disallow Upload
None		PLC Security Setting	Communication Settings • Put/Get enabled by default • Secure communication with HMI disabled by default

Note the following points:

- After Migration, System Block->Communication->LLDP is enabled.
- After Migration, System Block->Communication->SNMP is enabled.

Migrating symbol tables to global variable tables

When you migrate a project created in STEP 7-Micro/WIN SMART V2.8 to V3.0, the symbols, constants, and variables will be transferred as below:

- The symbol table with constant symbols is split into two tables, with constants being placed in the constant table and the rest being placed in the global variable table. Which means, If there are constant symbols in the symbol table "Table_1" in the project need to be migrate, in STEP 7-Micro/WIN SMART V3, a new table called "Table_1_Const" are added. Then, the constant symbols from "Table_1" are moved to "Table_1_Const".
- The system symbol table is migrated with the same name.
- The migrated system variable table only contains the new system variables that are added in version 3.0. You can restore the missing system symbols with the function "Update system variable table".
- If there is an overlap or complete match between the memory defined by the DB (valid DB line) and the variables in the global variable table (valid variable address), and if the initial value of the global variable is 0, the Initial value cell of the variable are disabled. In this case, the user can only set the initial value of the variable in the DB.
- The table property and editable fields have consistent editable properties as before the migration.
- The data type of the variable is empty.

After migrating, symbol table will be transferred as below:

Symbol table_xx in STEP 7-Micro/WIN SMART V2.8	Global variable table in STEP 7-Micro/WIN SMART V3.0	
symbol name	variable name	symbol table_xx
address	Bind	
comment	comment for variable	
constant symbol	constant name	constant table_xx[1]

[1] A default constant table is created if there are no constants in the original program.

Note

- The initial value for variables is 0. Only for the user level variable tables and library level variable tables, the default value is displayed and editable.
- When editing DB and variable, system will check if there is any address overlap between the two, and if there is, the variable cell cannot be edited.
- After migrating the symbols to variables, the ValidFlag remains unchanged (underlined or highlighted).

FB instance table

After migrating, a default FB instance table is created.

Local variable

- If the local variable table of the original project's SBR contains string variables, the variable is migrated as STRING<254>. When this SBR is called, if the operand corresponding to the String variable is a variable, the variable must be STRING<254>.

Wizard

- **Motion**: STEP 7-Micro/WIN SMART V3 allows you Migrate a V2.8 project to STEP 7-Micro/WIN SMART V3 and modify it either with motion wizard or the new motion control. For more information, go Open loop motion control(Legacy) (Page 879).
- **HSC**: The project with HSC created in Step7 Micro/win smart V2.8 can be opened and used directly in Step7 Micro/win smart V3 as before.

- **PROFINET**
  - After migration, the I-Device will support two Ethernet ports.
  - If the "PROFINET Interface- Parameters Allocated by Upper-Level- Controller" checkbox is selected in the I-Device, please note the following: The version of the IO device configured in the Profinet controller must match the version of the actual connected- I-Device device.
  - If the original program configured I-Device, description of I-Device updated to CPU 3.0.
- **Web server**

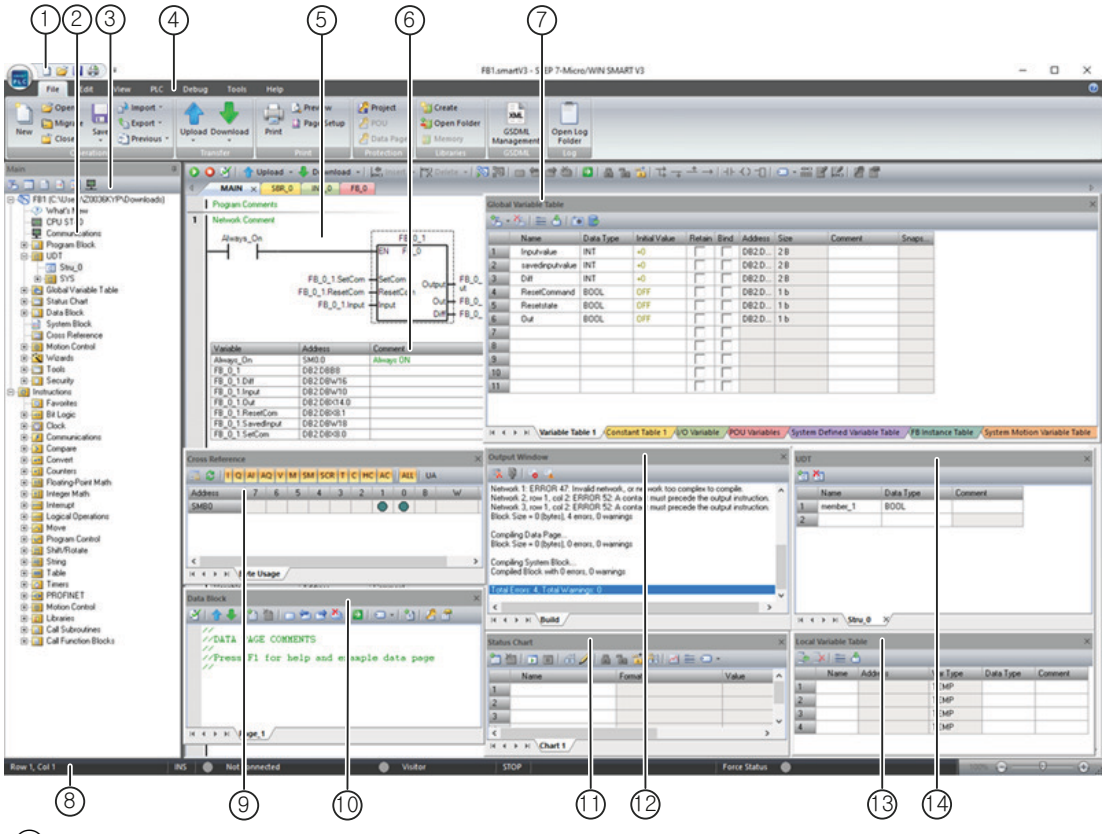
Note

Web server users are lost after migration

When a project in Micro/WIN SMART V2.8 configured web server users, after migration, the users are lost.

7.3.2 Using STEP 7-Micro/WIN SMART V3 user interface

The STEP 7-Micro/WIN SMART V3 user interface appears below. Note that each editing window can be docked or floated and arranged on the screen as you choose. You can display each window separately, as shown below, or you can combine windows such that each one is accessible from a separate tab:



① Quick access toolbar (Page 149)
② Project tree (Page 149)
③ Navigation bar (Page 149)
④ Menus (Page 149)
⑤ Program editor (Page 149)
⑥ Variable information table
⑦ Global variable table (Page 156)
⑧ Status bar (Page 149)
⑨ Cross reference window (Page 149)
⑩ Data block (Page 742)
⑪ Status chart (Page 149)
⑫ Output window (Page 154)
⑬ Local variable table (Page 737)
⑭ UDT (Page 164)

7.3.3 Using STEP 7-Micro/WIN SMART V3 to create your programs

Quick access toolbar

The quick access toolbar appears just above the menu tabs. The quick access file button provides quick and easy access to most of the functions of the File menu, and to recent documents. The other buttons on the quick access toolbar are for the file functions New, Open, Save, and Print.

Project tree

The project tree displays all of the project objects and the instructions for creating your control program. You can drag and drop individual instructions from the tree into your program, or you can double-click an instruction to insert it at the current location of the cursor in the program editor.

The Project tree organizes your project:

- Right-click the project to set a project password or set project options
- Right-click the Program Block folder to insert new subroutines, function blocks and interrupt routines.
- Open the Program Block folder and right-click a POU to open the POU, edit its properties, password-protect it, or rename it.
- Right-click a Status Chart or Global Table folder to insert new charts or tables.
- Open the Status Chart or Global Table folder and right-click the icon in the instruction tree or double-click the appropriate POU tab to open it, rename it, or delete it.

Note

Increased security for project, POU, and data block (data page) passwords

STEP 7-Micro/WIN SMART V2.3 increased the security for passwords from previous versions. If you are working with a project that you created with a previous version of STEP 7-Micro/WIN SMART V3, re-enter your passwords to activate the increased security.

Navigation bar

The Navigation bar appears at the top of the project tree and provides quick access to objects on the project tree. Clicking a navigation bar button is equivalent to expanding the project tree and clicking the same selection. The navigation bar presents groups of icons for accessing different programming features of STEP 7-Micro/WIN SMART V3.

Menu ribbon strips

STEP 7-Micro/WIN SMART V3 displays a menu ribbon strip for each menu. You can minimize the menu ribbon strip to save space by right-clicking in the menu ribbon strip area and selecting "Minimize the Ribbon".

Program editor

The program editor contains the program logic and a variable table where you can assign symbolic names for temporary program variables. Subroutines, Function blocks and interrupt routines appear as tabs at the top of the program editor window. Click the tabs to move between the subroutines, interrupts, and the main program.

STEP 7-Micro/WIN SMART V3 provides three editors for creating your program:

- Ladder logic (LAD)
- Statement list (STL)
- Function block diagram (FBD)

With some restrictions, programs written in any of these program editors can be viewed and edited with the other program editors.

You can change the editor to LAD, FBD, or STL from the Editor section of the View menu ribbon strip. You can configure the default editor at startup from the Options button of the Settings area of the Tools menu ribbon strip.

Status bar

The status bar, which is located at the bottom of the main window, provides information on the editing mode or online status operations that you perform in STEP 7-Micro/WIN SMART V3.

Output window

The Output Window keeps a list of the most recently compiled POUs (Page 843) and any errors that occurred during the compilation. If you have the Program Editor window open as well as the Output Window, you can double-click an error message in the Output Window to scroll your program automatically to the network where the error is located.

7.3.4 Using wizards to help you create your control program

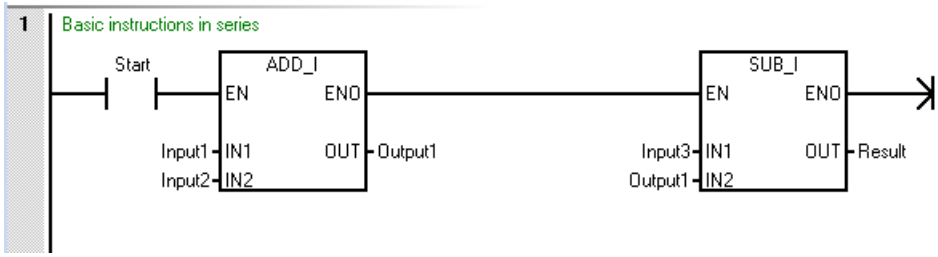
STEP 7-Micro/WIN SMART V3 provides the following wizards to make aspects of your programming easier and more automatic:

- High speed counter
- Motion (V2.x)
- PID
- PWM (Pulse Width Modulation)
- Text Display
- Get/Put
- Data Log (standard CPUs only)
- PROFINET
- Web server

To start a wizard, select that wizard from the STEP 7-Micro/WIN SMART V3 Tools menu ribbon strip or from the Wizards node in the project tree. You can press F1 when a wizard is displayed and get wizard details from the online Help system.

7.3.5 Features of the LAD editor

The STEP 7-Micro/WIN SMART V3 Ladder Logic (LAD) editor allows you to build programs that resemble an electrical wiring diagram. Ladder programming is the method of choice for many PLC programmers and maintenance personnel; it is an excellent language for beginning programmers. The ladder programs allow the CPU to emulate the flow of electric current from a power source through a series of logical input conditions that in turn enable logical output conditions. The logic is separated into networks. The program is executed as dictated by the program, one network at a time, from left to right and then top to bottom. Once the CPU has reached the end of the program, it starts over again at the top of the program.



The various instructions are represented by graphic symbols and include three basic forms.

	Contacts	Represent logic input conditions analogous to switches, buttons, internal conditions and so on.
	Coils	Usually represent logic output results analogous to lamps, motor starters, interposing relays, internal output conditions and other outputs.
	Boxes	Represent additional instructions such as timers, counters, or math instructions.

The networks that you can build in ladder logic range from the simple to the very complex. You can create networks with midline outputs; you can even connect multiple box instructions in series. Box instructions that can be connected in series are labeled with an Enable Output (ENO) line. If a box has power flow at the EN input and executes without error, the ENO output passes power flow to the next element. ENO can be used as an enable bit that indicates the successful completion of an instruction. The ENO bit is used with the top of stack to affect power flow for execution of subsequent instructions.

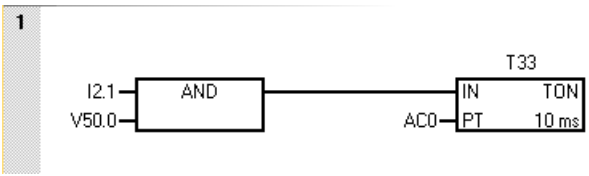
Points to consider

The main points to consider when you select the LAD editor are:

- Ladder logic is easy for beginning programmers to use.
- Graphical representation is often easy to understand, and is popular around the world.
- You can always use the FBD editor (Page 152) or STL editor (Page 152) to display a program created with the LAD editor.

7.3.6 Features of the FBD editor

The STEP 7-Micro/WIN SMART Function Block Diagram (FBD) editor allows you to view the instructions as logic boxes that resemble common logic gate diagrams. There are no contacts and coils as found in the LAD editor, but there are equivalent instructions that appear as box instructions. The program logic is derived from the connections between these box instructions. The output from one instruction (such as an AND box) can be used to enable another instruction (such as a timer) to create the necessary control logic. This connection concept allows you to easily solve a wide variety of logic problems, just as you can with the other editors.



If a box has power flow at the EN input and executes without error, the ENO output passes power flow to the next element. ENO can be used as an enable bit that indicates the successful completion of an instruction. The ENO bit is used with the top of stack to affect power flow for execution of subsequent instructions.

Points to consider

The main points to consider when you select the FBD editor are:

- The graphical logic gate style of representation is good for following program flow.
- You can use the LAD editor (Page 151) or STL editor (Page 152) to display a program created with the FBD editor.
- The expandable AND/OR boxes make it easier to draw complex input combinations.

7.3.7 Features of the STL editor

The STEP 7-Micro/WIN SMART Statement List (STL) editor allows you to create control programs by using STL instruction mnemonics and parameters as text. In general, the STL editor is more suitable for experienced programmers who are familiar with PLCs and logic programming. The STL editor also allows you to create programs that you could not otherwise create with the Ladder Logic or Function Block Diagram editors. This is because you are programming in the native language of the CPU, rather than a graphical editor where some restrictions must be applied in order to compile the diagrams correctly.

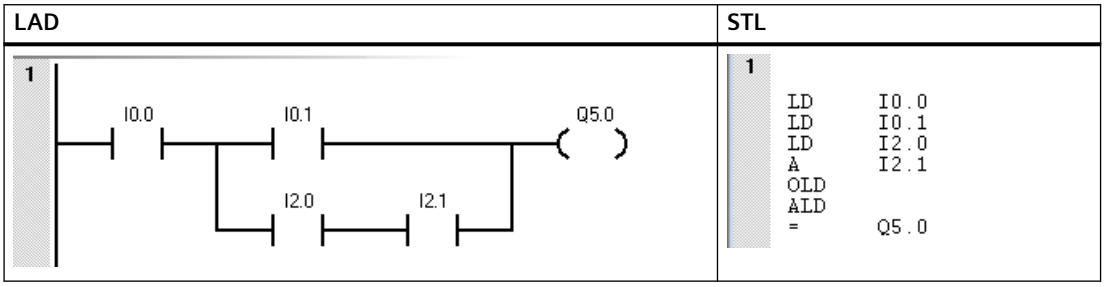
Example of a statement list program

```
1
    LD      I0.0
    LD      I0.1
    LD      I2.0
    A       I2.1
    OLD
    ALD
    =       Q5.0
```

Text-based statement list is similar to assembly language programming. The CPU executes each statement in the list, from top to bottom. After all networks are processed, execution restarts at the top of the list. S7-200 SMART CPUs use a logic stack to resolve AND/OR control logic. The LAD and FBD editors automatically transform AND/OR diagram logic into STL instructions that perform the equivalent logic using the logic stack. In STL programming, you must use the stack instructions and handle the stack operations on your own.

All LAD and FBD programs must successfully transform into STL programs, because the STL version of a LAD or FBD program is what is actually downloaded to the PLC.

The following figure shows a simple program in LAD and the corresponding program in STL.



The table below shows what is happening in the 32-bit logic stack:

Table 7-2 Instructions

Stack	LD I0.0	LD I0.1	LD I2.0	A	OLD	ALD
S0	I0.0	I0.1	I2.0	I2.0 AND I2.1	(I2.0 AND I2.1) OR I0.1	I0.0 AND [(I2.0 AND I2.1) OR I0.1]
S1		I0.0	I0.1	I0.1	I0.0	
S2			I0.0	I0.0		
…						
S31						

Points to consider

The main points to consider when you select the STL editor are:

- STL is most appropriate for experienced programmers.
- STL sometimes allows you to solve problems that you cannot solve easily with the LAD or FBD editor.
- A program created with the STL editor could result in a smaller program size compared to the program size of the equivalent control logic created in the LAD or FBD editor.
- While you can always use the STL editor to view or edit a program that was created with the LAD editor (Page 151) or FBD editor (Page 152), the reverse is not always true. You cannot always use the LAD editor or FBD editor to display a program that was written with the STL editor.

7.4 Data block (DB) editor

The data block allows you to assign constants (Page 120) (either numeric values or character strings) to specific locations of V memory. You can make value assignments to byte (V or VB), word (VW), or double word (VD) addresses. You can also enter optional comments, preceded by // double forward slashes.

- The first line of the data block must have an explicit address assignment. You can use a memory address (absolute address) or a variable name from the global variable table (Page 156) that you have previously assigned to an address.
- Subsequent lines can have explicit or implicit address assignments. An implicit address assignment is made by the editor when you type multiple data values after a single address assignment, or type a line that contains only data values. The editor assigns an appropriate amount of V memory based on your previous address allocations and the size (byte, word, or double word) of the data value(s).
- The data block editor is a free-form text editor; however, it expects an address or variable name in the first position. If you are continuing an implicit data value entry, enter at least one space in the address position before entering the implicit data value assignment. After you finish typing a line and press the ENTER key, the data block editor formats the line (aligns columns of addresses, data, and comments; capitalizes V memory addresses) and redisplays it. The data block editor accepts uppercase or lowercase letters and allows commas, tabs, or spaces to serve as separators between addresses and data values.
- Pressing CTRL–ENTER after completing an assignment line auto-increments the address to the next available address.

Note

If variables with data types are used in the DB, it must be ensured that the values in the DB are consistent with the data types. The DB cannot use complex structure variables.

Example: Data block page

```
Data Block
VB1     249, 250, 251                 // Multiple data values on a single line
                                      // Implicit address assignments
                                      // VB2 holds data value of 250.
                                      // VB3 holds data value of 251.

VB4     252                           // Cannot use previously-assigned addresses (VB0 - VB3)

        253, 254, 255                 // Line with no explicit address assignment results in
                                      // implicit assignment of data values to VB5, VB6, and VB7

VW8     256, 257                      // New data type: Word
                                      // Implicit assignment of 257 to VW10

// Data value assignments must be valid for the data type addessed (Byte, Word, or Double Word)

        65536                         // Error indicates that data value 65536 is too large
                                      // Implicit VW12 address is a Word data type

// You can omit the B, W, or D size designation from an address.
// The data block editor assigns the value to VB, VW, or VD memory according to size of value.

V12     258                           // Word value explicitly assigned to VW12
        65537                         // Double word value implicitly assigned to VD14
```

Note: Enter a space before the data values on the lines where you enter no explicit address.

Example: Direct address and number values

```
Data Block
VB0     255                           // Byte value at VB0
VW2     256                           // Word value at VW2
VD4     700.59                        // Double word real number at VD4
VB8     -35                           // Byte value at VB8
VW10    16#0A47                       // Hex number for Word value at VW10
VD14    146879                        // Double word value at VD14
VW20    2, 4, 8, 16, 32, 64           // Table of Word values starting at VW20
        -2, -4, -8, -16, -32, -64     // Values that extend to multiple lines cannot be in column 1
                                      // The first column is reserved for a memory address or symbol

VB45    "Up"                          // Two-byte string starting at VB45
VW90    65535                         // Word value at VW90
V100    255, 'abc', 65536, 1.0        // Values for undefined memory types starting at VB100.
                                      // Mixing different sizes is allowed.
VB110   255                           // Same as entering B110 255
VW120   255                           // Same as entering W120 255
VD130   700.59                        // Same as entering D130 700.59
V140    255, 'abc', 65535, 1.0        // Same as entering 140: Must start in column 1
VW150   2#1010101010101010            // Binary Word value
VD152   2#10101010101010101010101010101010 // Binary Double word value
```

Example: Global variable address and global variable assignment

Complex structure variables cannot be used in DB.

Global Variable Table

	Variable Name	Data Type	Initial Value	Retain	Bind	Address	Length	Comment	Snaps...
1	power_enable	BOOL	OFF			DB2.D...	1 b		
2	home_exe	BOOL	OFF			DB2.D...	1 b		
3	rel_exe	BOOL	OFF				1 b		
4	abs_exe	BOOL	OFF			DB2.D...	1 b		
5	⊞ TriggerInput	MC_TRIGGER_REF				DB2.D...	3 B		
6	probe_exe	BOOL	OFF			DB2.D...	1 b		
7	reset_exe	BOOL	OFF				1 b		
8	jog_fwd	BOOL	OFF				1 b		
9	jog_bwd	BOOL	OFF				1 b		
10	stop_exe	BOOL	OFF				1 b		
11									

Variable Table 1 | Constant Table 1 | I/O Variable | POU Variables | System Defined Variable Table | FB Instance Table | Syst

Data Block

```
Symbolic_Address Symbolic_Constant // assigns VB100 a value of 200
```

Page_1

Example: Alternate binary entry methods and resultant binary assignment

You can enter values of 1 or 0 for binary assignments, or "true", "false", "on" or "off" (in either lower, upper, or mixed case). The data block editor interprets your entry and shows the resultant binary assignment.

Data Block

```
v0.2 1 0 true false on off
```

Data Block

```
VB0     2#1010100
```

7.5 Global variable table

A global variable is a variable name you assign to a memory address or a constant. You can create global variable names for the following memory types: I, Q, M, SM, AI, AQ, V, S, C, T, HC. variables defined in the global variable table are global in scope. You can use your defined global variables in all of the Program Organizational Units (Page 141) (POUs) of your program. If you make a variable name assignment in a local variable table (Page 159), the variable is local in scope. It only applies to the POU where you defined it. You can define variables either before or after you create your program logic.

Opening a global variable table

To open a global variable table from STEP 7-Micro/WIN SMART V3, use one of the following methods:

- Select "Global Variable Table" from the component drop down list in the Windows area of the View menu.
- Open the Global Variable Table Folder in the project tree. Click the "Insert Global Variable Table" button to add a new table.

Assigning global variables in the globle variable table

To assign a global variable to an address or constant value, follow the procedure below:

1. Open the variable table.
2. Type the variable name (for example, Input1) in the Name column. The maximum number of characters in a variable name is 128 single-byte characters.

   Note

   If you have selected to view both variable and absolute view of operands for your project, lengthy variable names are truncated with a tilde (~) in the program editor. You can place your mouse pointer over the truncated name to see the entire name displayed in a tooltip.

3. Type the initial value (for example, 123) in the Initial Value column. Note that to assign a string constant to a global variable, you enclose the string constant in double quotation marks.

	Name	Data Type	Initial Value	Retain	Bind	Address	Comment
1	Input	INT		☐	☑	VW100	
2	Output	INT		☐	☑	VW200	
3	EarlyValue	REAL	123.0	☐	☐		
4	LastValue	REAL	234.0	☐	☐		
5				☐	☐		

4. You can check the "Bind" option and enter the specific address in the PLC in the address bar. **Note**: If you don't check the "Bind" option, the system will assign the variable an address in DB2 during the compilation process.
5. You can check the "Retain" option to retain the variable.

   Note

   Retention for unbound and bound variables

   - If the variable is not bound to a specific address and the "Retain" option is selected, the system assigns a DB2 address to the variable and ensures data retention after a power failure.
   - If a variable is bound to a specific address, the effect of retention depends on the retentive configuration in the system block. If only some addresses in the system block are set for retention, data in the non-retained addresses will be lost after a power failure, compromising the retention function's effectiveness.

6. Optionally, type in a comment up to a maximum of 79 characters.

You can resize the width of the columns in the variable table editor as needed.

Note

You can create several variable tables; however, you cannot use the same variable name more than once as a global variable assignment.

Syntax rules and indication of errors

STEP 7-Micro/WIN SMART V3 indicates erroneous or incomplete variable assignments with color and wavy underlining:

Variable Name	Data Type	Initial Value	Retain	Bind	Address
2name	BOOL		☐	☑	I0.0
			☐	☑	VBB0
Begin	BOOL		☐	☑	I0.2

Red text indicates invalid syntax.
A variable cannot begin with a numeral.
"VBB0" is an invalid address.
Begin is a reserved word and invalid as a variable name.

Variable Name	Data Type	Initial Value	Retain	Bind	Address
Pump1	BOOL		☐	☑	I0.0
Pump1_1	BOOL		☐	☑	I0.0
Pump1_2	BOOL		☐	☑	I1.0

A green wavy underline indicates invalid use.

A "_number" suffix is automatically added to duplicate variable names.

"Pump1" is a duplicate name.

I0.0 is a duplicate address.

Observe the following syntax rules when defining variables:

- Variable names can contain alphanumeric characters, underscores, and extended characters from ASCII 128 to ASCII 255. The first character cannot be a numeral.
- Do not use keywords as variable names.
- The maximum length for a variable name is 128 characters.

Note

When you correct an erroneous variable name or address, press the TAB key, ENTER key, or an ARROW key to complete the edited correction.

Indirect addressing

When referencing a global variable in the program editor, you can use indirect notation (& and *) with global variable names as with direct addresses. For more information about indirect addressing, see the topic on direct and indirect addressing.

Inserting additional rows

Use one of the following methods to insert additional rows in the global variable/constant table:

- Right-click a cell of the global variable/constant table and select "Insert>Row" from the context menu. STEP 7-Micro/WIN SMART V3 inserts a new row above the current location.
- From the Insert area of the Edit menu ribbon strip, select "Row". STEP 7-Micro/WIN SMART V3 inserts a new row above the current location of the cursor in the global variable/constant table.

Deleting variables/constants

Use one of the following methods to delete additional rows in the global variable/constant table:

- Right-click a cell of the global variable/constant table and select "Delete>Row" from the context menu. STEP 7-Micro/WIN SMART V3 deletes a the row in the current location.
- From the Delete area of the Edit menu ribbon strip, select "Row". STEP 7-Micro/WIN SMART V3 deletes the row in the current location of the cursor in the global variable/constant table.

Sorting a variable/constant table

You can sort a variable table by the name or by the Address column in either ascending or descending alphabetical order. In the Address column, numeric constants sort above string constants, which sort above addresses.

To sort a column, click either the Name or Address column header to sort by that value. To reverse the order of the sort, click the column again. STEP 7-Micro/WIN SMART V3 displays an up or down arrow next to the column that is sorted to indicate the sort selection.

Note

You can print variable tables from the Print area of the File menu ribbon strip.

You can view variables on a network-by-network basis by displaying the variable information table.

7.6 Local variable table

A local variable table allows you to define local variables that are local to a specific POU. The following situations define when to use a local variable:

- You want to create portable subroutines/FBs that do not make references to absolute addresses or global variables.
- You want to use interim variables (local variables declared as TEMP) to perform calculations in order to free up PLC memory.
- You want to define inputs and outputs for your subroutines/FBs.

If these descriptions do not fit your situation, you do not need to use local variables; you can make all of your variables global by defining them in the globale variable table (Page 156).

Understanding local variables

You can use the local variable table of the program editor to assign local variables that are unique to an individual subroutine, function block or interrupt routine.

Local variables can be used as parameters that are passed in to a subroutine/FB and can be used to increase the portability or reuse of a subroutine/FB.

Each POU (Page 141) in your program has its own local variable table, with 1k bytes of L memory (1008 bytes if you are programming in LAD or FBD). These local variable tables allow you to define local variables that are restricted in scope: a local variable is only valid inside the POU where it was created. By contrast, global variables, which are valid in every POU, can only be defined in the global variable table.

You assign a declaration type (TEMP, IN, IN_OUT, OUT or STATIC) and a data type, but not a memory address, when you make assignments in a Local Variable Table; the Program Editor automatically assigns memory locations in L memory for all local variables.

- For subroutine, a local variable table address assignment associates a local variable name with an L memory address. For FB, ony TEMP associates with and L memory address.
- The Local Variable Table of main, sunroutine and interrupt routine does not support assign a default value directly to a local variable.
- Except TEMP, The Local Variable Table of function block supports assign a value directly to a local variable .

Note

For subroutine, local data values are not initialized to zero by the PLC. You must initialize the local variables that you use, in your program logic.

Declaration types for local variables

The type of local variable assignment you can make depends on the POU where you are making the assignment. The main program , interrupt routines, function blocks and subroutines can use temporary (TEMP) variables. Temporary variables are only available while the block is being executed and are then free to be overwritten when the block is completed.

Data values can be passed as parameters in and out of a subroutine/FB as follows:

- If you want to pass a data value into a subroutine/FB, then create a local variable in the local variable table of subroution/FB and specify its declaration type as IN.
- If you want to pass a data value established in the subroutine/FB back to the calling routine, then create a local variable in the local variable table of subroution/FB and specify its declaration type as OUT.
- If you want to pass an initial data value into a subroutine/FB, perform an operation that may modify the data value, and pass the modified result back to the calling routine, then create a local variable in the local variable table of subroution/FB and specify its declaration type as IN_OUT.

Declaration type	Description
IN	Input parameter provided by the calling POU.
OUT	Output parameter returned to the calling POU.
IN_OUT	Parameter whose value is supplied by the calling POU, modified by the subroutine, and then returned to the calling POU.
TEMP	Temporary variable that is saved temporarily in the local data stack. Once the POU has been executed completely, the value of the temporary variable is no longer available. Temporary variables do not keep their value between POU executions.
STATIC	FBs only, STATIC is used to store static intermediate results in the FB instance. STATIC can only be used inside FB to store static data (can hold the value of the last cycle).

Data type checking for local variables

When you pass local variables as parameters for a subroutine/FB, the data type that you have assigned in the Local Variable Table of that subroutine/FB must match the data type of the value in the calling POU.

Example

You call SBR0 from main program, using a global variable called INPUT1 as an input parameter of the subroutine.

Inside the Local Variable Table of SBR0, you have defined a local variable called FIRST as an input parameter.

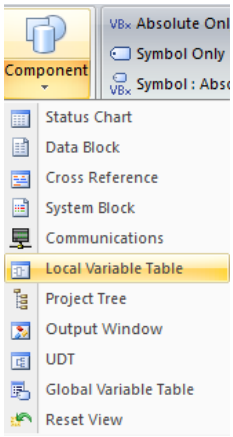
When main program calls SBR0, the value of INPUT1 is passed to FIRST.

The data types of INPUT1 and FIRST must match.

If INPUT1 is a REAL and FIRST is a REAL, the data types match. If INPUT1 is a REAL but FIRST is an INT, the data types do not match and the program cannot be compiled until this error is corrected.

Viewing the local variable table

To view the local variable table for the POU selected in the program editor, select "local variable table" from the Component drop-down list in the Windows area of the View menu.



Note

You can put the local variable table on the quick access toolbar for easy access.

Making assignments in the local variable table

Note

Make the assignments in the local variable table **before** using local variables in your program. When you use variable names in your program, the program editor checks first the Local Variable Table of the appropriate POU, and then the global variable table. If the variable name is undefined in both places, the program editor treats it as an undefined global variable which is indicated by a green wavy underline. The program editor does not automatically re-read the local variable table and make corrections to your program logic. If you later make a data type assignment that defines that local variable name (in the local variable table), you must manually insert a pound symbol (#) in front of the name, like this: #UndefinedLocalVar (in the program logic). For this reason, declaring the local variables prior to usage minimizes the programming effort.

The maximum limit of input/output parameters for each subroutine call is 64. If you attempt to download a program that exceeds this limit, STEP 7-Micro/WIN SMART V3 returns an error.

To make an assignment in a local variable table, follow the procedure below.

1. Ensure that the correct POU is displayed in the Program Editor window by clicking, if necessary, on the tab of the desired POU. (Since every POU has its own local variable table, you need to make sure that you are making assignments to the correct POU.)
2. Display the local variable table if it is not already visible by selecting "local variable Table" from the Component drop-down list in the Windows area of the view menu.Do not preface the name with a pound symbol in the local variable table. Pound symbols are only used to precede local variables in the program code.

3. Choose a row that has the right variable type for the kind of local variable that you want to define, and type a name for the local variable in the Name field.
   - If you are making an assignment in main program or an interrupt routine, the local variable table contains only TEMP variables.
   - If you are making an assignment in a subroutine, the local variable table contains IN, IN_OUT, OUT, and TEMP variables.
   - If you are making an assignment in a function block, the local variable table contains IN, IN_OUT, OUT, STATIC and TEMP variables.

   Note

   Local variable names are permitted to contain a maximum of 23 alphanumeric characters and underscores. They are also permitted to contain extended characters (ASCII 128 to ASCII 255). The first character is restricted to alpha and extended characters only. It is illegal to use keywords as variable names, or to use names that begin with a number or contain characters that are not alphanumeric or in the extended character set.

   Local variable names are downloaded and stored in CPU memory. The use of longer variable names may reduce the memory available to store your program.

4. Click the mouse pointer in the Data Type field and use the list box to select an appropriate data type for the local variable.

   Note

   When you assign local variables as parameters for subroutines/FBs, you must ensure that the data type that you assign to the local variable does not conflict with the operand being used in the subroutine/FB call.

5. For function blocks, you can assign the default value and select retentive status.
6. Optionally provide a comment describing your local variable.
7. If you use the FB local variable table, you can set default value and retain the variable.

Entering additional local variables

The local variable table displays a fixed number of rows for local variables. To add more rows to the table, select a row in the table of the variable type that you want to add and click the Insert button in the local variable table window. A new row is automatically generated above the row you selected, and is for the same variable type that you selected.

You can also add rows by right-clicking an existing row and selecting **Insert > Row** or **Insert > Row Below** from the context menu.

Deleting local variables

To delete a local variable, select it in the local variable table and click the Delete button. Alternatively you can delete a row by right-clicking it and selecting **Delete > Row** from the context menu.

Collapse/expand the local variable table

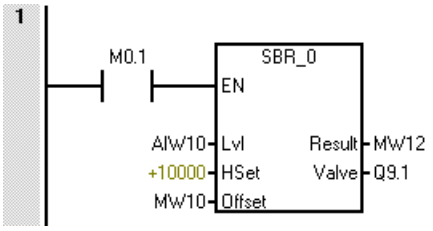
The collapse/expand button allows you to collapse or expand the Array/UDT type variable in the local variable table. By clicking, you can collapse or expand the Array/UDT variable to conveniently view its contents.

Local variable table example

The following example shows a typical local variable table for SBR_0, and then a call to SBR_0 from another program block.

Local Variable Table

	Name	Address	Var Type	Data Type	Comment
1	EN		IN	BOOL	
2	Lv1	L0.0	IN	BOOL	Tank trans...
3	HSet	L0.1	IN	BOOL	High setpoint
4	Offset	L0.2	IN	BOOL	Offset
5			IN		
6			IN_OUT		
7	Result	L0.3	OUT	BOOL	Result value
8	Valve	L0.4	OUT	BOOL	Control value
9			OUT		
10			TEMP		



7.7 UDT table

7.7.1 Overview

Description

A user-defined data type (UDT) is used for the declaration of a tag. It represents a data structure made up of several components of different data types. The nesting depth is hereby limited to 4 levels.

You can change a user-defined data type (UDT) centrally and use it repeatedly in the program. All locations of use are updated automatically. UDT can be used in the global variable table, variable table or FB instance table.

Advantages of the user-defined data types:

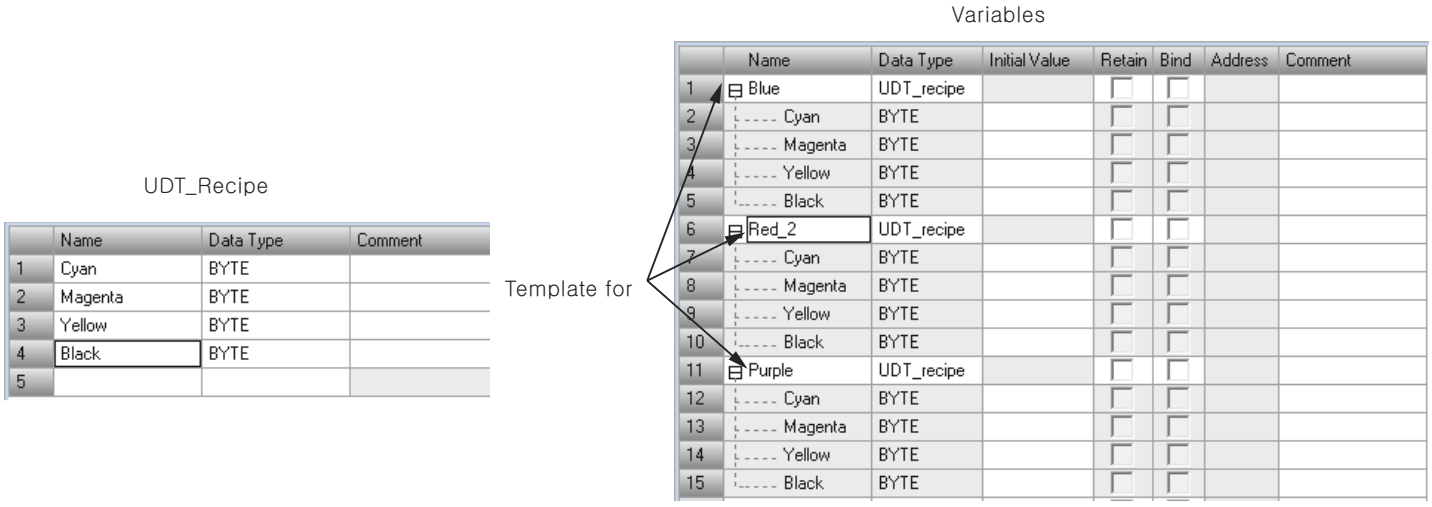
- Group data according to the process control
- Transferring a parameter as a data unit

Using a user-defined data type

You can declare user-defined data types as a type when creating variables. Based on this type, you can create a number of variables, all of which have the same data structure. These data blocks can be adapted by entering different actual values for the corresponding task.

For instance, create a user-defined data type for a recipe for blending paints. You can then assign this data type to several variables, each of which contains different quantity information.

The following figure shows this application:



user-defined data types can be used as templates for the creation of global data blocks with identical data structures.

Nesting depth and potential number of user-defined data types

user-defined data types can be nested to a depth of 4 hierarchy levels. This nesting depth is independent of the CPU used.

System defined UDT

To meet your specific requirements and enhance development efficiency, the system has predefined a series of UDTs. These types extend the system's basic data types, enabling you to represent and manipulate complex data structures more flexibly and accurately. By utilizing these UDTs, you can simplify the code writing process, improve the readability and maintainability of the code, and ensure data consistency and integrity.

7.7.2 Using UDT table

STEP 7-Micro/WIN SMART V3 predefined several of UDT tables for your convenience. You can also create your own UDT table and define UDTs in it. The predefined UDT table names start with `SYS_` and they cannot be edited.

Opening a UDT table

To open a UDT table, use one of the following methods:

- Select "UDT Table" from the component drop down list in the Windows area of the View menu.
- Open the UDT Table Folder in the project tree; select a table name; then press Enter or double-click the table name.

Add UDT table

To add a UDT table, use one of the following methods:

1. Select "UDT Table" in the project tree
2. Select the "Insert Table" from the shortcut menu.
3. Enter the table name in the popup window and click "Yes".

or

1. Open the UDT table.
2. Click the Add button, enter the table name in the popup window, and click "Yes".

Add UDT member

1. Open the UDT table.
2. Add the UDT member name (for example, Input1) and data type.
3. Optionally, type in a comment up to a maximum of 79 characters.

Inserting additional rows

Use one of the following methods to insert additional rows in the UDT table:

- Right-click a cell of the UDT table and select Insert>Row from the context menu. STEP 7-Micro/WIN SMART inserts a new row above the current location.

Deleting a UDT member

To delete a UDT member, you can delete a row by right-clicking it and selecting **Delete > Row** from the context menu.

Delete UDT table

To delete a UDT table, use one of the following methods:

- Select an existing UDT tab in the UDT view and use one of these methods to delete it:
  - From the Delete area of the Edit menu ribbon strip, click the down arrow beneath "Object" and select "UDT" from the drop-down menu.
  - Click the Delete button on the UDT toolbar.
- Click a UDT under "UDT" in the project tree and press the "Delete" button.
- Right-click a UDT under "UDT" in the project tree and select "Delete" from the context menu.

Example

Input I-points from IO communication into the variable, and output variable to Q-points

1. Create UDTs for I-points and Q-points. The arrangement and types of element variables should be set according to the communication object.

	Name	Data Type	Comment
1	Temprature	REAL	
2	Humidity	REAL	
3	Concentration	REAL	
4	Volume	REAL	
5	Times	INT	
6			

Figure 7-1 I_Point

	Name	Data Type	Comment
1	Valve_1	REAL	
2	Valve_2	REAL	
3	Valve_3	REAL	
4	Valve_4	REAL	
5	Start	BOOL	
6	Pause	BOOL	
7			

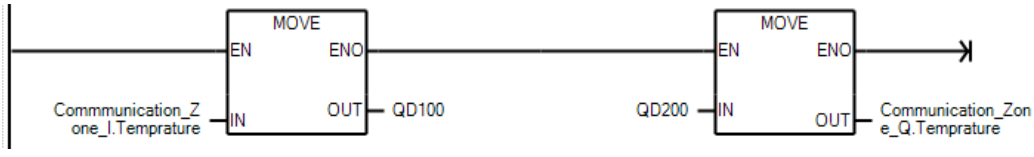
Figure 7-2 Q_Point

2. Use UDT in variable table.

	Name	Data Type	Initial Value	Retain	Bind	Address	Comment
17	Communication_Zone_I	Stru_I		☐	☑	IB100	
18	Temprature	REAL		☐	☑	ID100	
19	Humidity	REAL		☐	☑	ID104	
20	Concentration	REAL		☐	☑	ID108	
21	Volume	REAL		☐	☑	ID112	
22	Times	INT		☐	☑	IW116	
23	Communication_Zone_Q	Stru_Q		☐	☑	QB100	
24	Valve_1	REAL		☐	☑	QD100	
25	Valve_2	REAL		☐	☑	QD104	
26	Valve_3	REAL		☐	☑	QD108	
27	Valve_4	REAL		☐	☑	QD112	
28	Start	BOOL		☐	☑	Q116.0	
29	Pause	BOOL		☐	☑	Q116.1	
30	Input_1	DWORD		☐	☐		
31				☐	☐		

Variable Table 1 | Constant Table 1 | I/O Symbols | POU Variables | System Defined Variable Table

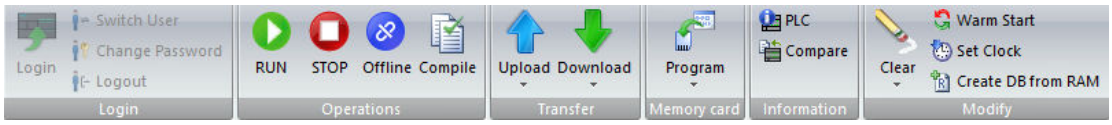
3. Write a MOVE program.



7.8 PLC error reaction

7.8.1 Error handling in the S7-200 SMART CPU

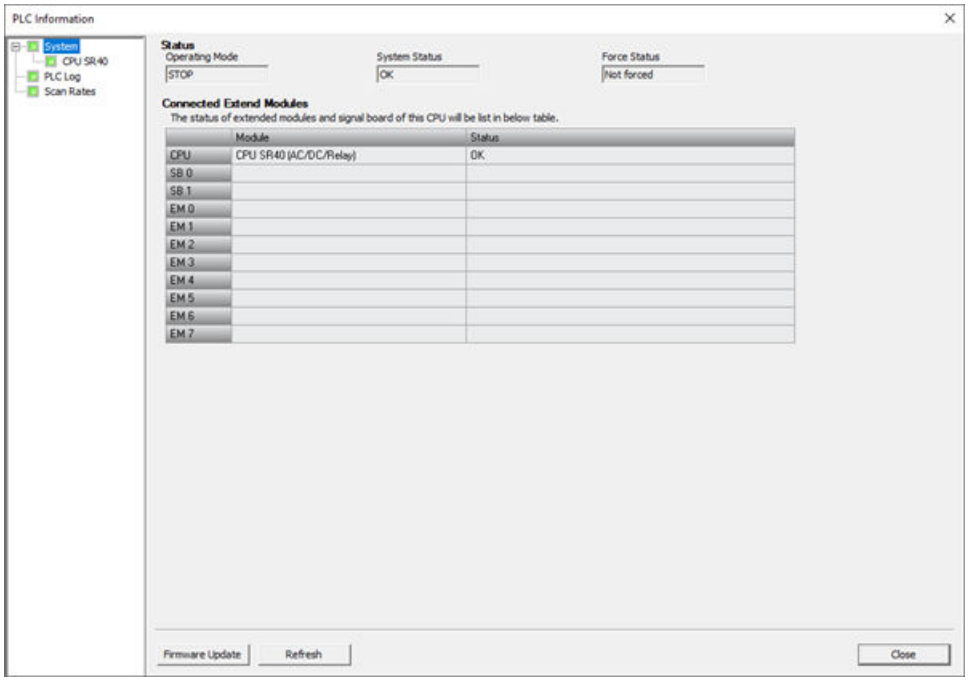
To view PLC information or perform a firmware update, click the PLC button in the Information area of the PLC menu ribbon strip.



The PLC information dialog displays the following information:

- System information
  - Connected CPU status
  - Configured PROFINET device status
- PLC Log
- PROFINET Alarm
- Scan Rates

STEP 7-Micro/WIN SMART V3 displays the PLC Information dialog:



Note

The "Refresh" button is used for updating the information. No matter where you click the "Refresh" button, all the PLC information is updated.

The "Firmware Update" button is used for updating the firmware.

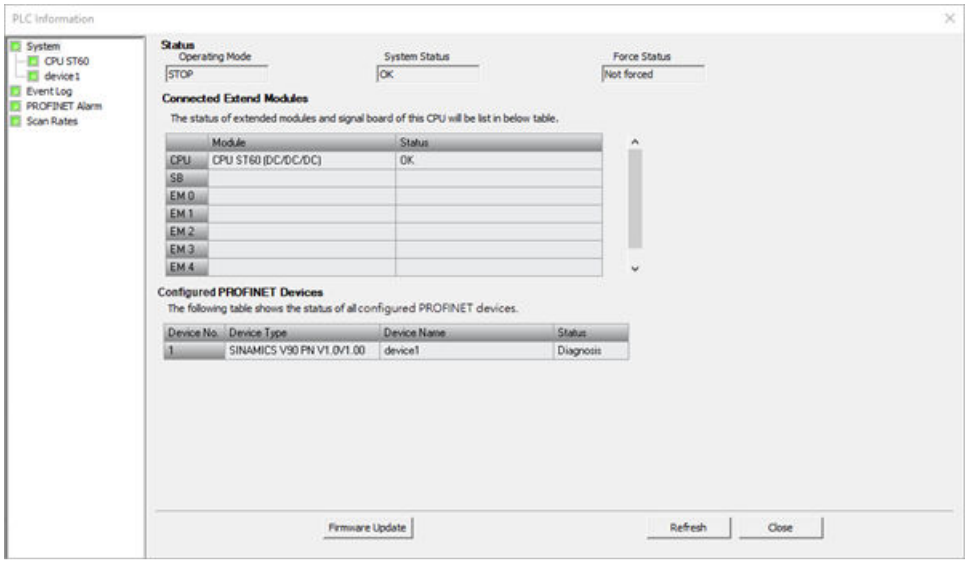
Note the following information:

- The PLC provides SM bits for programmed reactions to errors. Refer to the listing of the SM bits (Page 1067).
- The GET_ERROR (Get non-fatal error code) program instruction returns the PLC's current nonfatal error code and clears the non-fatal error information latched in the PLC. Refer to the GET_ERROR instruction (Page 426) for details.

7.8.2 System information

7.8.2.1 System

To view information for the system, click the "System" button. The PLC Information dialog shows the system status, the status of connected extend modules, and the status of configured PROFINET devices.



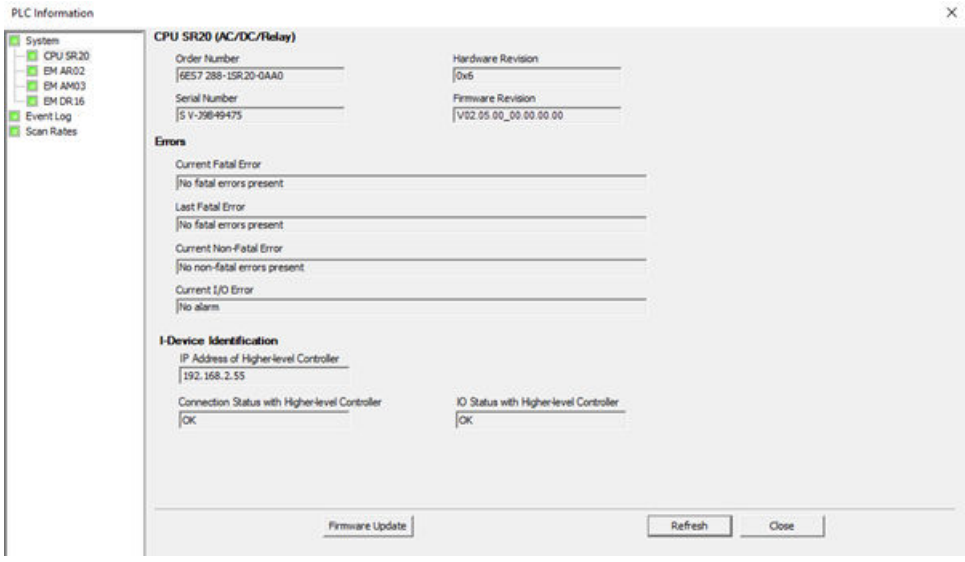
The "System" dialog displays the following information:

- Status: the status of system
  - Operating Mode: PLC operating modes (RUN or STOP)
  - System Status: the status of system (OK or Error)
  - Force Status: whether the variable is forced or not
- Connected Extend Modules: the status of extend modules and CPU signal board
- Configured PROFINET Devices: the status of PROFINET devices
  The status is as follows:
  - Not connected: The controller cannot connect with the device.
  - OK
  - Diagnosis: An alarm is reported.

7.8.2.2 CPU

To view information for a specific device, click the appropriate device under "System".

The PLC Information dialog shows you the device identification information, device status and module status as follows:



- Connected CPU: the CPU model
  The dialog lists the following CPU information:
  - Order Number
  - Hardware Revision
  - Serial Number
  - Firmware Revision
- Errors: the error information. To identify specific errors, refer to error codes (Page 1065).
  - Current Fatal Error: the latest fatal error
  - Last Fatal Error: The "Last Fatal Error" field shows the previous fatal error code generated by the CPU. This value is retained after a power cycle. This location is cleared whenever all memory of the CPU is cleared.
  - Current Non-Fatal Error: the latest non-fatal error
  - Current I/O Error: the latest I/O error
- I-Device Identification: the I-Device information.

  Note

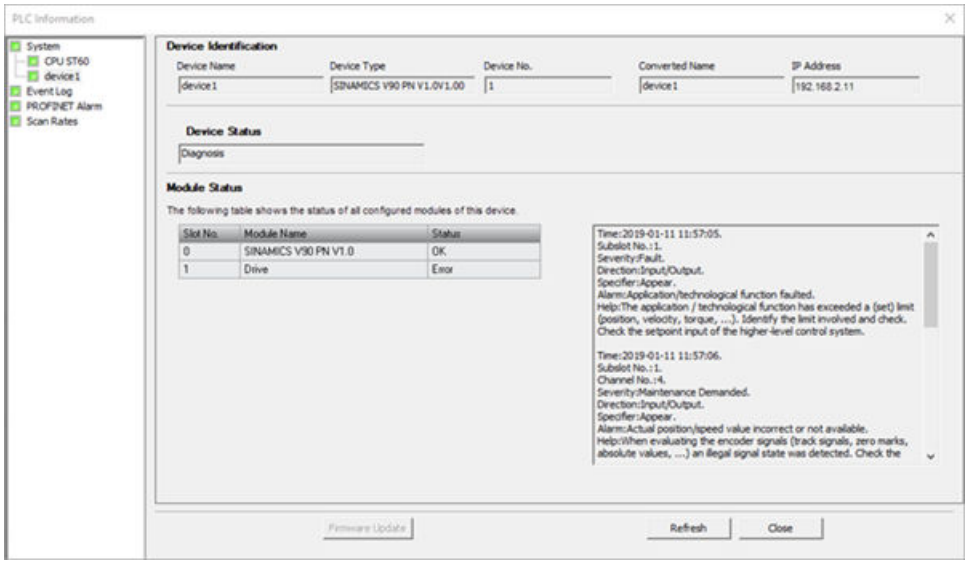
  If the CPU works as an I-Device, the CPU dialog displays the I-Device Identification information.

  - IP Address of Higher-level Controller: If I-Device is not connected with its higher-level controller, "--.--.--.--" is displayed here.
  - Connection Status with Higher-level Controller:
    Not connected: The I-Device is not connected with the higher-level controller.
    Diagnosis: The I-Device is connected with the higher-level controller, but the I-Device configuration doesn't match with the higher-level controller configuration.
    OK
  - IO Status with Higher-level Controller:
    Not connected: The I-Device is not connected with the higher-level controller.
    IO data error
    OK

7.8.2.3 PROFINET device

To view information for a specific device, click the appropriate device under "System".

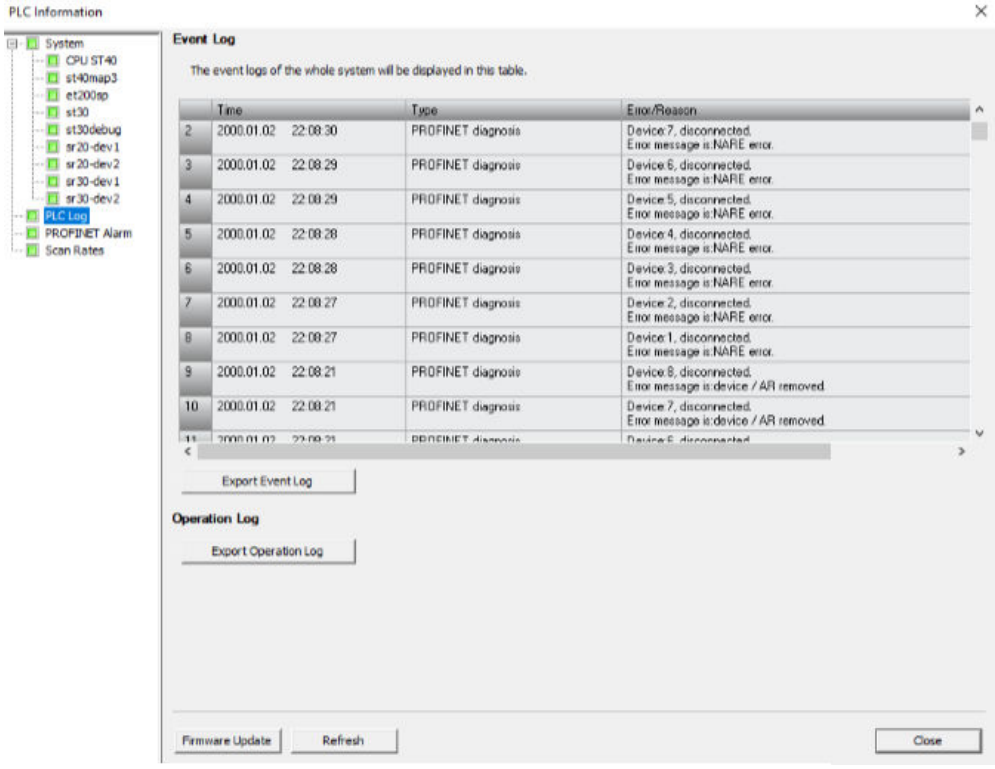
The PLC Information dialog shows you the device identification information, device status and module status as follows:



The "Module Status" field shows the module status. If the error occurs in any module, the detailed error information that includes slot and channel number, error description and solution description displays at the right side.

7.8.3 PLC log

To view the event log and operation log for the PLC, click "PLC Log". The PLC Information dialog displays a table of time-stamped events. The table lists the time, event type and error or reason information.

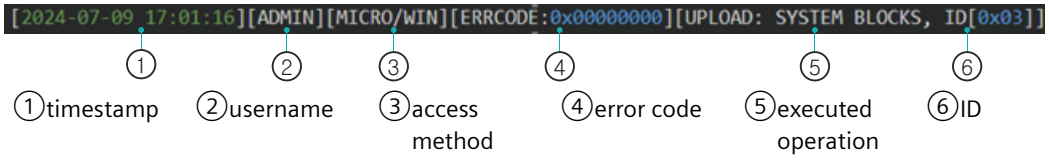


The maximum number of displayed event log is 128.

Export log

To export the log, click the Export Event Log or Export Operation Log button and browse to the folder where you want to save the log.

The exported data format for operation log is as follows, including timestamp, username, access method, error code, executed operation, and parameters.

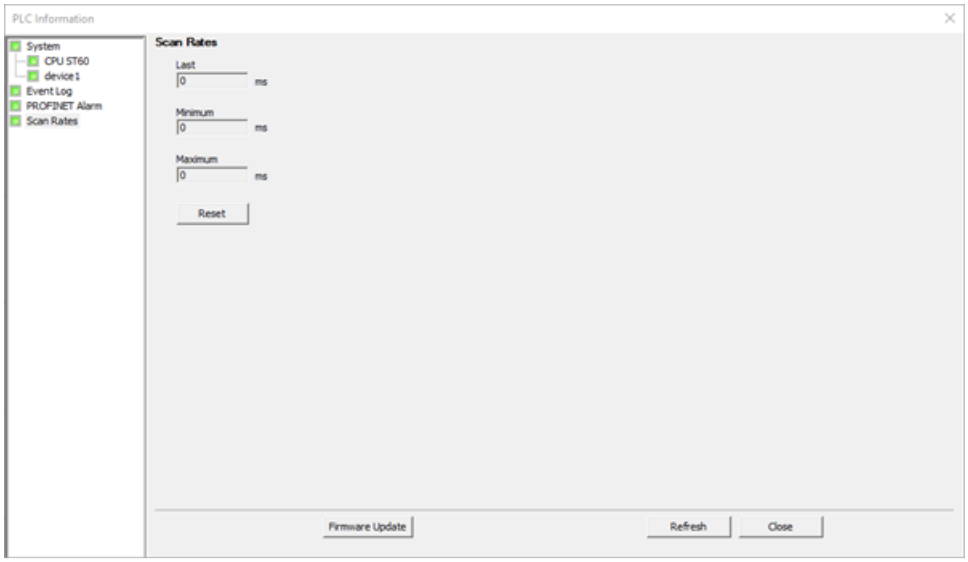


Note

The Operation Log stores the most recent 512K Bytes of operation logs. When the operation log exceeds 512K Bytes, events are recorded in the event log. You should note and save these events if they are needed for auditing or other purposes.

7.8.4 Scan rates

To view scan rate statistics, click "Scan Rates".



The "Scan Rates" dialog displays the following information:

- Last: the last scan rate
- Minimum: the minimum scan rate
- Maximum: the maximum scan rate

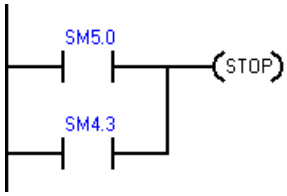
If you click "Reset" button, the scan rate information in PLC is cleared. Then you can click "Refresh" button to display the updated scan rate information.

7.8.5 Non-fatal errors and I/O errors

The CPU does not change to STOP mode when it detects a non-fatal error. It only logs the event in SM memory and continues with the execution of your program. However, you can design your program to force the CPU to STOP mode when a non-fatal error is detected.

The following sample program shows a network of a program that is monitoring two of the global non-fatal error bits and changes the CPU to STOP whenever either of these bits = 1.

Table 7-3 Example logic for detecting a non-fatal error condition

LAD		STL
	When an I/O error or a run-time error occurs, go to STOP mode	**Network 1** **LD SM5.0** **O SM4.3** **STOP**

Non-fatal errors are those indicating problems with the construction of the user program or with certain instruction execution problems in the user program. I/O errors are those indicating problems with the I/O of the CPU, signal board, and expansion modules. You can use STEP 7-Micro/WIN SMART V3 to view the error codes that were generated by the non-fatal and I/O errors.

Click the PLC button from the Information section of the PLC menu ribbon strip, to see the current error status of a PLC connected to STEP 7-Micro/WIN SMART V3.

Table 7-4 Non-fatal error types

	Description
Program-compile errors in the CPU	The CPU compiles the program as it downloads. If the CPU detects that the program violates a compilation rule, it aborts the download and generates an error code. (A program that was already downloaded to the CPU would still exist in the permanent memory and would not be lost.) After you correct your program, you can download it again.
I/O device errors	After power-up and after a system block download, the CPU verifies that the I/O configuration stored in the system block matches the CPU, signal board, and expansion modules that are actually present. Any mismatch results in the generation of a configuration error for the device. During runtime, devices can detect other I/O problems (such as missing user power or input value exceeding limits) that cause the CPU to generate an I/O error. The CPU stores module status information in special memory (SM) bits. Your program can monitor and evaluate these bits. SM5.0 is the global I/O error bit and remains set while any I/O error condition exists.
Program execution errors	Your program can create error conditions during execution. These errors can result from improper use of an instruction or from the processing of invalid data by an instruction. For example, an indirect-address pointer that was valid when the program compiled could point to an out-of-range address if the program modified the pointer during execution. Modifying a pointer to an invalid address is an example of a run-time programming problem. The CPU sets SM4.3 upon the occurrence of a run-time programming problem. SM4.3 remains set while the CPU is in RUN mode. The program can execute the GET_ERROR instruction (Page 426) to get the current non-fatal error code and reset SM4.3 to OFF.

Refer to the non-fatal error code list (Page 1062) for a description of compile rule violations and run-time programming problems.

Refer to the description of the SM bits (Page 1064) for more information about the SM bits used for reporting I/O and program execution errors.

7.8.6 Fatal errors

Fatal errors cause the PLC to stop the execution of your program. Depending upon the severity of the fatal error, it can render the PLC incapable of performing any or all functions. The objective for handling fatal errors is to bring the PLC to a safe state from which the PLC can respond to interrogations about the existing error conditions.

When a fatal error is detected, the PLC changes to STOP mode, turns on the STOP and the ERROR LED, overrides the output table, and turns off the outputs. The PLC remains in this condition until the fatal error condition is corrected.

Once you have made the changes to correct the fatal error condition, use one of the following methods to restart the PLC:

- Turn the PLC power off and then on.
- Using STEP 7-Micro/WIN SMART V3, click the "Warm Start" button in the modify area of the PLC ribbon strip. This forces the PLC to restart and clear any fatal errors.

Restarting the PLC clears the fatal error condition and performs power-up diagnostic testing to verify that the fatal error has been corrected. If another fatal error condition is found, the PLC again sets the ERROR LED, indicating that an error still exists. Otherwise, the PLC begins normal operation.

Some error conditions can render the PLC incapable of communication. In these cases, you cannot view the error code from the PLC. These types of errors indicate hardware failures that require the PLC to be repaired; they cannot be fixed by changes to the program or clearing the memory of the PLC.

Refer to the fatal error code list (Page 1065) for details.

7.9 Program edit in RUN mode

WARNING

Risks when downloading a program in RUN mode

When you download program changes to a PLC in RUN mode, your changes immediately affect process operation. You have no margin for error; mistakes in your programming edits can cause death or serious injury to personnel, and/or damage to equipment. Only qualified personnel should perform a program edit in RUN mode.

Overview

The "program edit in RUN mode" feature allows you to make changes to a program and download them to your PLC without switching to STOP mode.

- You can make minor changes to your current process without having to shut down.
  Example: Change a parameter value.
- You can debug a program more quickly with this feature.
  Example: Invert the logic for a normally open or normally closed switch.

If you download changes to a real process (as opposed to a simulated process, which you might do in the course of debugging a program), be sure to think through the possible safety consequences to machines and machine operators before you download.

You can download only the program block (OB1, subroutines, function block and interrupts) during a program edit in RUN mode. You cannot download the system block or the data block during a program edit in RUN mode.

Prerequisites for editing in RUN mode

You cannot download your program edits to a PLC that is in RUN mode unless you have met these prerequisites:

- Your program must compile successfully.
- You must have successfully established communications between the computer where you are running STEP 7-Micro/WIN SMART V3 and the PLC.
- You must login PLC.
- You must provide a password for a protected POU to open the block (for normal editing, edit in RUN mode, and program status operations).

If you change the PLC to STOP mode while a program edit in RUN mode is in progress, the PLC aborts the editing session.

Possible complications

To help you decide whether to download your program modifications to the PLC in RUN mode or STOP mode, consider the following effects from various types of program modifications made during a RUN mode edit:

- If you delete the control logic for an output, the output maintains its last state until the next power cycle or transition to STOP mode.
- If you delete HSC, Motion, or PLS functions that were running at the time of the edit in RUN mode, then these functions continue to run until the next power cycle or transition to STOP mode.
- If you delete ATCH or DTCH instructions in a RUN mode edit but do not delete the associated interrupt routine, then the interrupt routine continues to execute whenever the controlling event occurs until the next power cycle or transition to STOP mode.
- If you add ATCH instructions that are conditional on the first scan flag, the CPU does not enable these events until the next power cycle or STOP-to-RUN mode transition.
- If you delete an ENI or DISI instruction, activated interrupt routines events still continue to operate until the next power cycle or transition from RUN to STOP mode.
- If you modify the table address of a RCV instruction and the RCV instruction is active at the time of the edit in RUN mode, then the PLC writes the received data to the old table address. The PLC does not use the new address until the current receive request (to the old address) completes. Because you have edited your program, if the program looks for the data in the new address, the data will not be there. GET and PUT instructions function similarly.
- The PLC does not execute logic that is conditional on the first scan flag until after a power cycle or a transition from STOP to RUN mode. The startup of the modified program after a RUN mode edit does not set the first scan flag.

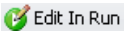
Handling positive or negative transitions

To minimize the process impact of changes that involve the relocation of positive transition (EU) and negative transition (ED) instructions in your program during a RUN mode edit, STEP 7-Micro/WIN SMART V3 temporarily allocates a number to each transition instruction included in your program. For each transition instruction that you add in your program during a RUN mode edit, you must assign it a unique identification number. To assist you in selecting an unused number, STEP 7-Micro/WIN SMART V3 provides an edge usage tab on the cross reference window, available when you activate the Program Edit in RUN Mode feature. This table lists all EU/ ED instructions that are currently in use in your program, so you can use this list to guide you in making changes to your program.

Performing a program edit and download in RUN mode

To initiate a program edit in RUN mode, follow these steps:

1. From the Settings area of the Debug menu ribbon strip, click the Edit In Run button.

   Note

   If you have not saved your current program in the program editor, STEP 7-Micro/WIN SMART V3 prompts you to save your project. You can use the same name or you can change the name.

2. From the warning dialog, click the "Continue" button to confirm that you want to proceed with editing your program in RUN mode. STEP 7-Micro/WIN SMART V3 uploads the program that is currently stored in the CPU and displays it in the program editor, where you can make the changes you need.

After you make the desired editing changes, you must download them before they can take effect in the CPU. Once you initiate a download, you cannot perform other tasks in STEP 7-Micro/WIN SMART V3 until the download completes.

Examine the output window to see whether any compile errors exist (for instance, duplicate EU or ED numbers). You can double-click the error message to edit the offending network in the program editor.

Specifying CPU allocation (background time)

During a program edit in RUN mode, the CPU requires time to compile the modified program in the background while it continues to execute the currently loaded program. In the system block (Page 181), you can configure the amount of background time that is available for the compilation. Note that you can only download the system block when the CPU is in STOP mode.

7.10 Features for debugging your program

STEP 7-Micro/WIN SMART V3 provides the following features to help you debug your program:

- Adding bookmarks in your program to make it easy to move back and forth between lines of a long program
- Tracking references in your program with the cross reference table (Page 737)
- Using a status chart (Page 742) to display PLC data values and status
- Displaying status in the program editor (Page 739)

For more information about debugging your program, refer to the chapter on diagnostics and troubleshooting (Page 737).

PLC device configuration

8

8.1 Configuring the operation of the PLC system

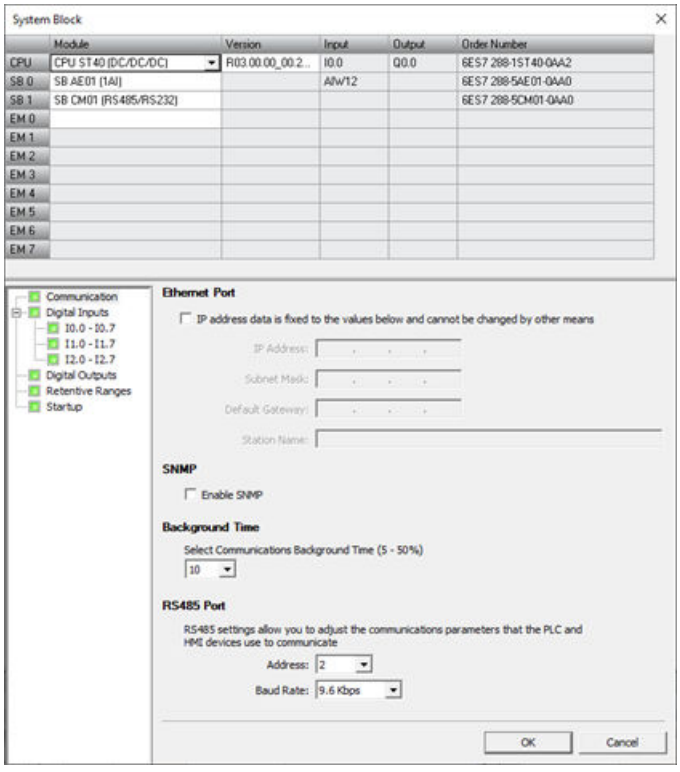
8.1.1 System block

The system block provides configuration of the S7-200 SMART CPU, signal boards, and expansion modules.

Use one of the following methods to view and edit the system block to set up CPU options:

- Click the "System Block" button on the navigation bar.
- Select "System Block" from the Component drop-down list in the Windows area of the View menu ribbon strip.
- Select the "System Block" node, then press Enter; or double-click the "System Block" node in the project tree.

STEP 7-Micro/WIN SMART opens the system block, and displays the configuration options that are applicable for your CPU type.



Hardware configuration

The top part of the System Block dialog displays the modules that you have configured and allows you to add or delete modules. Use the drop-down lists to change, add, or delete the CPU model, signal board, and expansion modules. As you add modules, the input and output columns display the assigned input and output addresses.

Note

Optimally, select the CPU model and firmware version in the system block to be the model and firmware version of the actual CPU you plan to use. When downloading your project, if the CPU model and firmware version in the project does not match the model and firmware version of the connected CPU, STEP 7-Micro/WIN SMART issues a warning message. You can continue with the download, but if the connected CPU does not support the resources and capabilities that the project requires, a download error occurs.

Module options

The bottom part of the system block dialog displays options for the module that you select in the top part. Click any node in the configuration options tree to modify the project configuration for the selected module.

The system block includes the following configuration options for CPU modules:

- Communication (Page 183)
- Digital inputs and pulse catch bits (Page 186)
- Digital outputs (Page 189)
- Retentive Ranges (Page 190)
- Startup (Page 192)

Configuration options specific to other devices such as analog inputs (Page 193), analog outputs (Page 196), RTD analog inputs (Page 198), Thermocouple (TC) analog inputs (Page 202), RS485/RS232 CM01 communications signal board (Page 205), Battery BA01 signal board (Page 207), and additional digital inputs and outputs are accessible from the system block when you add those modules.

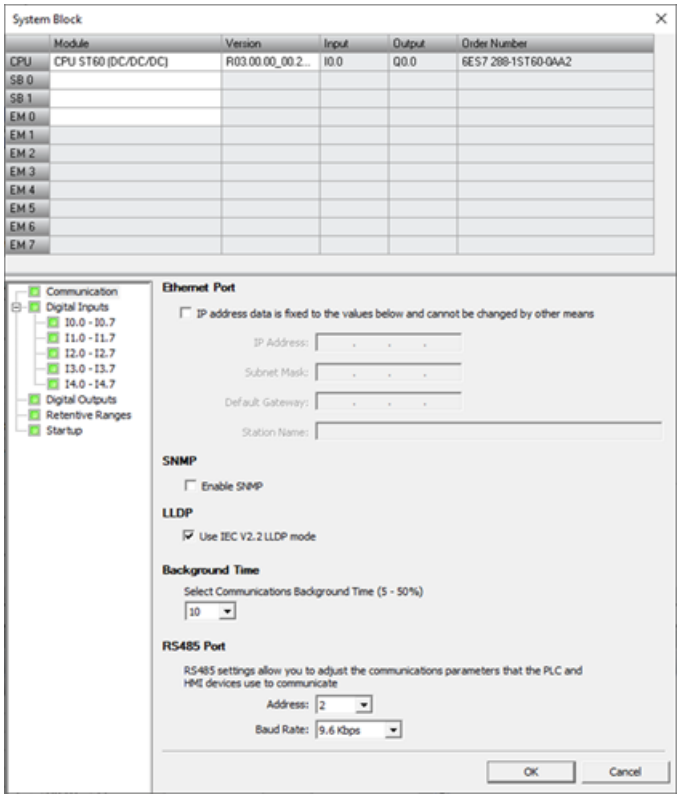
You must establish communications between STEP 7-Micro/WIN SMART and your CPU before you can download or upload a system bock.

You can then download a modified system block in order to provide the CPU with a new system configuration. New properties that you enter take effect when you download (Page 82) the modifications to the CPU.

Alternatively, you can upload an existing system block from the CPU in order to make your STEP 7-Micro/WIN SMART project configuration match that of the CPU.

8.1.2 Configuring communication

Click the Communication node of the system block (Page 181) dialog to configure the Ethernet port, SNMP, background time, and RS485 port.



Ethernet port

If you want your CPU to obtain its Ethernet network port information from the project, click the "IP address data is fixed to the values below and cannot be changed by other means" checkbox. You can then enter the following Ethernet network information:

- IP address: Each device must have an Internet Protocol (IP) address. The device uses this address to deliver data on a more complex, routed network.
- Subnet mask: A subnet is a logical grouping of connected network devices. Nodes on a subnet are usually located in close physical proximity to each other on a Local Area Network (LAN). The subnet mask defines the boundaries of an IP subnet. A subnet mask of 255.255.255.0 is generally suitable for a local network.

- Default gateway: Gateways (or IP routers) are the link between LANs. Using a gateway, a computer in a LAN can send messages to other networks, which might have other LANs behind them. If the destination of the data is not within the LAN, the gateway forwards the data to another network or group of networks where it can be delivered to its destination. Gateways rely on IP addresses to deliver and receive data packets.
- Station name: The station name is the name by which this CPU is identified on the network. Use a name that helps you identify the CPU on the Communications dialog.

Note

The station name follows the standard DNS (Domain Name System) naming conventions. The S7-200 SMART CPUs limit the station name to a maximum of 63 characters, which can consist of the lower case letters a through z, the digits 0 through 9, the hyphen character (minus sign) and the period character.

The CPU prohibits certain names:

- The station name must not have the format n.n.n.n where n is a value of 0 through 999.
- You cannot begin the station name with the string port-nnn or the string port-nnn-nnnnn where n is a digit 0 through 9. For example, port-123 and port-123-45678 are illegal station names. A station name cannot start or end with a hyphen or period.

SNMP

If you want to use SNMP to monitor and manage network devices, enable SNMP by ticking the box in front of 'Enable SNMP'.

Use IEC V2.2 LLDP mode

Default for devices which support IEC V2.3 mode:

- The option "Use IEC V2.2 LLDP" mode is disabled (no checkmark). The PROFINET interface is set to IECV2.3 mode (IEC 61158 V2.3).
- Select this option to set the PROFINET interface to IEC V2 2 mode. This may be necessary when you have split a system into several dependent projects (e.g., in case of an I-device configuration).

Default for devices which do not support IEC V2.3 mode:

- The option "Use IEC V2.2 LLDP mode" is selected (checkmark is set) and cannot be changed.
- If all devices with PROFINET interfaces are completely configured and the parameters assigned in the current project, keep the default setting.

Background time

You can configure the percentage of the scan cycle time that is dedicated to processing the communication requests. As you increase the percentage of time that is dedicated to processing communication requests, you are increasing scan time, which makes your control process run more slowly. The scan time only increases if there are communication requests to process.

The default percentage of the scan time dedicated to processing communication requests is set to 10%. This setting provides a reasonable compromise for processing compilation/ status

operations, while minimizing the impact to your control process. You can adjust this value by 5% increments up to a maximum of 50%.

As you add more communication partners to the S7-200 SMART CPU, additional background time is required to handle the requests from those partners. GET and PUT instructions need additional resources to create and maintain connections to other devices. The EM DP01 PROFIBUS DP module requires additional communication background time if you have HMIs or other CPUs communicating with the S7-200 SMART CPU through the EM DP01. Open User Communication (OUC) also adds an additional load to the CPU and may require additional background time.

RS485 port

Use these settings to adjust the communication parameters for system protocols for the onboard RS485 port. The system protocols are used when connecting to a programming device or HMI devices:

- RS485 port address: Click the scroll buttons to enter the desired CPU address (1-126). The default port address is 2.
- Baud Rate: Choose the desired data baud rate from the dropdown list (9.6 Kbps, 19.2 Kbps, or 187.5 Kbps).

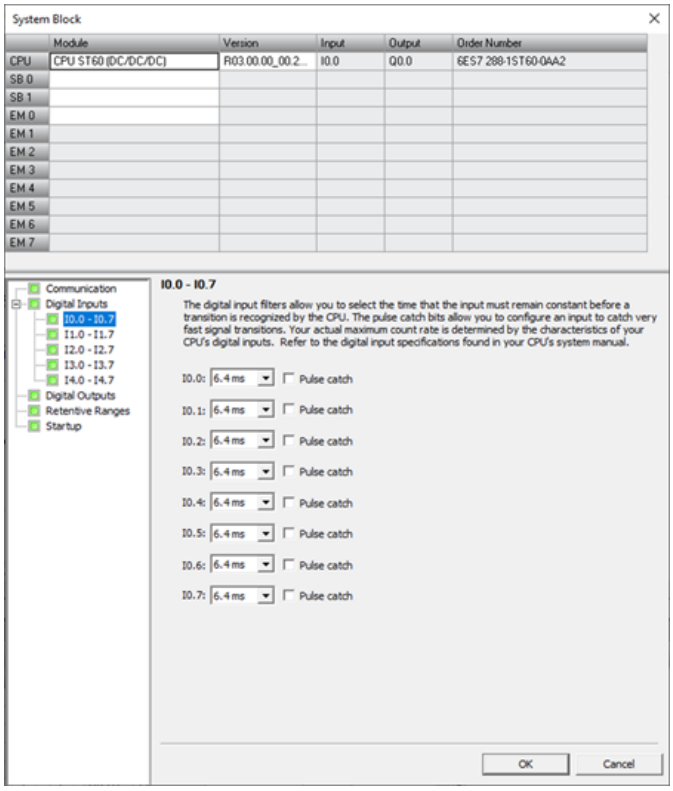
Note

You can make the following RS485 communication connections for S7-200 SMART CPUs:

- Use a USB-PPI cable to program all CPU models through any serial port, including the RS485 port, the signal board port, and the DP01 PROFIBUS port.
- Use the RS485 and RS232 ports for HMI access (Data read/write) and Freeport communications.

8.1.3 Configuring the digital inputs

Click the Digital Inputs node of the system block (Page 181) dialog to configure digital input filters and pulse catch bits.



Digital input filters

You can filter digital input signals by setting an input delay time. This delay helps to filter noise on the input wiring that could cause inadvertent changes to the states of the inputs. When an input state change occurs, the input must remain at the new state for the duration of the delay time in order to be accepted as valid. The filter rejects noise impulses and forces input lines to stabilize before the data is accepted.

The S7-200 SMART CPU 3.0 allows you to select an input delay time for all of its digital input points. The quantity of input points available is dependent upon your CPU model.

The first sixteen input points (I0.0 through I0.7 and I1.0 through I1.7) support an expanded set of delay time choices (selectable to one of seven settings in the range of 0.2 ms to 12.8 ms or one of seven settings in the range of 0.2 µs to 12.8 µs).

The default filter time for all input points is 6.4 ms.

To set an input delay, follow these steps:

1. Select the time of the delay from the drop-down list beside one or more inputs.
2. Click the OK button to enter the selections.

WARNING

Risks with changes to filter time for digital input channel

If the filter time for a digital input channel is changed from a previous setting, a new "0" level input value may need to be presented for up to 12.8 ms accumulated duration before the filter becomes fully responsive to new inputs. During this time, short "0" pulse events of duration less than 12.8 ms may not be detected or counted.

This changing of filter times can result in unexpected machine or process operation, which may cause death or serious injury to personnel, and/or damage to equipment.

To ensure that a new filter time goes immediately into effect, a power cycle of the CPU must be applied.

Pulse catch bits

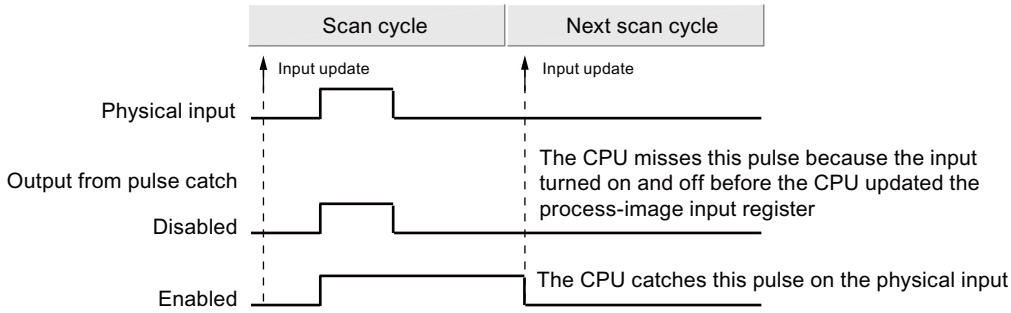
The S7-200 SMART CPU provides a pulse catch feature for digital input points. The pulse catch feature allows you to capture high-going pulses or low-going pulses that are of such a short duration that they would not always be seen when the CPU reads the digital inputs at the beginning of the scan cycle.

When pulse catch is enabled for an input, a change in state of the input is latched and held until the next input cycle update. This ensures that a pulse which lasts for a short period of time will be caught and held until the S7-200 SMART CPU reads the inputs.

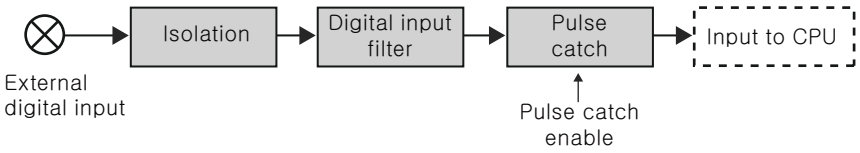
You can enable individual pulse catch operation for the first sixteen digital input points (I0.0 through I0.7 and I1.0 through I1.7), dependent upon the CPU model.

If your configuration includes an SB DT04, you can enable the two additional digital input points available on this signal board for pulse catch operation.

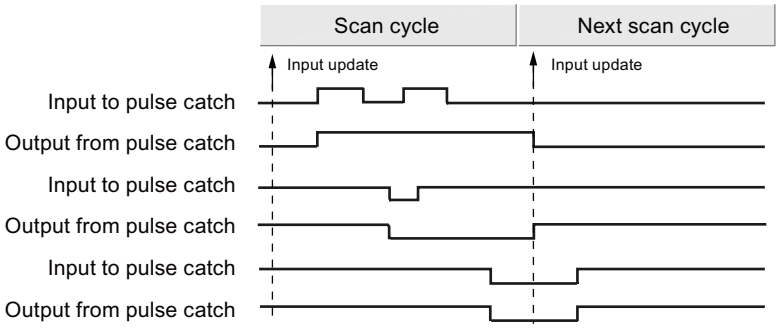
The figure below shows the basic operation of the S7-200 SMART CPU with and without pulse catch enabled:



Because the pulse catch function operates on the input after it passes through the input filter, you must adjust the input filter time so that the pulse is not removed by the filter. The figure below shows a block diagram of the digital input circuit:



The figure below shows the response of an enabled pulse catch function to various input conditions. If you have more than one pulse in a given scan, only the first pulse is read. If you have multiple pulses in a given scan, you should use the rising/falling edge interrupt events:

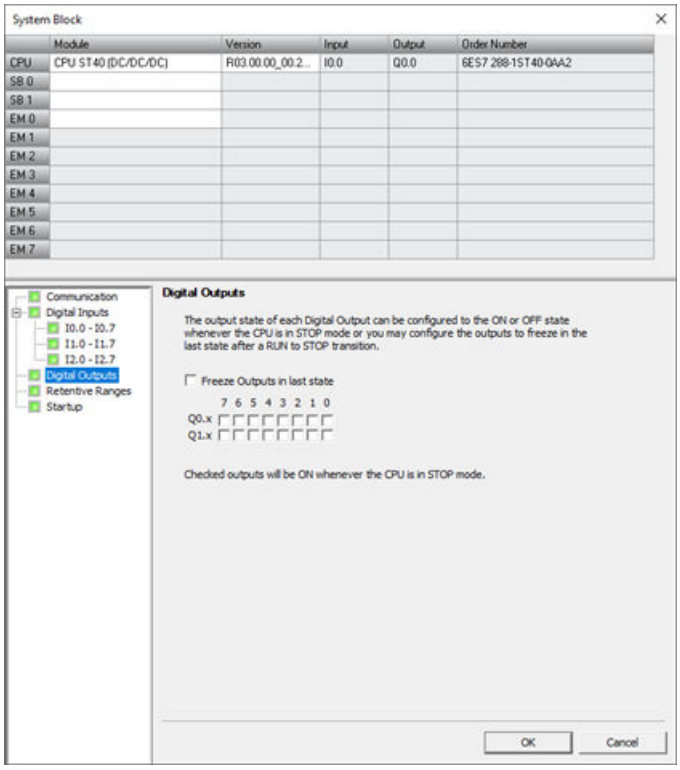


See also

S7-200 SMART CPU (Page 23)

8.1.4 Configuring the digital outputs

Click the Digital Outputs node of the system block (Page 181) to configure options for the digital outputs of the selected module.



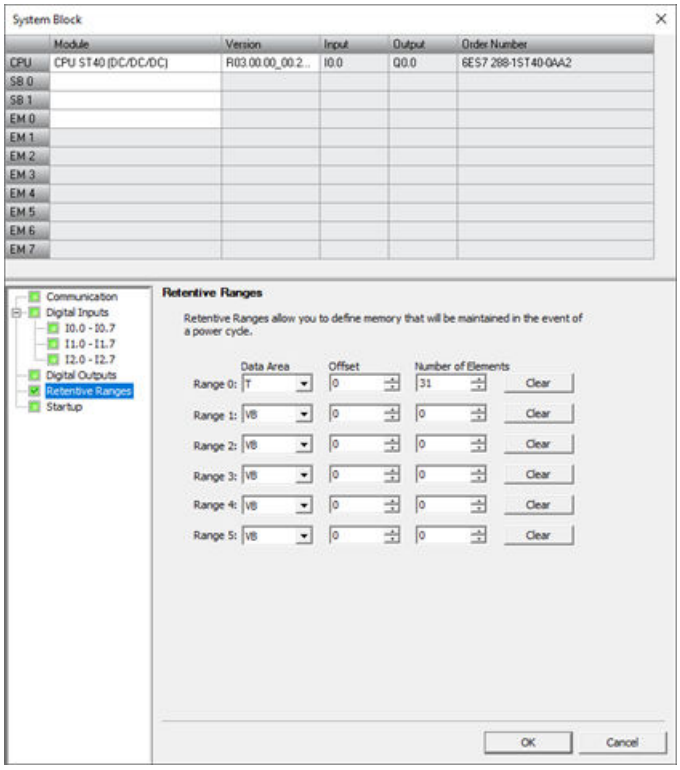
You can set digital output points to a specific value when the CPU is in STOP mode, or preserve the output states that existed before the transition to STOP mode.

You have two ways to set the digital output behavior in STOP mode:

- **Freeze Outputs in last state:** Click this checkbox to have all digital outputs frozen in their last states at the time of a RUN-to-STOP transition.
- **Substitute value:** If the Freeze Outputs in last state checkbox is not checked, this table allows you to select the desired state of each output whenever the CPU is in STOP mode. Click the checkbox for each output you want set to ON (1). The default substitute value for digital outputs is OFF (0).

8.1.5 Configuring the retentive ranges

Click the Retentive Ranges node of the system block (Page 181) dialog to configure ranges of memory that will be retained following a power cycle.



Configure the areas of memory you want to retain through power cycles. Enter new values for V, M, T, or C memory.

You can define ranges of addresses in the following memory areas to be retentive: V, M, T, or C . For timers, only the retentive timers (TONR) can be retained, and, for both timers and counters, only the current value can be retained (timer and counter bits are cleared on each power-up).

By default, the CPU has no defined retentive memory areas, but you can configure the retentive ranges:

- The S7-200 SMART CPU models CPU SR40, CPU ST40, CPU SR60, and CPU ST60 have a maximum of 40 Kbytes of retentive memory, including DB2 and V,M,T, C.
- The DB2 region can only be configured in the global variable table.

Data retention after CPU power interruption

The CPU performs the following actions regarding retentive memory at power down and power up:

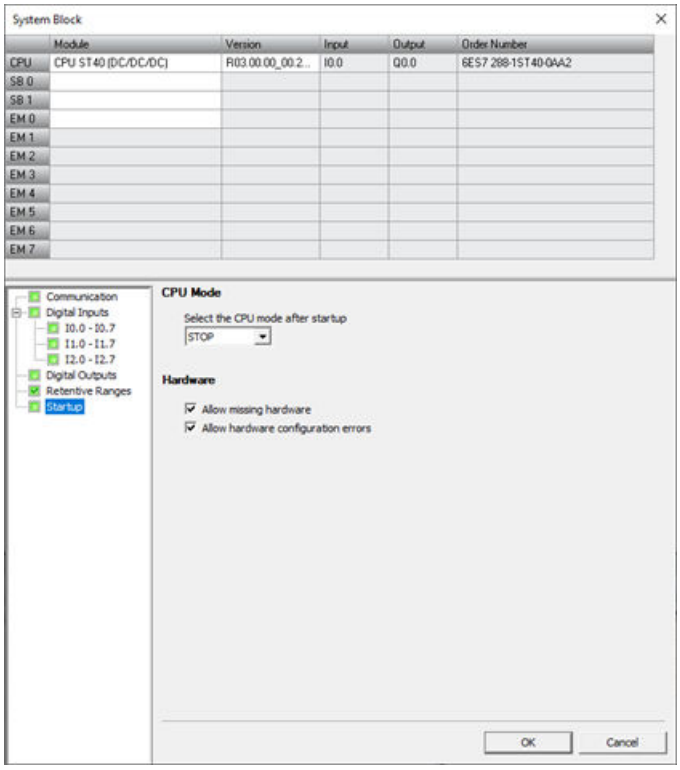
- **At power down:**
  The CPU saves the memory ranges designated as retentive to permanent memory.
- **At power up:**
  The CPU first clears V, M, C, T or DB2 memory, copies any initial values from the data block to V memory, and then copies the saved retentive values from permanent memory to RAM.

S7-200 SMART CPU memory addresses for retentive ranges

Data type	Desc.	CPU SR40 CPU ST40	CPU SR60 CPU ST60
V	Data Memory	VB0-VB81919	VB0-VB102399
T	Timers	T0-T31, T64-T95, T256 - T261, T268 - T291, T316 - T413	
C	Counters	C0-C511	
M	Flag bits	MB0-MB63	

8.1.6 Configuring the startup options

Click the Startup node of the system block (Page 181) dialog to configure startup options for the PLC.



CPU mode

From this dialog you can select the mode for the CPU after a startup. You have one of the following three choices:

- STOP
  The CPU shall always enter STOP mode after a power up or restart (default selection).
- RUN
  The CPU shall always enter RUN mode after a power up or restart. For most applications, especially those where the CPU operates independently without a connection to STEP 7-Micro/WIN SMART, the RUN startup mode selection is the correct choice.
- LAST
  The CPU shall enter the operating mode that existed prior to the last power up or restart. This selection can be useful during program development or commissioning. Be aware that a running CPU can enter STOP mode for a variety of reasons, such as the failure of an expansion module, occurrence of a scan watchdog timeout, the insertion of a memory card, or an erratic power up event. Once the CPU enters STOP mode, it will continue to enter STOP mode each time the CPU powers up. You must restore the CPU back to RUN mode (Page 82) from STEP 7-Micro/WIN SMART.

Hardware options

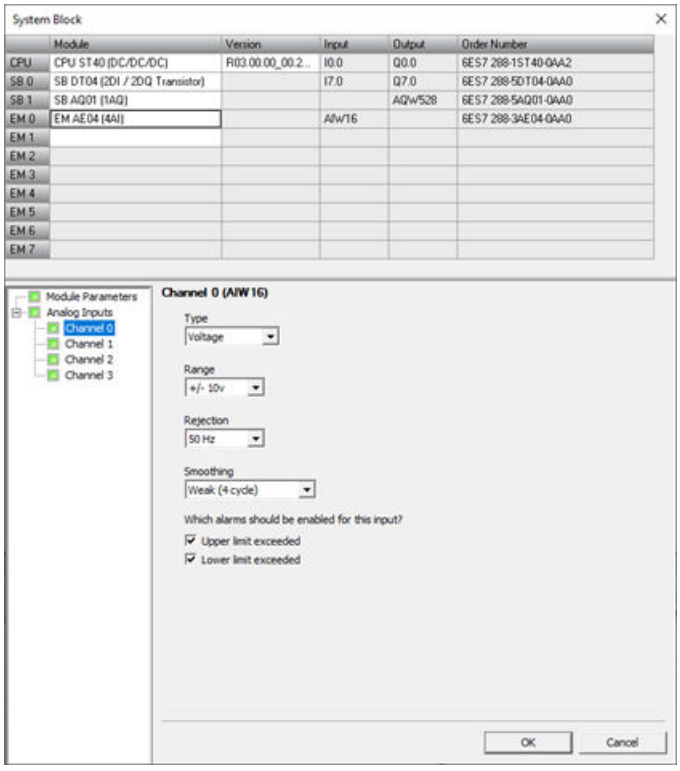
You can also configure the CPU to allow RUN mode operation under the following hardware conditions:

- One or more devices specified in the hardware configuration stored in the CPU are missing.
- A difference exists between the hardware configuration stored in the CPU and the devices actually present, resulting in configuration errors (for example, discrete input module in place of a configured discrete output module).

If you deselect one or both of the selections, the CPU is prohibited from entering RUN mode if any of the disallowed conditions are true.

8.1.7 Configuring the analog inputs

Click the Analog Inputs node of the system block (Page 181) dialog to configure options for an analog input module that you have selected in the top section.



Analog type configuration

For each analog input channel, you configure the type to be either voltage or current. The type selected for the even-numbered channels also applies to the odd-numbered channels: the type selection for Channel 0 also applies to Channel 1, and the type selection for Channel 2 also applies to Channel 3.

Range

You then configure either the voltage range or the current range for the channel. You can choose one of the following value ranges:

- +/- 2.5v
- +/- 5v
- +/- 10v
- 0 - 20ma

Rejection

Fluctuations in analog input values can also be caused by the response time of the sensor, or the length and condition of the wires carrying the analog signal to the module. In such cases, the fluctuating values could be changing too rapidly for the program logic to respond effectively. You can configure the module to reject signals to eliminate or minimize noise at the following frequencies:

- 10 Hz
- 50 Hz
- 60 Hz
- 400 Hz

Smoothing

You can also configure the module to smooth the analog input signal over a configured number of cycles, thus presenting an averaged value to the program logic. You have four choices for the smoothing algorithm:

- None (no smoothing)
- Weak
- Medium
- Strong

Alarm configuration

You select whether to enable or disable the following alarms for the selected channel of the selected module:

- Upper limit exceeded (value > 32511)
- Lower limit exceeded (value < -32512)
- User power (Configured in the system block "Module Parameters" node; see the figure below.)

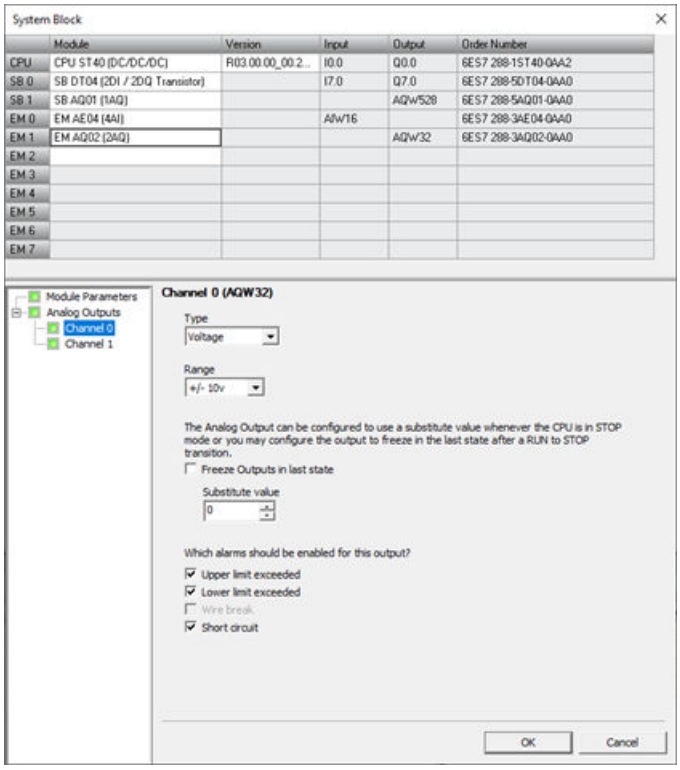
8.1.8 Reference to the analog inputs technical specifications

For further information on analog Input configuration options, refer to the following technical specifications:

- Range: "Measurement ranges of the analog inputs for voltage and current (SB and SM)" (Page 1026)
- Rejection: "Sample time and update times for the analog inputs" (Page 1026)
- Smoothing: "Step response of the analog inputs" (Page 1025)

8.1.9 Configuring the analog outputs

Click the Analog Outputs node of the system block (Page 181) dialog to configure options for an analog output module that you have selected in the top section.



Analog type configuration

For each analog output channel you configure the type to be either voltage or current.

Range

You then configure either the voltage range or the current range for the channel. You can choose one of the following value ranges:

- +/- 10 V
- 0 - 20 mA

Output behavior in STOP mode

You can set analog output points to a specific value when the CPU is in STOP mode or preserve the output states that existed before the transition to STOP mode.

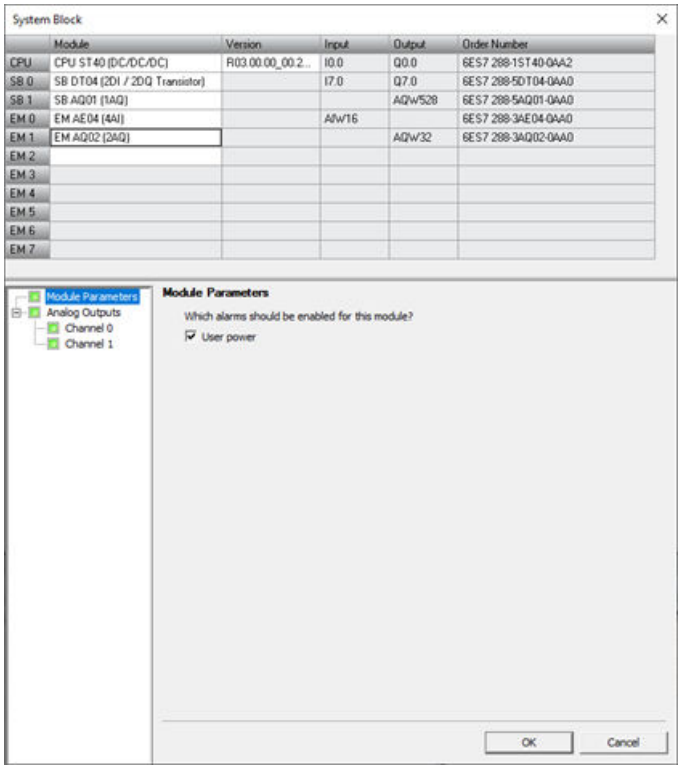
You have two ways to set the analog output behavior in STOP mode:

- Freeze outputs in last state: Click this checkbox to have all analog outputs frozen to their last values on a RUN-to-STOP transition.
- Substitute value: If the "Freeze outputs in last state" checkbox is not checked, you can enter a value (-32512 to 32511) that is applied to the output whenever the CPU is in STOP mode. The default substitute value is 0.

Alarm configuration

You select whether to enable or disable the following alarms for the selected channel of the selected module:

- Upper limit exceeded (value > 32511)
- Lower limit exceeded (value < -32512)
- Wire break (for current channels only)
- Short circuit (for voltage channels only)
- User power (Configured in the system block "Module Parameters" node; see the figure below.)

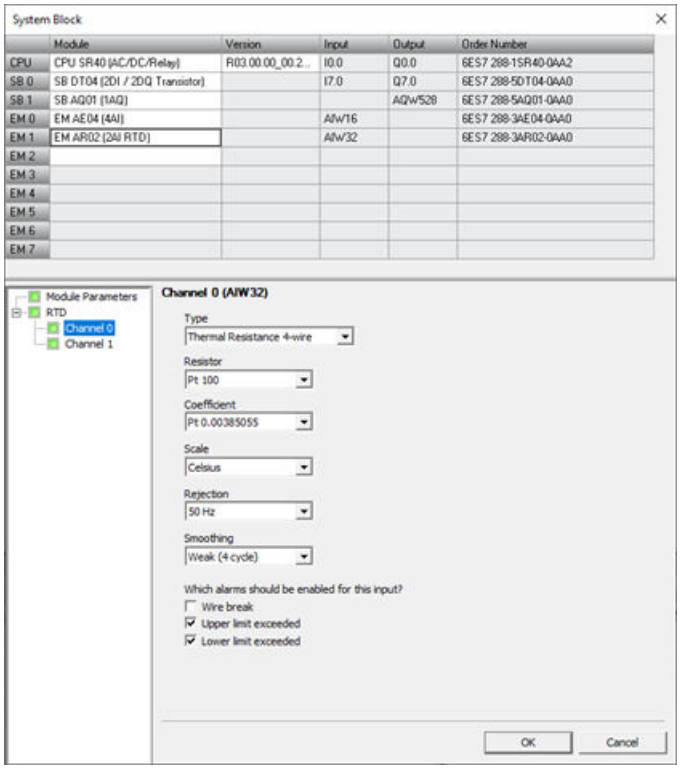


8.1.10 Reference to the analog outputs technical specifications

For further information on analog output range configuration, refer to the "Measurement ranges of the analog outputs for voltage and current (SB and SM)" (Page 1026) technical specification.

8.1.11 Configuring the RTD analog inputs

Click the RTD analog Input node of the system block (Page 181) dialog to configure options for an RTD analog input module that you have selected in the top section.



The RTD analog input module provides a current at terminals I+ and I- for resistance measurements. The current is fed to the resistance for measuring its voltage potential. The current cables must be wired directly to the resistance thermometer/resistor.

Measurements programmed for 4-or 3-wire connections compensate for line resistance and return considerably higher accuracy compared to 2-wire connections.

RTD type configuration

For each RTD input channel, you configure the type, choosing one of the following options:

- Resistance 4-wire
- Resistance 3-wire
- Resistance 2-wire
- Thermal Resistance 4-wire
- Thermal Resistance 3-wire
- Thermal Resistance 2-wire

Resistor

Depending upon the RTD type that you select, you can configure the following RTD resistors for the channel:

Table 8-1 RTD types and available resistors

RTD types	RTD resistors	
• Resistance 4-wire • Resistance 3-wire • Resistance 2-wire Note: For these RTD types and resistors, you cannot configure temperature coefficients or temperature scales.	• 48 ohms • 150 ohms • 300 ohms • 600 ohms • 3000 ohms	
• Thermal Resistance 4-wire • Thermal Resistance 3-wire • Thermal Resistance 2-wire	• Pt 10 • Pt 50 • Pt 100 • Pt 200 • Pt 500 • Pt 1000 • LG-Ni 1000	• Ni 100 • Ni 120 • Ni 200 • Ni 500 • Ni 1000 • Cu 10 • Cu 50 • Cu 100

Coefficient

Depending upon the RTD resistor that you select, you can configure the following RTD temperature coefficients for the channel:

RTD resistors	RTD temperature coefficients
• 48 ohms • 150 ohms • 300 ohms • 600 ohms • 3000 ohms	Note: For these RTD resistors, you cannot configure temperature coefficients or temperature scales.
• Pt 10 • Pt 50	• Pt 0.00385055 • Pt 0.003910
• Pt 100 • Pt 500	• Pt 0.00385055 • Pt 0.003916 • Pt 0.003902 • Pt 0.003920 • Pt 0.003910

RTD resistors	RTD temperature coefficients
• Pt 200 • Pt 1000	• Pt 0.00385055 • Pt 0.003916 • Pt 0.003902 • Pt 0.003920
• Ni 100	• Ni 0.006170 • Ni 0.006180 • Ni 0.006720
• Ni 120 • Ni 200 • Ni 500 • Ni 1000	• Ni 0.006180 • Ni 0.006720
• Cu 10	• Cu 0.00426 • Cu 0.00428 • Cu 0.00427
• Cu 50 • Cu 100	• Cu 0.00426 • Cu 0.00428
• LG-Ni 1000	• LG-Ni 0.005000

Scale

You configure a temperature scale for the channel, choosing one of the following options:

- Celsius
- Fahrenheit

Note

For the "Resistance 4-wire", "Resistance 3-wire", and "Resistance 2-wire" RTD types and associated resistors, you cannot configure temperature coefficients or temperature scales.

Rejection

Fluctuations in RTD analog input values can also be caused by the response time of the sensor, or the length and condition of the wires carrying the RTD analog signal to the module. In such cases, the fluctuating values could be changing too rapidly for the program logic to respond effectively. You can configure the module to reject signals to eliminate or minimize noise at the following frequencies:

- 10 Hz
- 50 Hz
- 60 Hz
- 400 Hz

Smoothing

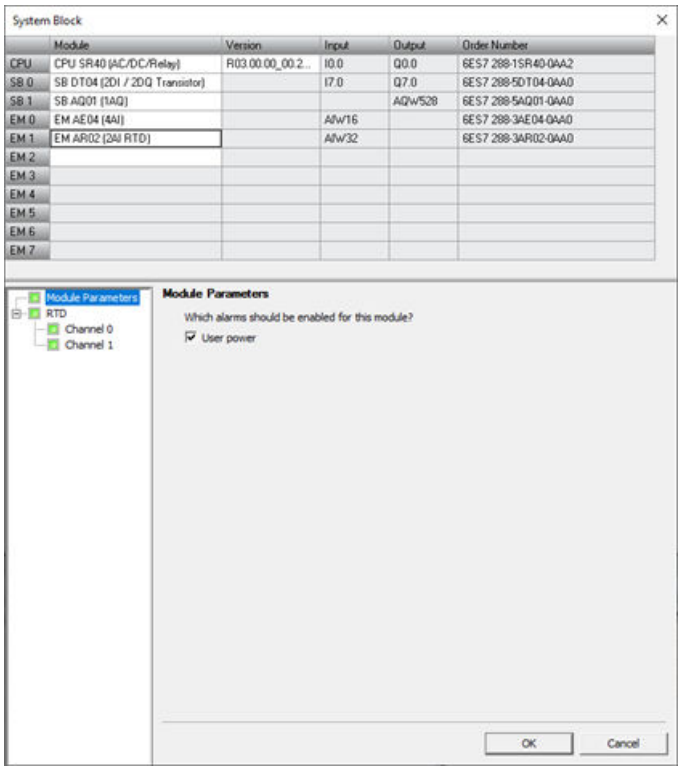
You can also configure the module to smooth the RTD analog input signal over a configured number of cycles, thus presenting an averaged value to the program logic. You have four choices for the smoothing algorithm:

- None
- Weak
- Medium
- Strong

Alarm configuration

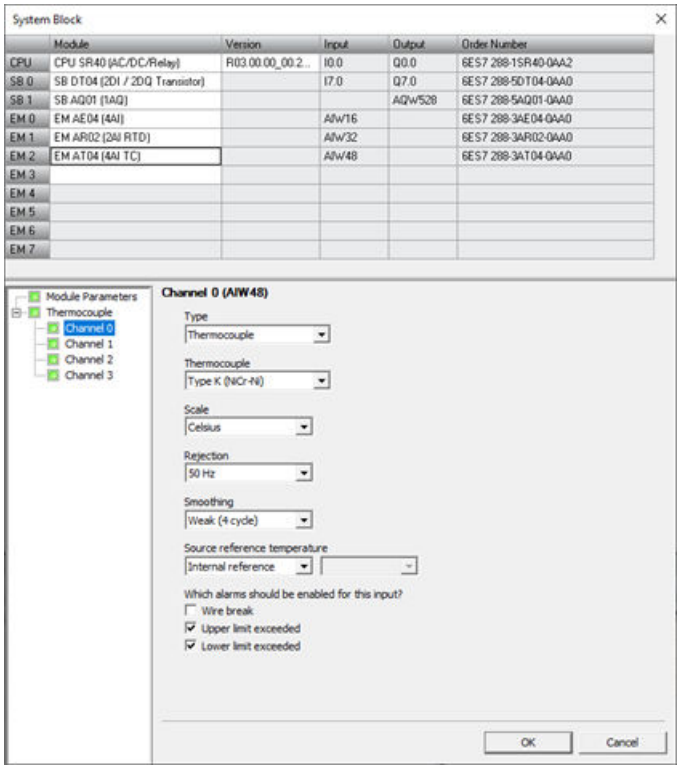
You select whether to enable or disable the following alarms for the selected channel of the selected RTD module:

- Wire break
- Upper limit exceeded
- Lower limit exceeded
- User power (Configured in the system block "Module Parameters" node; see the figure below.)



8.1.12 Configuring the TC analog inputs

Click the TC (Thermocouple) analog input node of the system block (Page 181) dialog to configure options for a TC analog input module that you have selected in the top section.



The TC analog expansion module measures the value of voltage connected to the module inputs.

Thermocouple type configuration

For each TC analog input module channel, you configure the type, choosing one of the following options:

- Thermocouple
- Voltage

Thermocouple

Depending upon the thermocouple type that you select, you can configure the following thermocouples for the channel:

- Type B (PtRh-PtRh)
- Type N (NiCrSi-NiSi)
- Type E (NiCr-CuNi)
- Type R (PtRh-Pt)

- Type S (PtRh-Pt)
- Type J (Fe-CuNi)
- Type T (Cu-CuNi)
- Type K (NiCr-Ni)
- Type C (W5Re-W26Re)
- TXK/XK (TXK/XK(L))

Scale

You configure a temperature scale for the channel, choosing one of the following options:

- Celsius
- Fahrenheit

Rejection

Fluctuations in thermocouple analog input values can also be caused by the response time of the sensor, or the length and condition of the wires carrying the thermocouple analog signal to the module. In such cases, the fluctuating values could be changing too rapidly for the program logic to respond effectively. You can configure the TC analog input module to reject signals to eliminate or minimize noise at the following frequencies:

- 10 Hz
- 50 Hz
- 60 Hz
- 400 Hz

Smoothing

You can also configure the module to smooth the thermocouple analog input signal over a configured number of cycles, thus presenting an averaged value to the program logic. You have four choices for the smoothing algorithm:

- None
- Weak
- Medium
- Strong

Source reference temperature

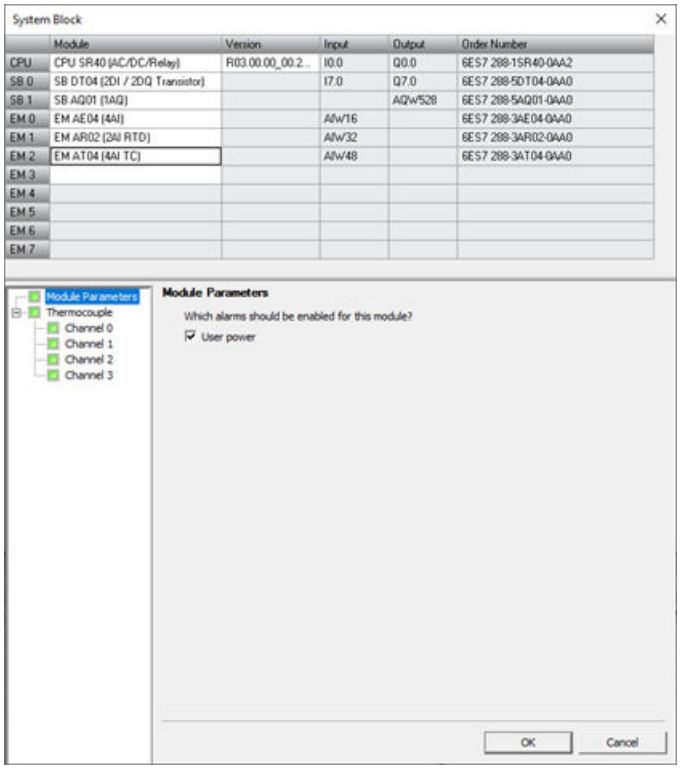
You configure a source reference temperature for each TC analog input module channel, choosing one of the following options:

- Set by parameter
- Internal reference

Alarm configuration

You select whether to enable or disable the following alarms for the selected channel of the selected TC analog input module:

- Wire break
- Upper limit exceeded
- Lower limit exceeded
- User power (Configured in the system block "Module Parameters" node; see the figure below.)



Basic operation of a thermocouple

Thermocouples are formed whenever two dissimilar metals are electrically bonded to each other. A voltage is generated that is proportional to the junction temperature. This voltage is small; one microvolt could represent many degrees. Measuring the voltage from a thermocouple, compensating for extra junctions, and then linearizing the result forms the basis of temperature measurement using thermocouples.

When you connect a thermocouple to the TC analog input module, the two dissimilar metal wires are attached to the module at the module signal connector. The place where the two dissimilar wires are attached to each other forms the sensor thermocouple.

Two more thermocouples are formed where the two dissimilar wires are attached to the signal connector. The connector temperature causes a voltage that adds to the voltage from the sensor thermocouple. If this voltage is not corrected, then the temperature reported will deviate from the sensor temperature.

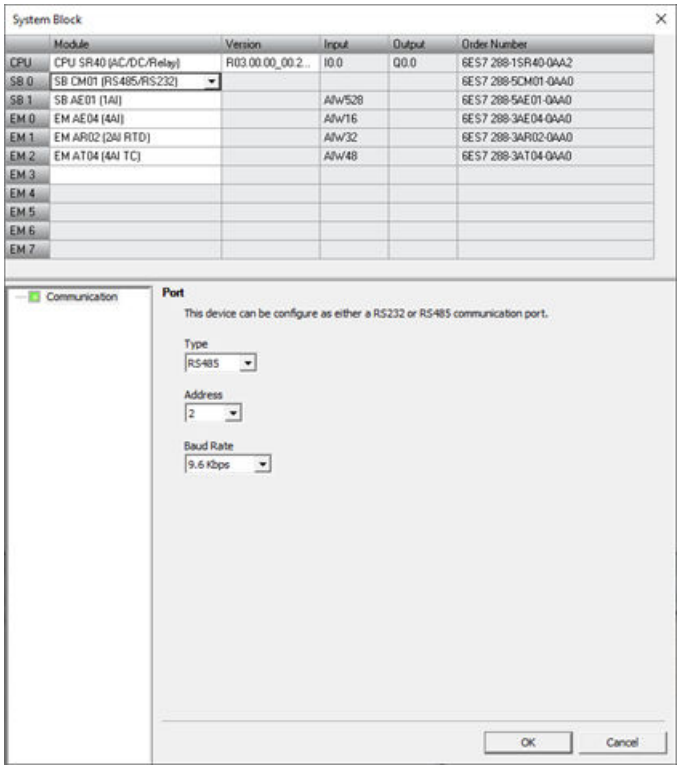
Cold junction compensation is used to compensate for the connector thermocouple. Thermocouple tables are based on a reference junction temperature, usually zero degrees Celsius. The cold junction compensation compensates the connector to zero degrees Celsius. The cold junction compensation restores the voltage added by the connector thermocouples. The temperature of the module is measured internally, and then converted to a value to be added to the sensor conversion. The corrected sensor conversion is then linearized using the thermocouple tables.

For optimum operation of the cold junction compensation, the thermocouple module must be located in a thermally stable environment. Slow variation (less than 0.1 °C/minute) in ambient module temperature is correctly compensated within the module specifications. Air movement across the module will also cause cold junction compensation errors.

If better cold junction error compensation is needed, an external iso-thermal terminal block may be used. The thermocouple module provides for use of a 0 °C referenced or 50 °C referenced terminal block.

8.1.13 Configuring the RS485/RS232 CM01 communications signal board

Click the CM01 communications signal board node of the system block (Page 181) dialog to configure options for an RS485/RS232 CM01 communications signal board that you have selected in the top section. You can configure a maximum of two signal boards.



CM01 signal board type configuration

You configure the CM01 signal board type from the dropdown list, choosing one of the following options:

- RS485
- RS232

Address

Click the scroll buttons to enter the desired port address (1-126), for the RS485 or RS232 port: The default port address is 2.

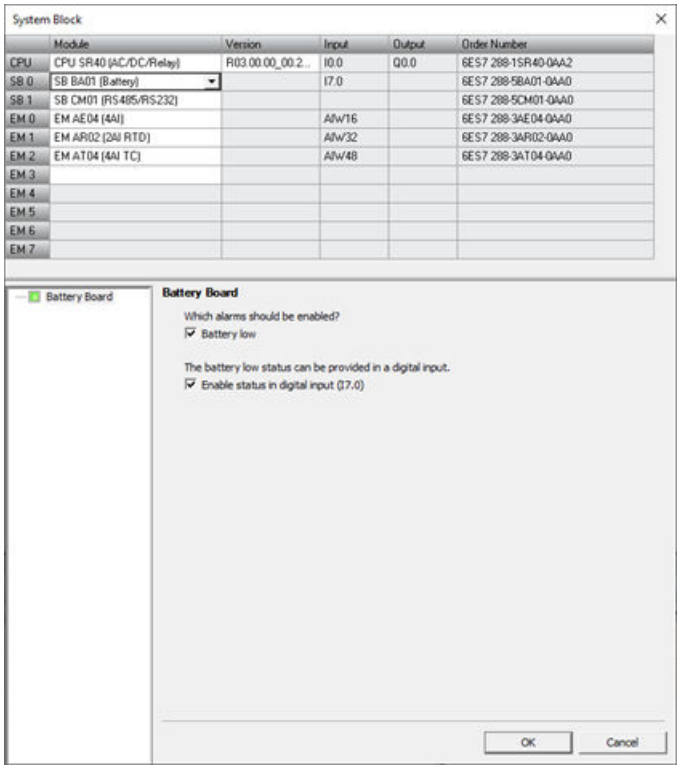
Baud rate

Choose the desired data baud rate from the dropdown list:

- 9.6 Kbps
- 19.2 Kbps
- 187.5 Kbps

8.1.14 Configuring the BA01 battery signal board

Click the BA01 battery signal board node of the system block (Page 181) dialog to configure options for a BA01 battery signal board that you have selected in the top section.



Enable bad diagnostic alarm

Click the “Enable bad diagnostic alarm” checkbox to trigger an alarm when the battery fails.

Enable status in digital input

Click the "Enable status in digital input" to enable a digital input to monitor the status of the signal board.

Operation of the Battery (BA01) Signal Board

The battery signal board contains a red LED that provides the customer a visual indication of the battery health. An Illuminated LED indicates a battery low condition.

The CPU automatically utilizes the real-time clock on the signal board and performs the battery test and battery health LED operation, whether or not the System block contains a configuration for the signal board.

The battery signal board System block configuration contains selections that allow the customer to report the battery low state as a diagnostic alarm and/or to report the battery state (1=battery low, 0 = battery OK) in the LSB of the configured image register input byte for the device (for example, I7.0). The customer must select the battery signal board in the

System block configuration in order to gain access to the additional battery health reporting options.

Note

The CPU supports inserting and configuring only one BA01.

8.1.15 Clearing PLC memory

To clear designated areas of PLC memory, follow these steps:

1. Ensure that the PLC is in STOP mode.
2. Click the Clear button from the Modify area of the PLC menu ribbon strip.

WARNING

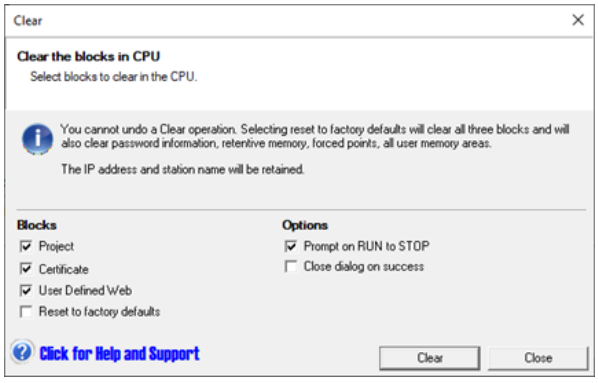
Effect of clearing PLC memory on outputs

Clearing the PLC memory affects the state of digital and analog outputs. The default is for digital and analog outputs to use a substitute value of 0. If you have defined substitute values other than 0 or chosen "Freeze" for your digital or analog outputs, you need to be aware that when you delete the system block, you are deleting the substitute and freeze information and, as a result, your outputs shall return to the default value of 0. Furthermore, if you perform a selective clear such that you keep your system block but delete your program block, then your analog outputs are frozen at their current value. Until you download a new program block, the only way to make changes to the state of the analog outputs is by means of the status chart.

If the S7-200 SMART PLC is connected to equipment when you clear the PLC memory, changes to the state of the digital outputs can be transmitted to the equipment. If you clear PLC memory without planning for the consequences to your digital and analog outputs, your equipment could operate in an unpredictable fashion, which could result in death or serious injury to personnel and/or damage to equipment.

Always follow appropriate safety precautions and ensure that your process is in a safe state before clearing the PLC memory.

3. Select what to clear - Project and User Defined Web, or select "Reset to factory defaults".
4. Click the Clear button.



Clearing the PLC memory requires the PLC to be in STOP mode and then deletes the selected blocks or resets the PLC to the factory-set defaults, depending on your selection. A clear operation does not clear the IP address, station name, or reset the time-of-day clock.

When executed, the "Reset to factory defaults" setting deletes all blocks, resets all user memory to the initial powerup state, and resets all Special Memory (Page 1067) to initial values.

What to do if you forget the PLC password

If you forget the PLC password (Page 52), you can clear the PLC memory using a reset-to-factory-defaults memory card (Page 210) that you have made for this purpose (standard CPU models).

Clearing the PLC using a reset-to-factory defaults memory card

For standard CPUs, you can clear the PLC using a previously-made reset-to-factory defaults memory card.

WARNING

Inserting a memory card into a CPU

Inserting a memory card into a CPU in RUN mode causes the CPU to automatically transition to STOP mode. You cannot change the CPU to RUN mode if a memory card is inserted.

Inserting a memory card into a running CPU can cause disruption to process operation, possibly resulting in death or severe personal injury.

Always ensure that the CPU is in STOP mode (Page 82) prior to inserting a memory card.

To clear the PLC using this card follow these steps:

1. Insert the reset-to-factory-defaults memory card. The CPU goes to STOP mode and flashes the STOP LED.
2. Power cycle the CPU. The CPU flashes the RUN/STOP LEDs until the reset is complete (about one second), and then flashes the STOP LED indicating that the reset is finished.

3. Remove the memory card.
4. Power cycle the CPU. The CPU is reset to the factory defaults. The former IP address and baud rate settings are cleared, but the time-of-day clock is unaffected.

After the CPU is reset, you can assign a new password and begin programming, or load a program from another program transfer memory card (Page 131) or from your hard disk.

Note

If you load a password-protected program from a memory card or file on your hard disk, you must enter the password to access the protected areas. You cannot access a password-protected program component without a password, nor can you clear an assigned password without entry of the password.

8.1.16 Creating a reset-to-factory-defaults memory card

You can create a memory card that will return a standard S7-200 SMART CPU to a factory default state. You can use this reset-to-factory-defaults memory card if you ever want to clear the contents of a standard CPU. To create a reset-to-factory-defaults memory card, follow these steps:

1. Using a card reader and Windows explorer, delete all contents from a microSDHC card.
2. Create a simple text file with an editor such as Notepad that contains one line with the string "RESET_TO_FACTORY". (Do not enter the quotation marks.)
3. Save this file to the microSDHC card root level under the file name "S7_JOB.S7S".
4. Label the card and store it in a safe place for future use.

See also

Clearing PLC memory (Page 208)

8.2 High-speed I/O

High-speed counters

The CPU provides integrated high-speed counter functions that count high speed external events without degrading the performance of the CPU. Refer to the "Product overview" (Page 23) chapter for the rates supported by your CPU. Dedicated inputs exist for clocks, direction control, and reset, where these functions are supported. You can select single phase, dual phase, or AB quadrature phase for varying the counting rate. For more information, refer to the description of the high-speed counter instructions (Page 325).

High-speed pulse output

The standard CPU models support high-speed pulse outputs that generate either a high-speed pulse train output (PTO) or pulse width modulation (PWM) on certain outputs. Refer to the "Product overview" (Page 23) chapter for the quantity and rates supported by your CPU.

The PTO function provides a square wave (50% duty cycle) output for a specified number of pulses (from 1 to 2,147,483,647 pulses) and a specified frequency (in Hz). You can program the PTO function to produce either one train of pulses or a pulse profile consisting of multiple trains of pulses. For example, you can use a pulse profile to control a stepper motor through a simple ramp up, run, and ramp down sequence or more complicated sequences.

The PWM function provides a fixed cycle time with a variable duty cycle output, with the cycle time and the pulse width specified in either microsecond or millisecond increments. When the pulse width is equal to the cycle time, the duty cycle is 100 percent, and the output is turned on continuously. When the pulse width is zero, the duty cycle is 0 percent, and the output is turned off.

Refer to the pulse output instruction (Page 351) for more information. The chapter on open-loop motion control provides additional information using PWM (Page 879).

Open-loop motion control

The standard CPU models support an open-loop motion control capability. Motion profiles can be constructed and executed, interactive movement can be performed under user program control, and a number of built-in reference point seek sequences are available.

Depending upon configuration, open-loop motion support in the CPU requires the use of certain CPU resources, such as high-speed outputs, high-speed counters, and edge interrupts.

Refer to the "Product overview" (Page 23) chapter for the quantity of motion axes and pulse rates supported by your CPU.

Refer to the chapter on open-loop motion control (Page 879) for a full description of the motion capabilities in your CPU.

Web server

9

S7-200 SMART has a built-in Web server which enables you to operate the S7-200 SMART from a PC or from a mobile device.

In this approach, you can access the S7-200 SMART CPU using a connected device (conventional PC, tablet or smart phone with Web browsing capabilities) through the CPU's IP address or URL.

The Web server allows you to use the mouse pointer or the touch screen, depending on the device you are using, to perform fast and easy operations on the virtualized S7-200 SMART.

S7-200 SMART supports **HTTPS** for Web server communication.

Supported Web browsers

The Web server supports the following web browsers:

For PC:

- Google Chrome with version 63.0 or higher
- Mozilla Firefox with version 67.0 or higher
- Microsoft Edge with version 18.0 or higher
- MacOS Safari with version 12.0 or higher

For mobile device:

- IOS Safari with version 13.0 or higher
- Android Chrome with version 80.0 or higher
- Android Firefox with version 68.0 or higher

Note

Note on using Web browsers:

- The most recent version of Google Chrome, Microsoft Edge, Mozilla Firefox and Apple Safari provide better usability and security.
- User's privacy data, such as user name, password, cookie, temporary files, history and web form information are managed by browsers. The proper security protect strategy shall be configured in browser settings.

Configure the Web server

Before you can use the Web server, you need to complete the following tasks:

- Configuring the Web server with the Web server wizard (Page 215)
- Configuring the HTTPS certificates for the Web server with certificate management (Page 218)

Standard Web pages

The S7-200 SMART includes standard Web pages that you can access from the web browser of your PC or from a mobile device:

- Introduction: entry point to the web pages
- Start page: general information about the CPU
- Module information: information about the local and remote modules
- Watch table: allows the user to view the watch tables that you previously configured in the Web server wizard
- Data log: allows the user to view a list of all data logs on the CPU, download a data log from the CPU to your computer, clear a data log from the CPU, and retrieve or clear a data log from the CPU
- Event log: allows the user to view a list of all event logs on the CPU, or to download a data log from the CPU to your computer
- Status chart: you can edit the chart rows to add the CPU data addresses you want to monitor from the Web server
- Set clock: set the CPU time

Note

If your device is under vulnerability risk, Siemens suggests that you clear the certificates (Page 208) and regenerate them for the CPU.

User defined Web pages

The S7-200 SMART supports you to access user defined Web pages and get the CPU runtime information. To create user defined Web pages, you can use the SMART Web Editor (SWE) tool to create the project and download the user defined web resource files to the connected CPU; or you can use other HTML authoring software of your choice. Refer to User defined Web pages (Page 236) chapter for specific infomation.

Refer to *SMART Web Editor Online Help* for detailed information about this tool.

You can access the user defined Web pages from a PC or mobile device through the connected CPU's IP address or URL, or through the User define home tab on the standard Web page.

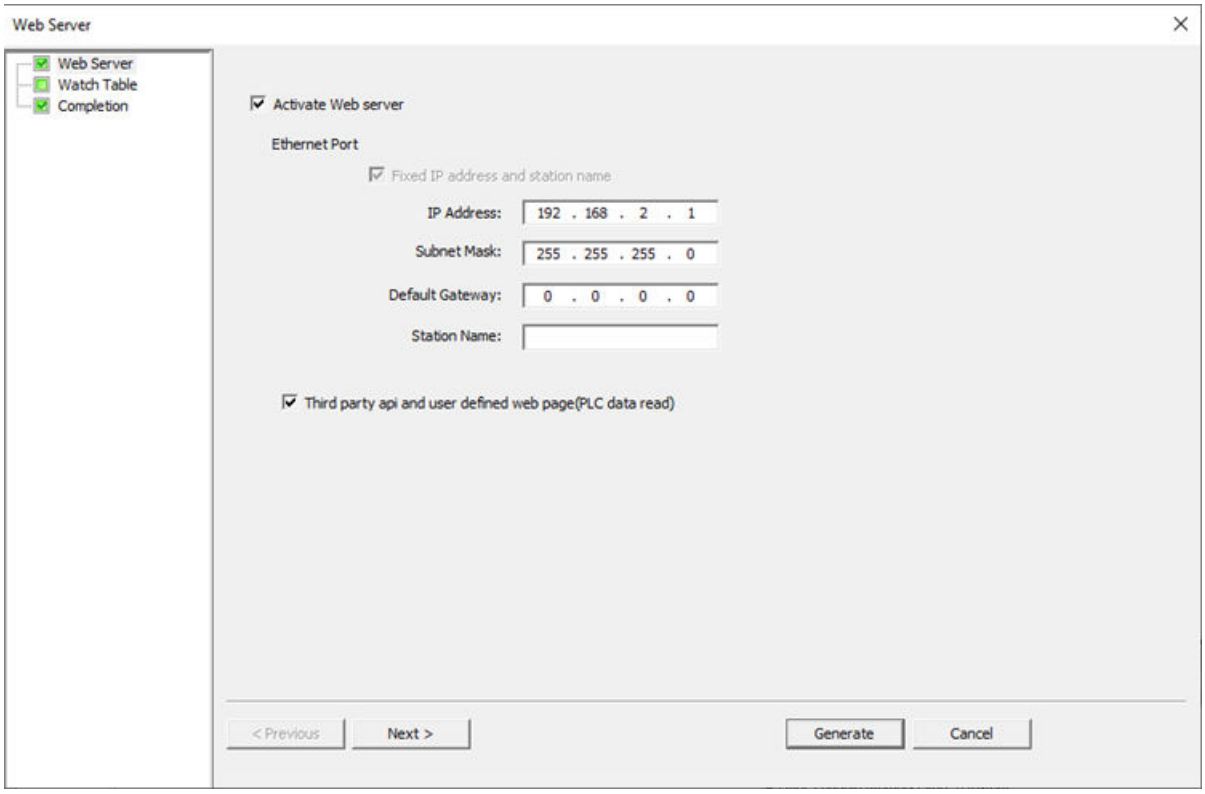
9.1 Configuring the Web server with the Web server wizard

9.1.1 Activating the Web server

You can activate the Web server of a S7-200 SMART CPU module in STEP 7-Micro/WIN SMART V3 from the Web server wizard.

To activate the Web server, follow these steps:

1. Open the Web server wizard.
2. In the Web Server window, select "Activate Web Server".
3. Enter the IP address and station name (optional) of the CPU module to which you intend to connect.



Note

If you activate the Web server, the S7-200 SMART CPU can only use a fixed IP.

Note

Always warm start or power cycle the CPU after changing the name or IP address through PPI download. Otherwise the change will not take effect.

4. Select "Third party api and user defined web page (PLC data read)" to set read permission of the connected CPU data on user defined Web pages.
5. Click "Next" to continue the Web server configuration or click "Generate" to save and complete the configuration.

WARNING

Unauthorized access to S7-200 SMART device through the Web server

Unauthorized access to the device or changing PLC variables to invalid values could disrupt process operation and could result in death, severe personal injury and/or property damage.

Because enabling the Web server allows authorized users to perform operating mode changes and writes to PLC data, Siemens recommends that you observe the following security practices:

- Set proper privileges for users.
- Perform error-checking and range-checking on your variables in your program logic because Web page users can change PLC variables to invalid values.
- Use a secure Virtual Private Network (VPN) to connect to the S7-200 SMART PLC Web server from a location outside of your protected network.

9.1.2 Configuring the watch tables on the Web server

From the Web server wizard, you can add the watch tables that you want the Web server to display. When the project is downloaded to the S7-200 SMART CPU, you can view or modify the tag value in watch tables on the Web server.

The following table shows the user memory that you can access for the watch table:

Accessible user memory	Permissions
SM	Read only
V	Read and Write
DB2	Read and Write
I	Read only
Q	Read and Write
M	Read and Write
T (Timer)	Read and Write
T (Timer bit)	Read only
C (Counter)	Read and Write
C (Counter bit)	Read only
HC	Read only
AI	Read only
AQ	Read and Write

Create watch tables

To configure watch tables, follow these steps:

1. Open the Web server wizard.
2. Click "Watch Table" in the navigation panel.
3. Press the "Add" button to add a watch table. You can add up to three watch tables.
4. Enter names and comments for the watch tables.

   Note

   The rule for naming watch tables is:
   - The length of the name should be less than 254 characters, or 127 Chinese words.
   - The watch table name cannot contain special characters.

   Note

   Watch table comment should be within 256 bytes. For a comment which is longer than 256 bytes, STEP 7-Micro/WIN SMART V3 truncates it without reporting any errors.

   Watch table comments are not downloaded to the CPU.

5. Click "Next" to configure the added watch table.

Edit tags in the watch table

To edit watch tables, follow these steps:

1. Enter the tag name, absolute address and display format for the tags you want to monitor or modify.

   Note

   The rule for naming tags is:
   - The length of the name should be less than 256 characters, or 128 Chinese words.
   - The tag name cannot contain special characters.

2. You can edit the watch table with the following buttons:
   - Select: select the watching items
   - Add: add a new row
   - Delete: delete the selected row
   - Move up: move the selected row up
   - Move down: move the selected row down
3. You can select a display format from the drop-down list in the "format" column if you want to change this default setting.

NOTICE
Danger due to modifying an identical operand in parallel with different modify values in more than one watch table When working with more than one watch table, avoid modifying identical operands permanently multiple times with different modify values. If an identical operand is modified permanently with different modify values at the same time in different watch tables, all watch tables will display the last modified value, because the modify value assigned last will be used in this case.

Complete the configuration

1. Repeat the above procedures to configure all the watch tables you added.
2. Click "Next" to continue the Web server configuration or click "Generate" to save and complete the configuration.

After you download the configuration to the S7-200 SMART CPU, a user can access the watch tables that they have privileges to view.

9.2 Configuring the HTTPS certificates for the Web server with certificate management

Before you can use the S7-200 SMART Web server, you need to configure the certificates in the certificate management wizard in STEP-7 Micro/WIN SMART V3 first.

STEP 7-Micro/WIN SMART V3 provides two modes for downloading the certificate. In "External certificate" mode, the device certificate is signed by certificate authority provided by the user. In "internal certificate" mode, the device certificate is signed by the certificate authority (CA) generated by the CPU.

Note

Before configuring the Web server certificate, please update the PLC time to current actual time in "Set Clock", otherwise the validity period of the certificate may not be as expected.

Mode	Key / Algorithm	Algorithm
External certificate	CA key recommended	RSA2048, ECC P256
	CA signature algorithm recommended	SHA256, SHA384, SHA512
	Device certificate key	ECC P256
	Device certificate algorithm	SHA256, SHA512

Mode	Key / Algorithm	Algorithm
Internal certificate	CA key	ECC P256
	CA signature algorithm	SHA256
	Device certificate key	ECC P256
	Device certificate algorithm	SHA256

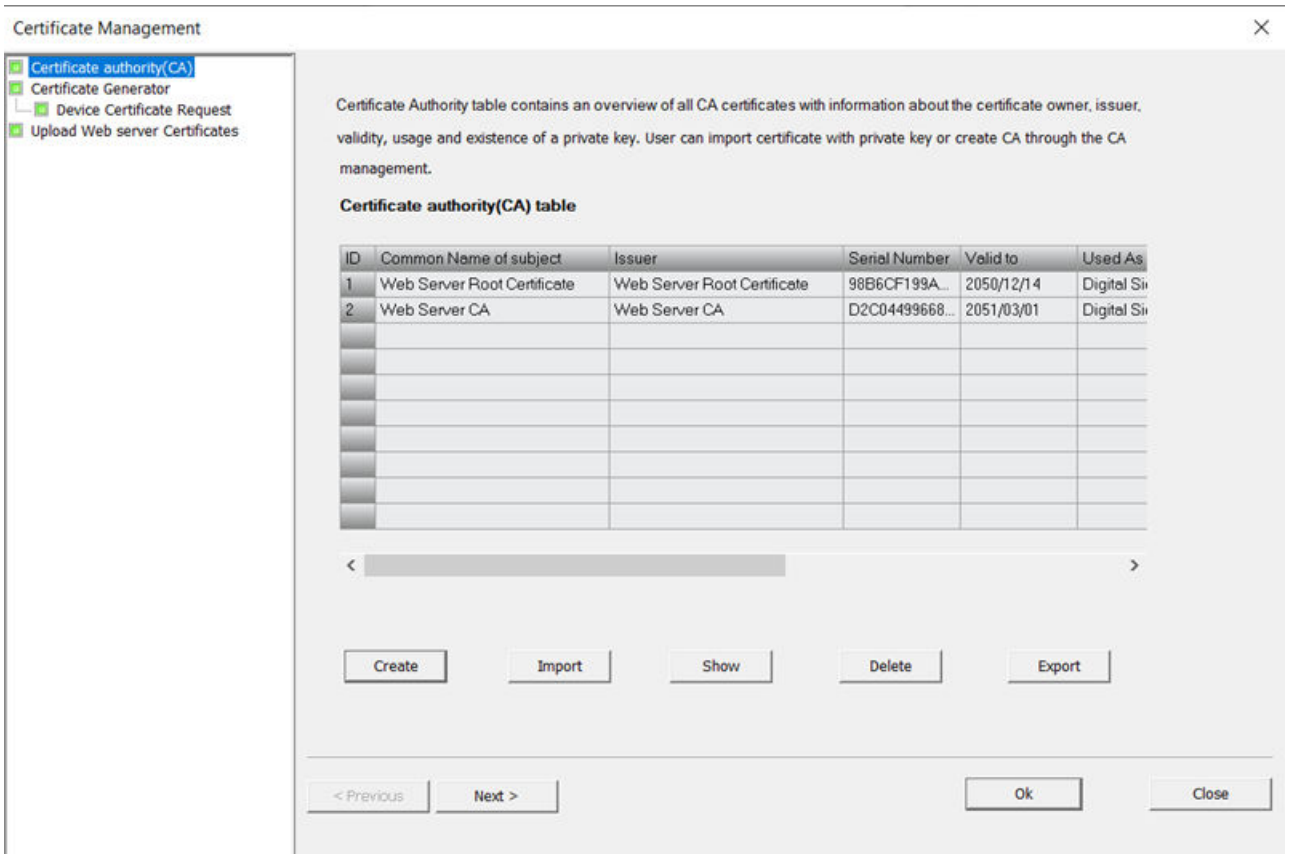
9.2.1 External certificate

If you use "external certificate" mode to download the certificates, you must create or import a certificate authority with the certificate management tool.

9.2.1.1 Manage the certificate authority

Certificate authority (CA) can assign a device certificate. The CA table contains all the CA created or imported to STEP 7-Micro/WIN SMART V3. You can create, import, show, delete or export in the CA table. The CA table can contain up to 10 items.

Create a certificate authority



1. Open the certificate management wizard.
2. Click the "Create" button below the CA table.
   The "Create Certificate Authority" window opens.
3. Enter the parameters for the new certificate authority.

Parameters	Description	Mandatory or not	Default value
Common name of subject	Device for which the certificate is valid. It can contain simple Chinese and traditional Chinese and should be within 128 UTF byte.	Yes	
Organization name of the subject	Organization the device is in. It can contain simple Chinese and traditional Chinese and should be within 128 UTF byte.	No	
Organization unit name of the subject	Unit name of the organization the device is in. It can contain simple Chinese and traditional Chinese and should be within 128 UTF byte.	No	
Locality of subject	Location of the device. It can contain simple Chinese and traditional Chinese and should be within 128 UTF byte.	No	
Country/Region	Country/Region the device is in. It can contain two ASCII bytes.	No	
Signature	The signature algorithm	Yes	sha256ECDSA
Valid from	The certificate is valid from this time	Yes	The present time
Valid until	The certificate is invalid from this time.	Yes	30 years from the present time

4. Add a Subject Alternative Name (Optional for CA).
   – Click the "Add" button in the subject alternative name table.
   – Select the type in the type drop-down list and enter its value.
   – You can delete a Subject Alternative Name by clicking the "Delete" button.
5. Enter the password in the Authentication window.

   Note

   A strong password provides greater security. Strong passwords must contain at least 12 characters, at least one lowercase letter, one uppercase letter, one number, and one special character. Keep the password secret and change it frequently.

6. Click "OK".

Import a certificate authority

1. Open the certificate management wizard.
2. Click the "Import" button below the CA table.
3. Navigate to the folder in which you saved the CA and select it.
4. Enter the password in the Authentication window.

Note

You can only import a p12 CA to STEP7-Micro/WIN SMART, and the size of the imported CA cannot exceed 4 KB.

Delete a certificate authority

1. Open the certificate management wizard.
2. Select the CA you want to delete and click the "Delete" button.

Show a certificate authority

1. Open the certificate management wizard.
2. Select the CA you want to check and click the "Show" button.

Export a certificate authority

1. Open the certificate management wizard.
2. Select the CA you want to export and click the "Export" button.
3. You can select the type for the CA in the "Save as type" drop-down list.

   Note

   You can export the CA as the following types: der, cer, crt, pem and p12.

4. Navigate to the folder in which you want to save the CA and click "Save".

9.2.1.2 Downloading the certificate by external mode

In "External certificate" mode, certificate management downloads device certificate signed by certificate authority you provide.

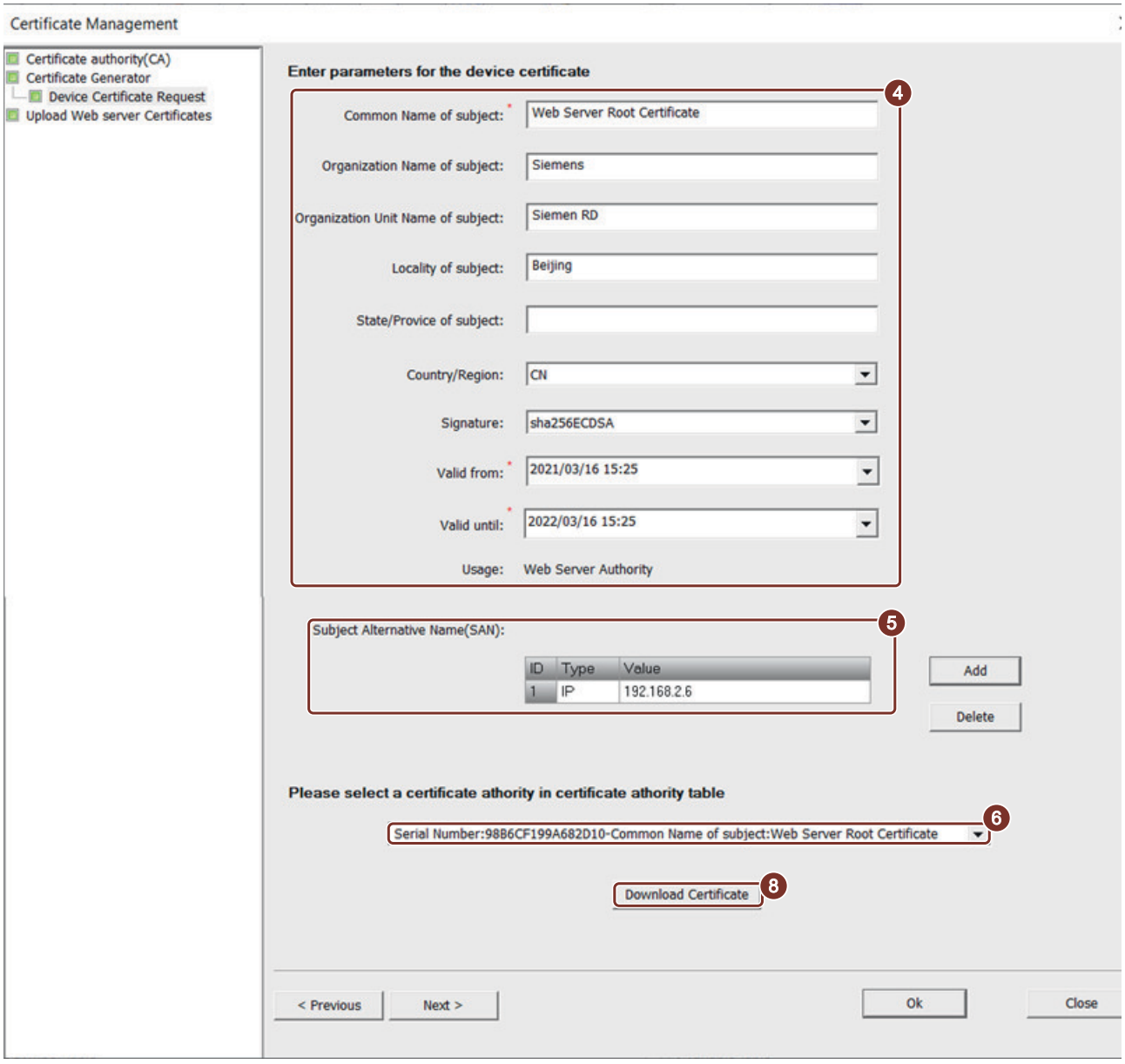
Procedure

To download the certificate by external mode, follow these steps:

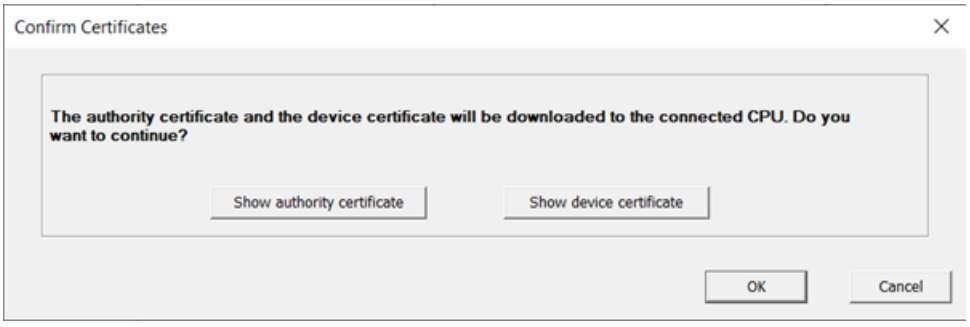
1. Set up the communication between STEP 7-Micro/WIN SMART V3 and the CPU.
2. Create or import a certificate authority in the CA table.

3. Select "External Certificate" as the generation mode.
4. Enter the parameters for the device certificate.
5. Set the subject alternative name.
   You can add IP, email, DNS and URI in SAN table. CPU IP is mandatory to fill.
6. Select a certificate authority in the CA table.
7. Enter the password in the Authentication window.

8. Click "Download Certificate".
   CA and device certificate are downloaded to the CPU.



9. Click "OK" in the confirm page to proceed or "Cancel" to cancel the download.



9.2.2 Downloading the certificates by Internal certificate

In "Internal certificate" mode, certificate management grants the CPU to generate the CA. The device certificate is signed by the internal CA.

Note

If the CPU already has internal CA, re-downloading internal certificate or transfer certificate through an SD card (Page 131)will only generate a new device certificate, while the certificate of the internal CA remains intact.

Procedure

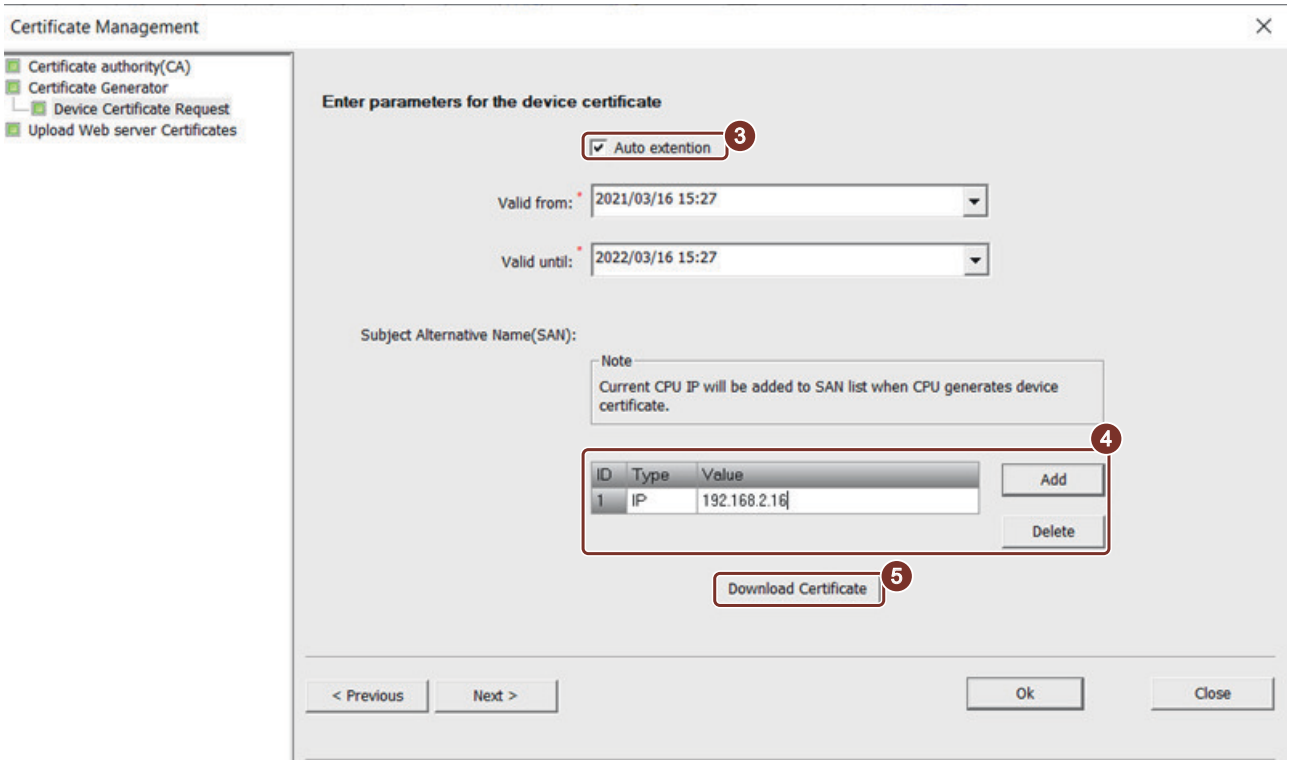
To download the certificate by internal mode, follow these steps:

1. Set up the communication between STEP 7-Micro/WIN SMART V3 and the CPU.
2. Select "Internal Certificate" as the generation mode.
3. If you want the CPU to extend the device certificate automatically when it expires, select the check box before "Auto extension".

   Note

   If the "Auto extension" is enabled, when the device certificate expires or will expire in next seven days, the CPU extends the device certificate automatically after power cycle.

4. Set the subject alternative name. (Optional for internal certificate)
5. Click "Download Certificate".
   The certificate grants the CPU to generate CA and assign device certificate.



9.3 Downloading and installing the certificate of CA

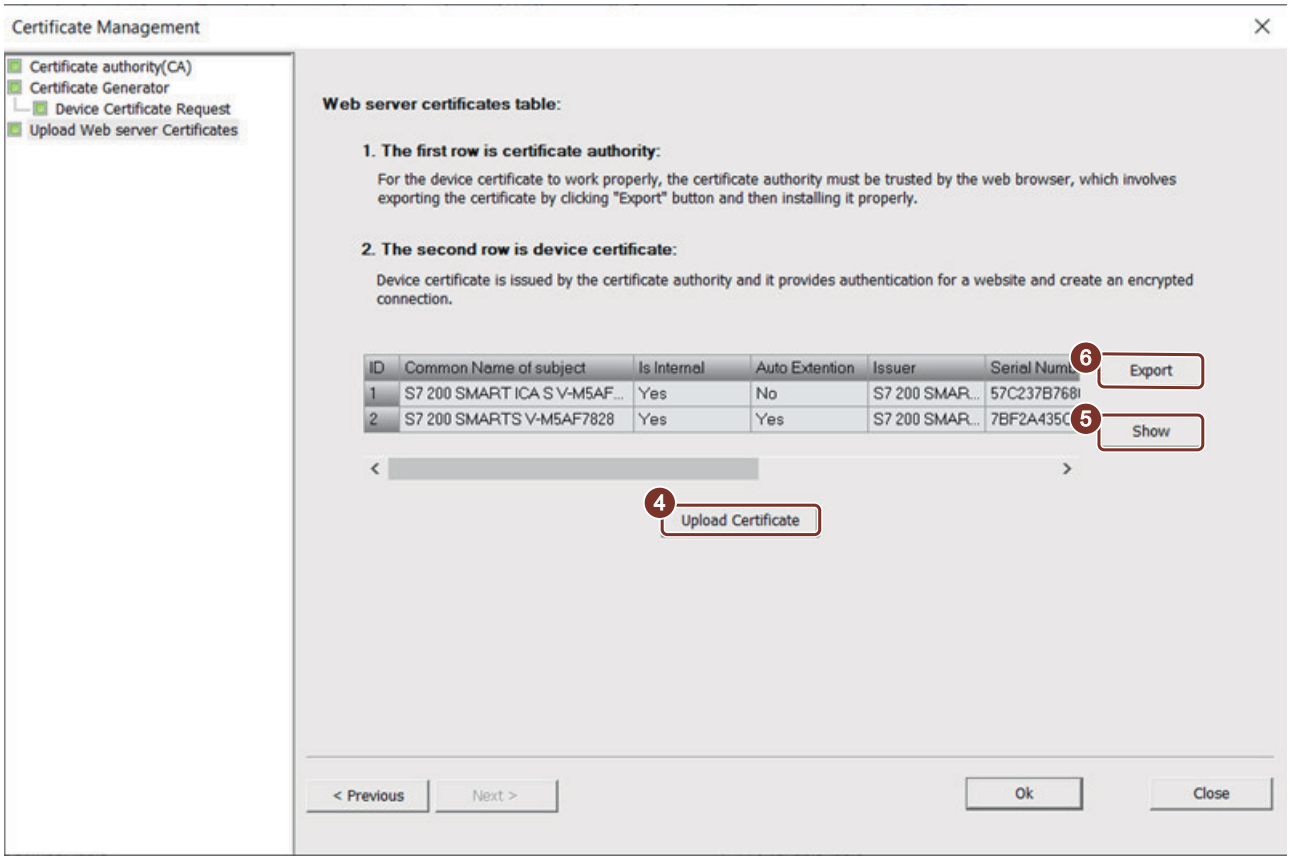
For visiting the Web server from a browser, you need to install the certificate of certificate authority (CA) in your Internet options.

9.3.1 Get the certificate of CA

You can get the certificate of certificate authority as follows:

1. Set up the Ethernet communication with the CPU.
2. Open the Certificate management wizard.
3. Select "Upload Web server certificates" in the navigation pane.
4. Click "Upload Certificate".
   The certificates on the CPU are uploaded in the Web server certificates table: The certificate in the first row is the CA.

5. To show a certificate, select it in the table and click "Show".
6. To export a certificate, select it in the table, click "Export", navigate to the target folder, and click "Save".

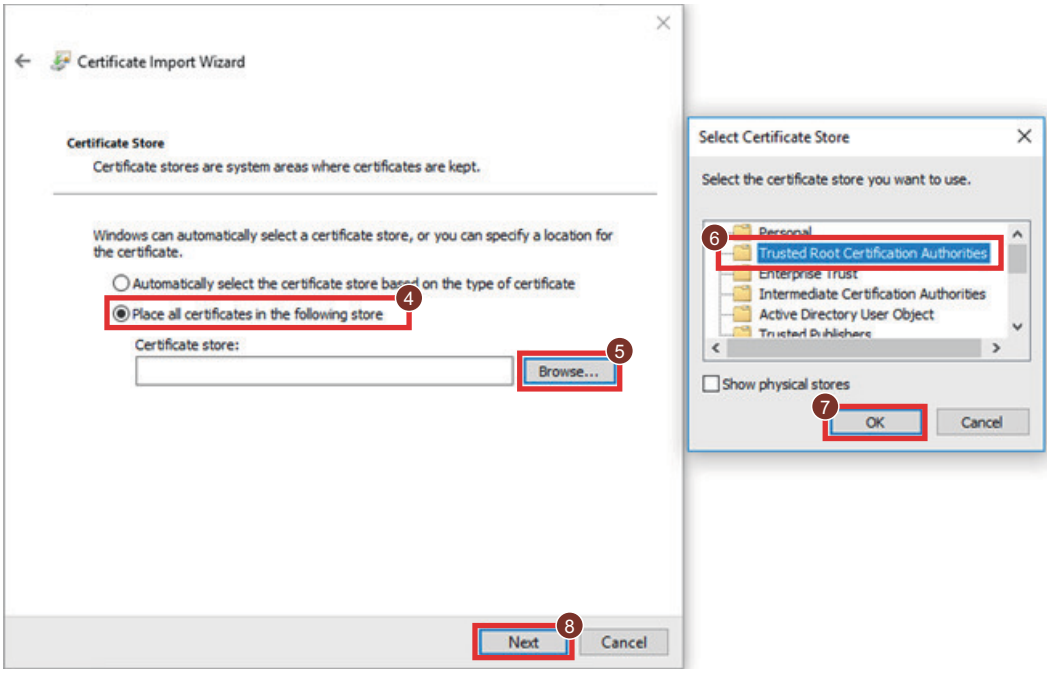


9.3.2 Installing the certificate for Windows

Install the certificate for Windows

1. Double click the certificate to start the installation.
2. Check the certification information in the pop-up window, then click "Install Certificate" to continue.
3. On the certificate import Wizard welcome page, select the "Store location", then click "Next".
   - If you select “Current User”, the certificate is only valid for the current user.
   - If you select "Local Machine", the certificate is valid for all users on this PC. Only the administrator can install the certificate as "Local Machine".
4. Select the check box of ④ and click ⑤ to store the certificate.
5. Trust the certificate by selecting ⑥ in the pop-up window and then clicking ⑦.

6. Click ⑧ to continue.



7. In the "Certificate import wizard" window, click "Finish" to confirm your selection.
8. In the security warning window, click "Yes" to confirm the installation.

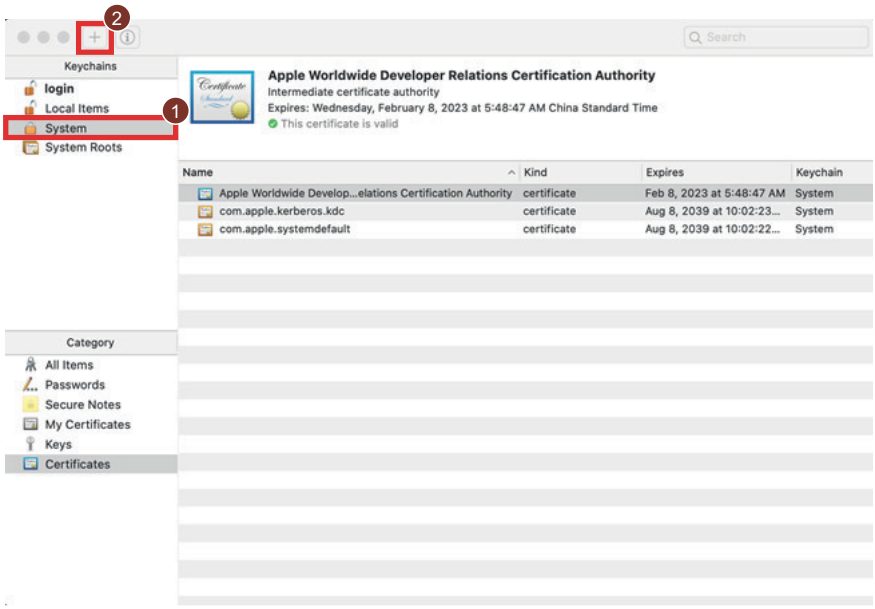
Import the certificate to Firefox

If Firefox still cannot trust the certificate after you have installed the certificate, follow the instruction from Firefox to import the certificate to Firefox under the guidance of the system administrator.

9.3.3 Installing the certificate for Mac OS

Install the certificate for MAC OS

1. Open the Keychain.
2. To add a certificate, select "system" ① , then click "+"②.



3. Select the certificate of CA, then click "Open" to add it.
4. Enter the password, then Click "Modify Keychain".
5. Double-click the certificate of CA to open it.
6. Trust the certificate by clicking "Trust" and setting "Secure Sockets Layer (SSL)" as "Always trust".
7. Enter the password and confirm the modification.

9.3.4 Installing the certificate for mobile devices

Follow the requirements of your operating system and browser to install the certificate under the guidance of the system administrator.

9.4 Accessing Web server from PC or mobile device

You can access the S7-200 SMART Web server from a PC or from a mobile device through the IP address of the S7-200 SMART CPU.

To access the S7-200 SMART Web server from a PC or mobile device, follow these steps:

1. Ensure that the S7-200 SMART and the PC or mobile device are on a common Ethernet network or are connected directly to each other with a standard Ethernet cable.
2. Open a Web browser and enter the URL "https://ww.xx.yy.zz", where "ww.xx.yy.zz" corresponds to the IP address of the S7-200 SMART CPU.

The Web browser opens the Introduction Web page.

Accessing web pages through a URL

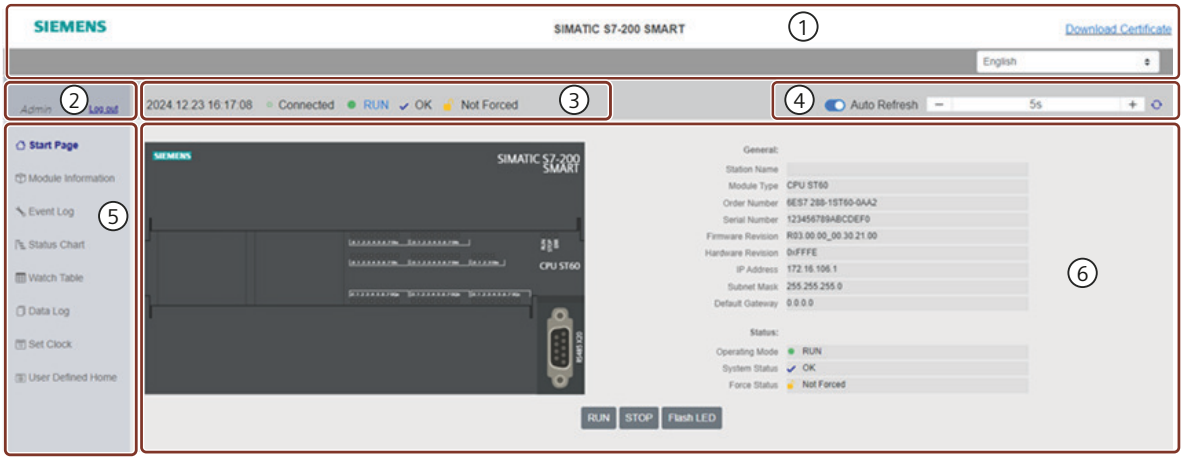
You can access web pages by entering the page URLs. The following table lists the corresponding URL of each page:

Web page	URL
Standard Web pages	
Introduction page	https://PLCIP
Start page	https://PLCIP/start
Module Information page	https://PLCIP/module
Event Log page	https://PLCIP/eventlog
Status Chart page	https://PLCIP/statuschart
Watch Table page	https://PLCIP/watchtable
Data Log page	https://PLCIP/datalog
Set Clock page	https://PLCIP/setclock
User defined Web pages	
Login page	https://PLCIP/user_define/login.html
Home page	https://PLCIP/user_define/

9.5 Standard Web pages

9.5.1 Layout of web pages

Regardless of whether you are viewing the page on a PC or on a mobile device, each page of the S7-200 SMART Web server has the same content area, but the layout and navigation controls vary based on the screen size and resolution of the device. On a PC or large mobile device, the layout of a Web page appears as follows:



① Header: Product name, a link for download certificate and selector for display language.

② Login/Logout

③ PLC status: The time, CPU operating mode, system status, force status and connection status are displayed.

Change the operation mode: you can change the operation mode by clicking "RUN" or "STOP".

④ Refresh button

- Auto refresh: If you select turn the "Auto Refresh" button to on, Web pages will refresh by default every five seconds. You can also set a custom refresh time period in seconds.
- manual refresh: Click to refresh the Web page.

⑤ Web page tree

⑥ Detailed information of a specific web page. You can use the following buttons to control the CPU

- RUN: Set the CPU to run mode.
- STOP: Set the CPU to stop mode.
- Start LED Flashes: Flash the LED light on the CPU.

Note

User defined home tab

The user defined home is the entry point to the home page of user defined Web pages.

9.5.2 Introduction

When you enter the URL of a specific Web page and press enter, the introduction page displays. The Introduction page is the welcome screen for entering into the S7-200 SMART Web pages. From this page, click "Enter" to access the S7-200 SMART Web pages.

To log in, follow these steps:

1. Enter the user name for the User Name field.
2. Enter the user password in the Password field.
3. Click "Log in".

Note

If you "disabled PLC access control" or "enabled PLC protection" in the control configuration, you can log in to the Web server using the default Admin account and its initial password.

If you enabled the user managment, you can log in with all users and their respective passwords you defined.

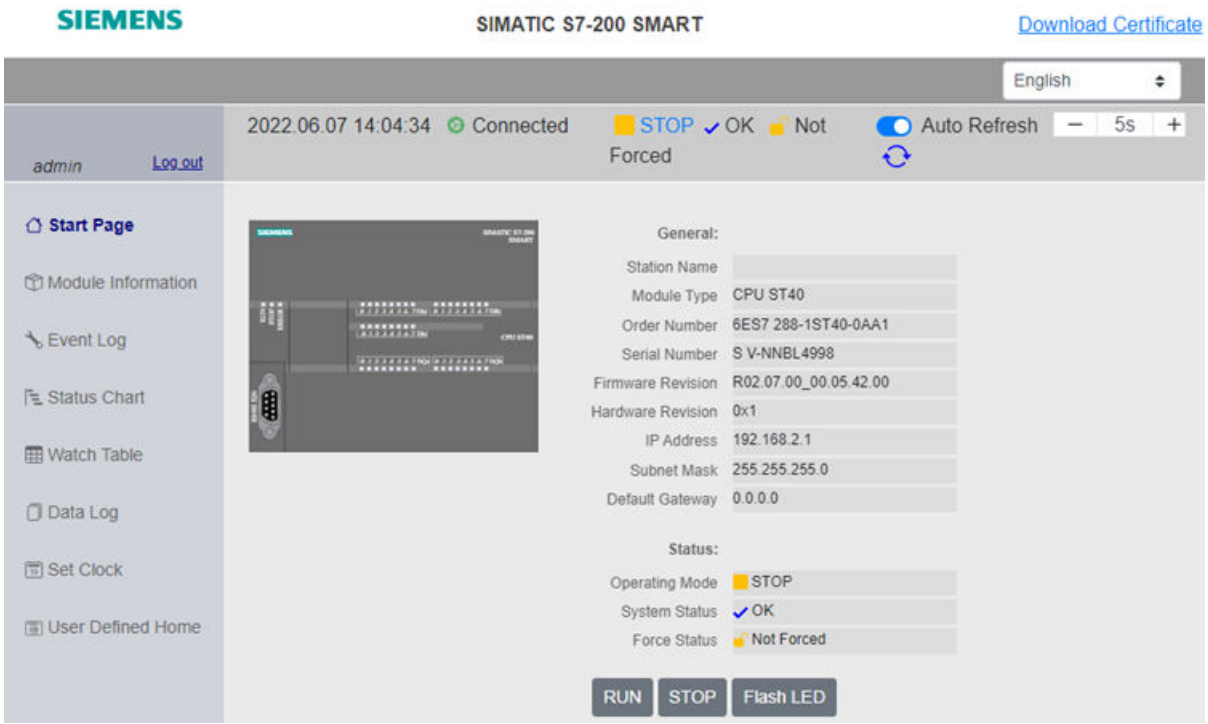
Your login session expires after thirty minutes of inactivity. If you switch on the Auto-refresh, the login session timeout resets, preventing the session from expiring.

9.5.3 Start page

The Start page displays the general information and status of the connected CPU.

You can control the CPU by clicking the following button:

- RUN: place the CPU in RUN mode
- STOP: place the CPU in STOP mode
- Flash LED: flashes the STOP, RUN, and FAULT lights



9.5.4 Module information

The Module Information page displays the CPU, extension module and PROFINET device information.

You have the following options for checking the module information:

- Expand All: display detailed information for each module.
- Collapse All: hide detailed information for each module.
- Show Details: display detailed information for one specific module.
- Hide Details: hide detailed information for one specific module.

9.5.5 Event log

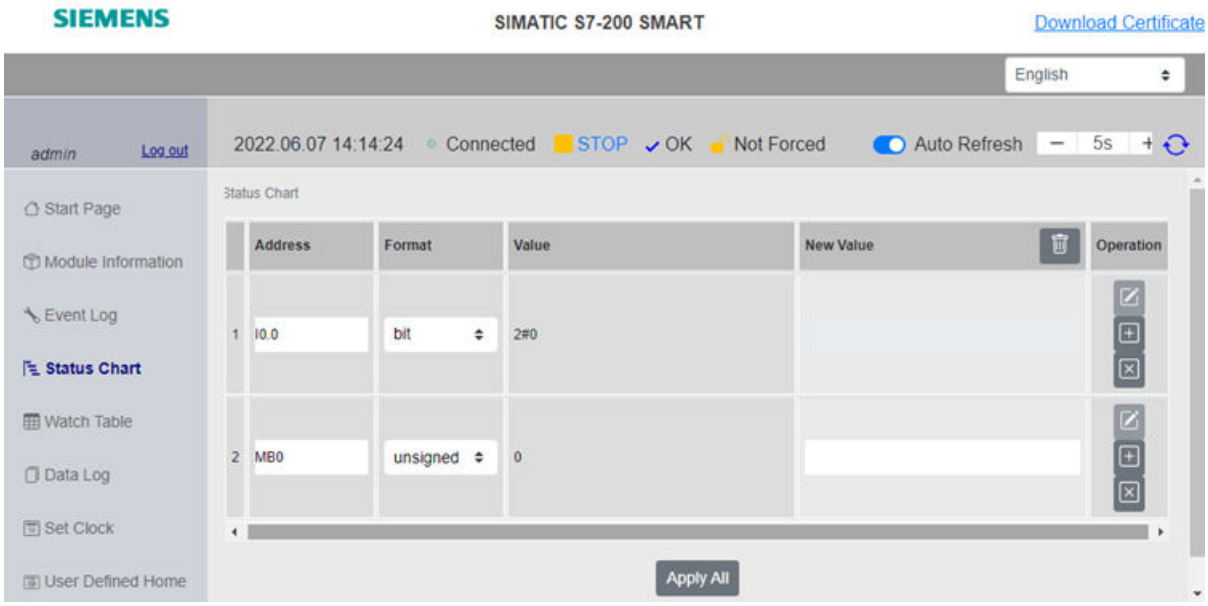
The Event Log page displays a table of time-stamped events. The table lists the time, event type, and error or reason information.

The maximum number of the possible event log entries is 32.

You can click "Event log" or the icon to download the event log in a Microsoft Excel spreadsheet.

9.5.6 Status chart

The Status Chart page allows you to monitor the PLC data values. You can assign a memory address, format and new value, as well as apply the changes to the CPU. Once you bookmark a status chart page, you can quickly access the same chart next time.



Note

- The maximum number of new addresses is 30.
- "Status chart read only" privilege is required for viewing the Status Chart page.
- "Status chart read and write" privilege is required for viewing and modifying the data value.

Note

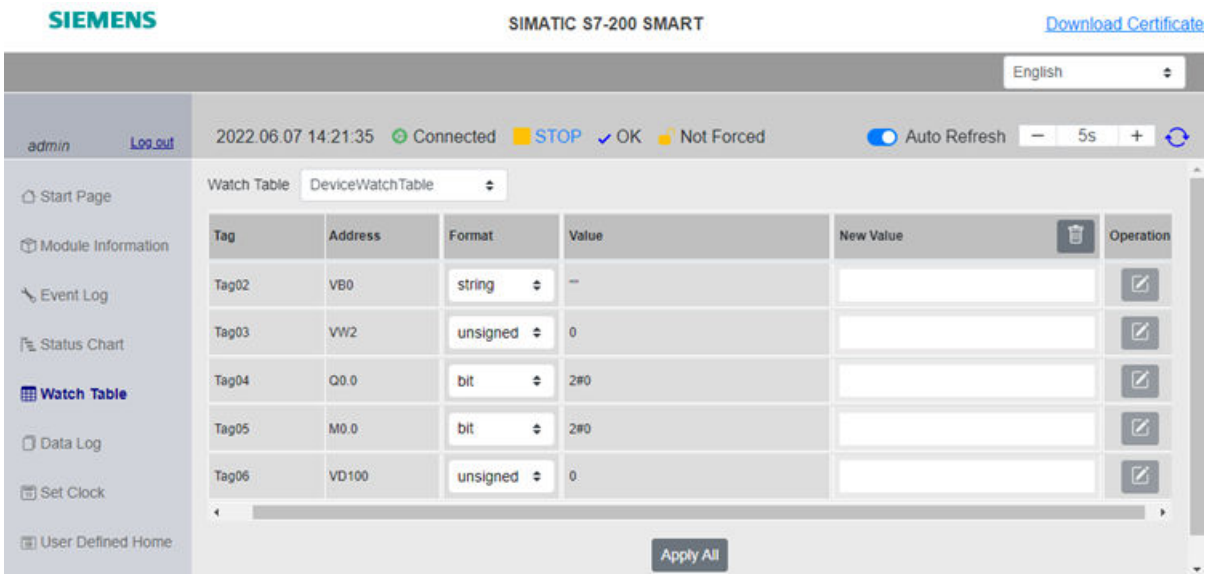
- The following memory areas cannot be written in: I, AI, SM, HC, T bit and C bit.
- If the CPU is in stop mode, applying changes is not supported in the Status Chart page.
- If more than one item in the Status Chart is changed, and there is an error code for one item, applying all changes cannot be successful.

9.5.7 Watch table

The Watch Table page allows you to check and modify the IO data.

Note

You must configure the watch table in the Web server wizard.



You can select a watch table from the drop-down list, and change the format or assign a new value for a tag.

You can clear a new value as needed or click the "Apply All" button to save all changes.

Note

- The following memory area cannot be written in: I, AI, SM, HC, T bit and C bit.
- If PLC is in stop mode, applying changes is not supported in the Watch Table page.
- If more than one item in Watch Table is changed, and there is an error code for one item, applying all changes cannot be successful.

9.5.8 Data log

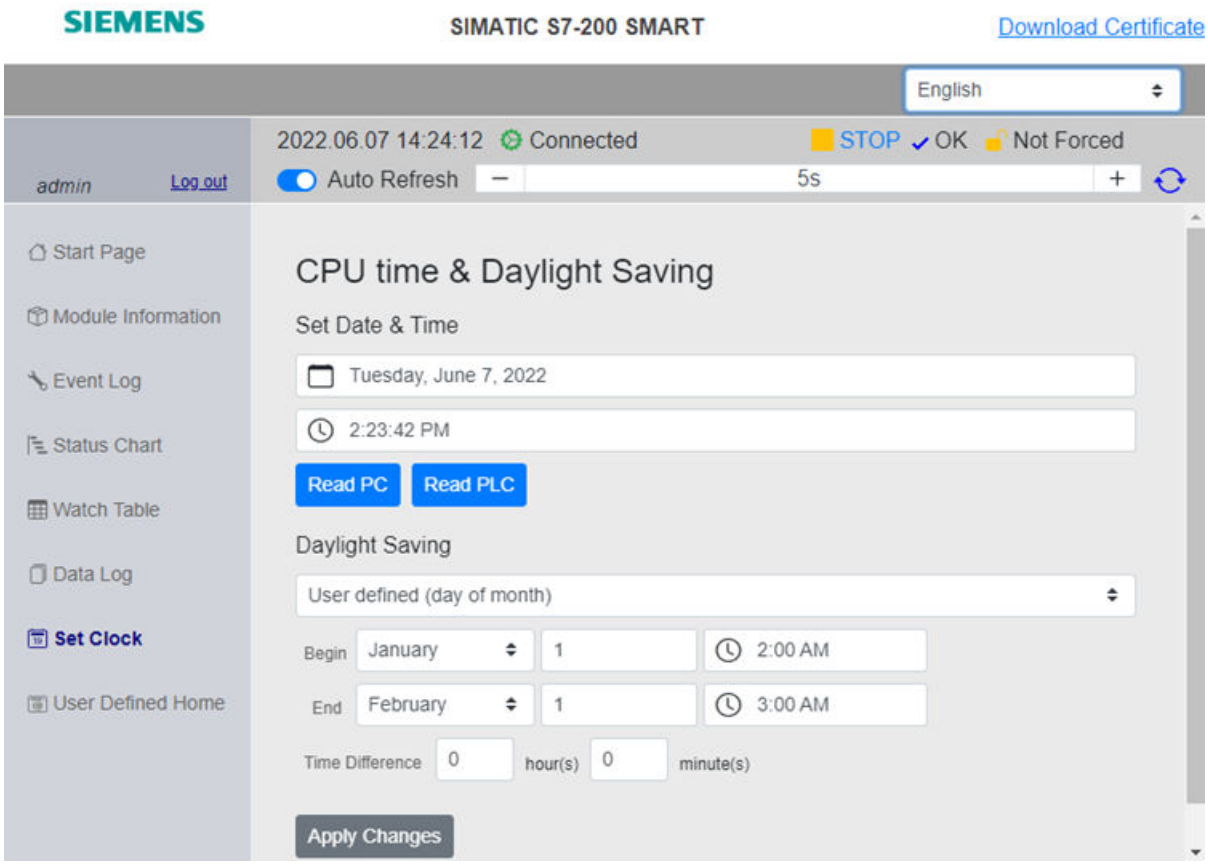
The Data Log page allows you to get access to data log files of the connected CPU.

Your selection for data log upload options in data log wizard is consistent with that of Data Log pages. For example, If you select the checkbox "Clear all records from the data log on upload" for upload options in the data log wizard, the Data Log page displays "Yes".

You can click the Data Log name to download the data log file in CVS format.

9.5.9 Set clock

The Set Clock page allows you to set the time and date and configure daylight saving time.



9.6 User defined Web pages

The S7-200 SMART supports user to custimize your own Web pages and connect to the CPU to get CPU runtime information. This process involves below steps:

1. Create use-defined Web page resource by SMART Web Editor or any other HTML editor tools. Refer to *SMART Web Editor Online Help* for detailed information about how to use this tool and how to create user defined web page resource.

   Note

   Bin file size.

   The size of bin file generated from the user defined web page resource cannot exceed 2 M, otherwise, the resource cannot be downloaded succesfully to the CPU.

2. Download user-defined Web resource files to the connected CPU (Page 237).
3. Configure Web server with Web server wizard (Page 215).

4. Configure the HTTPS certificates for the Web server with certificate management (Page 218).
5. Visit user-defined Web pages (Page 228).
   You can access user defined web pages through URL (Page 228), or use User defined Home tab as the entry point on the Standard web page.

WARNING

Unauthorized access to the CPU through user defined Web pages

Insecure coding of user-defined Web pages introduces security vulnerabilites such ascross-site scripting (XSS), code injection, and others.

Protect your S7-200 SMART CPU from unauthorized access by installing it in a secure fashion as outlined in the Operational Guidelines found on the Industrial Security Web site (http://www.siemens.com/industrialsecurity).

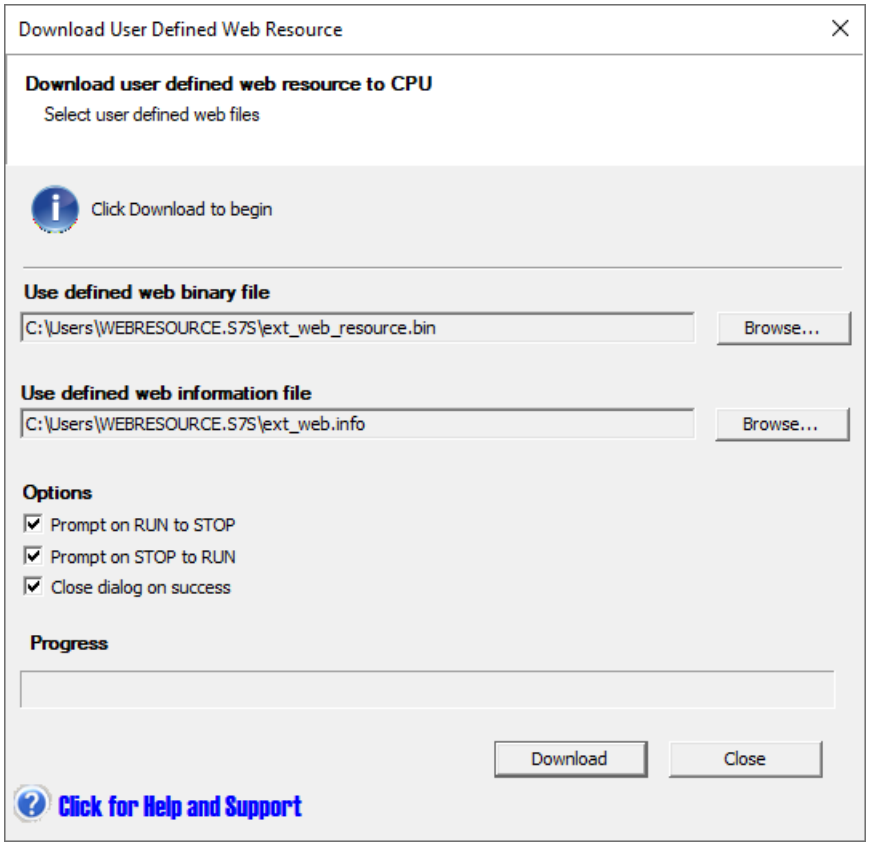
9.6.1 Downloading user-defined Web resource files

There are two methods to download the user-defined Web recource files to the connected CPU.

- Download web resource files to the connected CPU though Step 7-Micro/WIN SMART V3
- Transfer web resource files to the connected CPU through a microSDHC card

Downloading through Step 7-Micro/WIN SMART V3

1. Click the **Download** drop-down button in the **Transfer** section of the **File** or **PLC** menu, then select **User Defined Web**. The "Download User Defined Web Resource" window appears.
2. Click the **Browse** button to select web resource binary file and information file separately in the pop-up window.



3. Click the **Download** button to start the downloading process.

Transfering through microSDHC card

Put web resource files into the memory card at the root level, and the card is organized as follows:

- A folder named as WEBRESOURCE.S7S, containing web resource files: ext_web_resource.bin; ext_web.info
- A text file named as S7_JOB.S7S, containing words EXT_WEB

9.7 Web API (Application Programming Interface)

The CPU offers you a Web-based API (Web API) as an interface for reading and writing CPU data. The Web API enables you to:

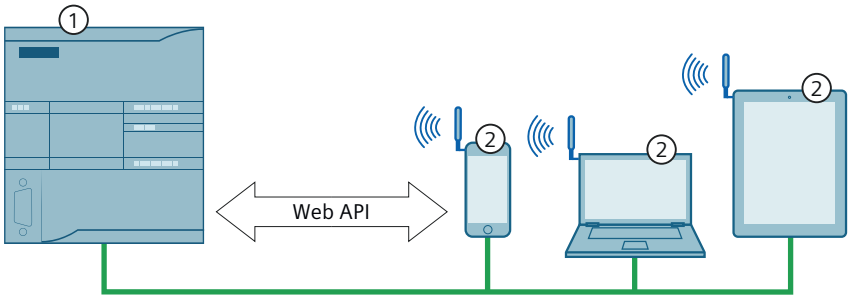
- implement web applications at the latest state-of-the-art technology.
- communicate with the Web server of the CPU via script and programming languages.
- create web applications that connect to multiple CPUs at the same time, for example, to create dashboards that visualize the status of multiple CPUs.

Note

The Web API can only be used for CPUs as of firmware version V2.7.

Web API connection

The following graphic shows an example of the Web API between CPU and terminal device.



① CPU
② Terminal device

CPU supports Web API through PROFINET port.

Web API function

The Web API provides you to remotely access CPU data. After configuring user permission, you can read or write the connected CPU data.

Web API methods

The Web API supports below methods. Refer to "*S7-200 SMART Web API development manual*" for detailed information about these instructions and programming templates.

Instruction	Description
login	log in CPU
logout	log out CPU
get_permission	get a list of authorised actions
read	read the process data of CPU

Instruction	Description
write	write the process data to CPU
browse	browse Web API list supported by CPU

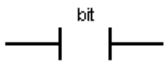
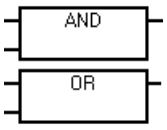
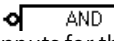
Program instructions

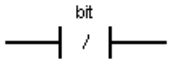
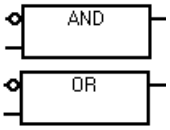
10

10.1 Bit logic

10.1.1 Standard inputs

LAD	FBD	STL	Description

		`LD bit`
		`A bit`
		`O bit`
			Test a bit value in memory (M, SM, T, C, V, S, L) or process image register (I or Q).
			LAD: Normally open and normally closed switches are represented by a contact symbol. If power flow is present on the left-side and the contact is closed, then power flows through the contact to the right-side connector and to the next connected element.
			• The Normally Open (N.O.) LAD contact is closed (ON) when the bit is equal to 1.
			• The Normally Closed (N.C.) LAD contact is closed (ON) when the bit is equal to 0.
			
			FBD: Normally open instructions are represented by AND/OR boxes. Box instructions can be used to evaluate Boolean signals in the same manner as ladder contact networks. Normally closed instructions are also represented by boxes. A normally closed instruction is created by placing the negation circle  , on a binary input signal connector. The number of inputs for the AND/OR boxes can be expanded to a maximum of 31 inputs.
			
			STL: The normally open contact is represented by the LD (load), AN (AND), and ON (OR) instructions. These instructions load, AND, or OR the value of the addressed bit with the top bit of the logic stack. The normally closed contact is represented by the LDN (Load NOT), AN (AND NOT), and ON (OR NOT) instructions. These instructions load, AND, or OR the logical NOT of the addressed bit value with the top bit of the logic stack.

		`LDN bit`
		`AN bit`
		`ON bit`	

Input / output	Data type	Operand
bit (LAD, STL)	BOOL	I, Q, V, M, SM, S, T, C, L,
Input (FBD)	BOOL	I, Q, V, M, SM, S, T, C, L, Logic flow
Output (FBD)	BOOL	I, Q, V, M, SM, S, T, C, L, Logic flow

FBD AND/OR input assignment

The editor feature described in the following table is active only if an input stub is selected and colored red, inside the FBD box cursor.

Input option	Place cursor	Tool button	Shortcut key
Add input	On box		+
Remove input	On box and bottom input		-

See also

Bit logic input examples (Page 253)

Logic stack overview (Page 243)

10.1.2 Immediate inputs

LAD	FBD	STL	Description
bit I	AND OR	LDI bit AI bit OI bit	The immediate instruction obtains the physical input value when the instruction is executed, but the process image register is not updated. An immediate contact does not wait on the PLC scan cycle to update; it updates immediately. The Normally Open Immediate contact is closed (ON) when the physical input point (bit) state is 1. The Normally Closed Immediate contact is closed (ON) when the physical input point (bit) state is 0. **LAD**: Normally open and normally closed immediate instructions are represented by contacts. **FBD**: Normally open immediate instructions are represented by the vertical immediate indicator AND in front of an input connection. The Normally closed immediate instruction is represented by the immediate indicator and negation circle AND in front of an input connection. The immediate indicator cannot be used when a logic flow connection is used instead of a physical input (I) bit address. FBD box instructions can be used to evaluate physical signals in the same manner as ladder contacts. The number of inputs for the AND/OR boxes can be expanded to a maximum of 31 inputs. **STL**: The Normally Open Immediate contact is represented by the LDI (Load Immediate), AI (AND Immediate), and OI (OR Immediate) instructions. These instructions load, AND, or OR the physical input value with the top of the logic stack. A normally Closed Immediate contact is represented by the LDNI (Load NOT Immediate), ANI (AND NOT Immediate), and ONI (OR NOT Immediate) instructions. These instructions immediately load, AND, or OR the logical NOT of the value of the physical input value with the top of the logic stack.
bit /I	AND OR	LDNI bit ANI bit ONI bit	

Input / output	Data type	Operand
bit (LAD, STL)	BOOL	I
Input (FBD)	BOOL	I

FBD editor input assignment

The editor feature described in the following table is active only if an input stub is selected and colored red, inside the FBD box cursor.

Input option	Place cursor	Tool button	Shortcut key
Add input	On box		+
Remove input	On box and bottom input		-
Toggle negate input	On box and input		F11
Toggle immediate input	On box and input		CTRL F11

See also

Bit logic input examples (Page 253)

Logic stack overview (Page 243)

10.1.3 Logic stack overview

The STEP 7-Micro/WIN SMART V3 program compiler uses the logic stack to transform the graphical I/O networks of LAD and FBD programs into STL (statement list) programs. The resultant STL program is logically the same as the original LAD or FBD graphical network and can be executed as a program list. All successfully compiled LAD and FBD programs have generated the underlying STL program and can be viewed as LAD, FBD, or STL.

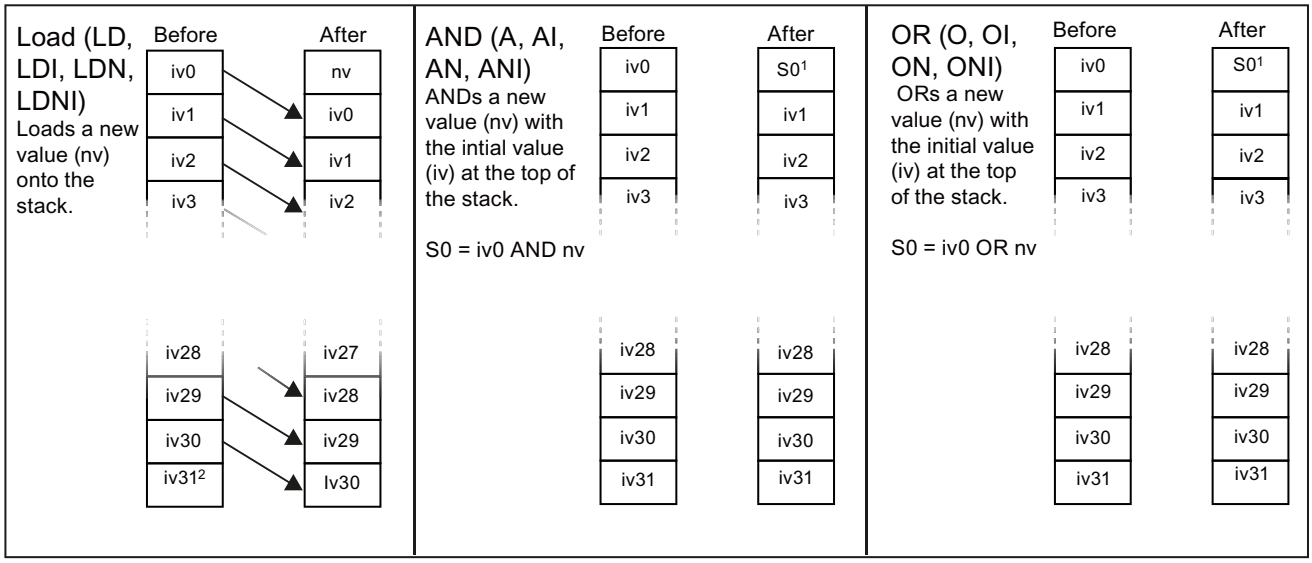
For LAD and FBD editing, the STL logic stack instructions are automatically generated and the programmer does not need to use the logic stack instructions.

You can also create STL programs directly with the STL editor. An STL programmer uses the logic stack instructions directly. Combination logic can be created in the STL editor that is too complex to be viewed in the LAD or FBD editor, but may be necessary for special applications.

All successfully compiled LAD and FBD programs can be viewed in STL, but not all successfully compiled STL programs can be viewed in LAD or FBD.

Input networks and the logic stack

As shown in the following figure, the CPU uses a logic stack to combine the logic states of STL inputs. In these examples, "iv0" to "iv31" identify the initial values of the logic stack levels, "nv" identifies a new value provided by the instruction, and "S0" identifies the calculated value that is stored in the logic stack.



[1] S0 identifies the calculated value that is stored in the logic stack.
[2] After the execution of a Load, the value iv31 is lost.

Output networks and the logic stack

ENO is a binary output for boxes in LAD and FBD. If a LAD box has power flow at the EN input and is executed without error, the ENO output passes power flow to the next LAD element. You can use the ENO as an enable bit that indicates the successful completion of an instruction. The ENO bit is used with the top of stack to affect power flow for execution of subsequent instructions. STL instructions do not have an EN input. The top of the stack must have a value of logic 1 for conditional instructions to be executed. In STL there is no ENO output. However, the STL instructions that correspond to LAD and FBD instructions with ENO outputs set a special ENO bit. This bit is accessible with the AND ENO (AENO) instruction.

STL	Description
AENO	AENO is used in the STL representation of LAD/FBD box ENO bit. AENO performs a logical AND of the ENO bit with the top of stack for the same effect as the ENO bit of a LAD/FBD box. The result of the AND operation is the new top of stack value.

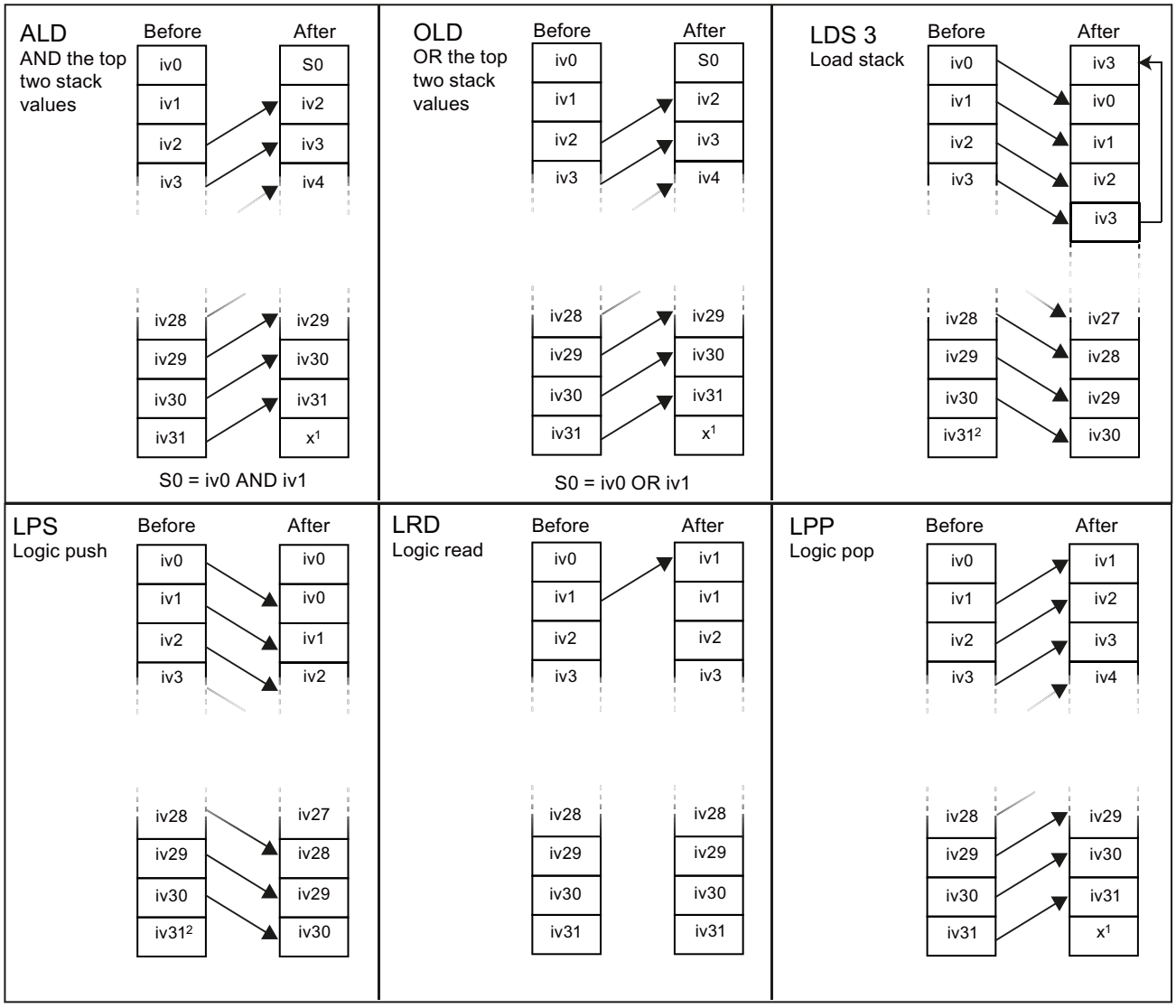
10.1.4 STL logic stack instructions

STL[1]	Description
ALD	The AND Load instruction (ALD) combines the values in the first and second levels of the stack using a logical AND operation. The result is loaded in the top of stack. After the ALD is executed, the stack depth is decreased by one.
OLD	The OR Load instruction (OLD) combines the values in the first and second levels of the stack, using a logical OR operation. The result is loaded in the top of the stack. After the OLD is executed, the stack depth is decreased by one.
LPS	The Logic Push instruction (LPS) duplicates the top value on the stack and pushes this value onto the stack. The bottom of the stack is pushed off and lost.
LRD	The Logic Read instruction (LRD) copies the second stack value to the top of stack. The stack is not pushed or popped, but the old top-of-stack value is destroyed by the copy.
LPP	The Logic Pop instruction (LPP) pops one value off of the stack. The second stack value becomes the new top of stack value.
LDS N	The Load Stack instruction (LDS) duplicates the stack bit (N) on the stack and places this value on top of the stack. The bottom of the stack is pushed off and lost.
AENO	AENO is used in the STL representation of the LAD/FBD box ENO bit. AENO performs a logical AND of the ENO bit with the top of stack for the same effect as the ENO bit of a LAD/FBD box. The result of the AND operation is the new top of stack.

[1] Not applicable for LAD or FBD

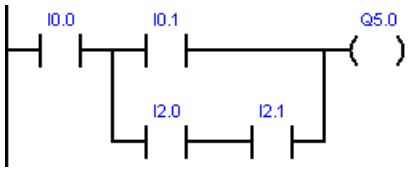
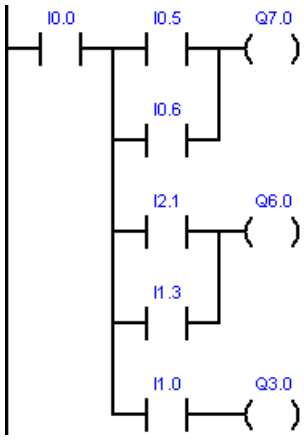
LDS (Load Stack) Input	Data type	Operands
N	BYTE	Constant (0 to 31)

As shown in the following figure, the CPU uses a logic stack to resolve the control logic. In these examples, "iv0" to "iv31" identify the initial values of the logic stack, "nv" identifies a new value provided by the instruction, and "S0" identifies the calculated value that is stored in the logic stack.

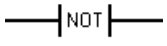
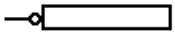


[1] The value is unknown (it could be either a 0 or a 1).
[2] After the execution of a Logic push or a Load stack instruction, value iv31 is lost.

Logic Stack example: LAD networks transformed into STL code

LAD	STL
	**Network 1** **LD I0.0** **LD I0.1** **LD I2.0** **A I2.1** **OLD** **ALD** **= Q5.0**
	**Network 2** **LD I0.0** **LPS** **LD I0.5** **O I0.6** **ALD** **= Q7.0** **LRD** **LD I2.1** **O I1.3** **ALD** **= Q6.0** **LPP** **A I1.0** **= Q3.0**

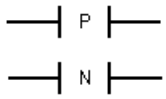
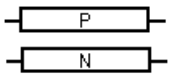
10.1.5 NOT

LAD	FBD	STL	Description
		**NOT**	The Not instruction (NOT) inverts the state of the power flow input. **LAD**: The NOT contact changes the state of power flow input. When power flow reaches the NOT contact, it stops. When power flow does not reach the NOT contact, it supplies power flow. **FBD**: The NOT instruction is represented as a graphical negation (bubble) symbol on Boolean box input connectors and functions as a logic state inverter. **STL**: The NOT instruction changes the value on the top of the stack from 0 to 1, or from 1 to 0.

See also

Bit logic input examples (Page 253)

10.1.6 Positive and negative transition detectors

LAD	FBD	STL	Description
		**EU** **ED**	The positive transition contact instruction (Edge Up) allows power to flow for one scan for each OFF-to-ON transition. The negative transition contact instruction (Edge Down) allows power to flow for one scan for each ON-to-OFF transition. S7-200 SMART CPUs support a combined total (positive and negative) of 1024 edge detector instructions in your program. **LAD**: Positive and negative transition instructions are represented by contacts. **FBD**: The transition instructions are represented by the P and N boxes. **STL**: The positive transition is detected by the EU (Edge Up) instruction. Upon detection of a 0-to-1 transition in the value on the top of the stack, the top of the stack value is set to 1; otherwise, it is set to 0. The negative transition is detected by the ED (Edge Down) instruction. Upon detection of a 1-to-0 transition in the value on the top of the stack, the top of the stack value is set to 1; otherwise, it is set to 0.

Input / output	Data type	Operand
IN (FBD)	BOOL	I, Q, V, M, SM, S, T, C, L, Logic flow
OUT (FBD)	BOOL	I, Q, V, M, SM, S, T, C, L, Logic flow

Note

Because the Positive Transition and Negative Transition instructions require an on-to-off or an off-to-on transition, you cannot detect an edge-up or edge-down transition on the first scan. During the first scan, the CPU saves the initial input state in a memory bit. On subsequent scans, these instructions compare the current state and the state of the memory bit, to detect a transition.

See also

Bit logic input examples (Page 253)

10.1.7 Coils: output and output immediate instructions

LAD	FBD	STL	Description
bit ( )	bit =	`= bit`	The Output instruction writes the new value for the output bit to the process image register. **LAD** and **FBD**: When the output instruction is executed, the S7-200 SMART turns the output bit in the process image register ON or OFF. The assigned bit is set equal to power flow state. **STL**: The value on the top of the stack is copied to the assigned bit.
bit (I)	bit =I	`=I bit`	The Output Immediate instruction writes the new value to both the physical output and the corresponding process image register location when the instruction is executed. **LAD** and **FBD**: When the output immediate instruction is executed, the physical output point (bit) is immediately set equal to power flow. The "I" indicates an immediate reference; the new value is written to both the physical output point and the corresponding process image register address. This differs from the non-immediate references, which only write the new value to the process image register. **STL**: The instruction immediately copies the value on the top of the stack to the assigned physical output bit and process image address.

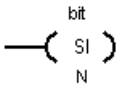
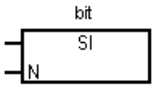
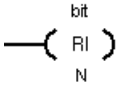
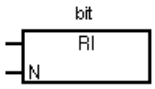
Input / output	Data type	Operand
Bit	BOOL	I, Q, V, M, SM, S, T, C, L
Bit (immediate)	BOOL	Q
Input (LAD)	BOOL	Power flow
Input (FBD)	BOOL	I, Q, V, M, SM, S, T, C, L, Logic flow

See also

Bit logic output examples (Page 255)

10.1.8 Set, reset, set immediate, and reset immediate functions

LAD	FBD	STL	Description
bit (S) N	bit S N	`S bit, N`	The Set (S) and Reset (R) instructions set (ON) or reset (OFF) the number of bits (N), starting at the address (bit). You can set or reset from 1 to 255 bits. If the Reset instruction specifies either a timer bit (T address) or counter bit (C address), the instruction resets the timer or counter bit and clears the current value of the timer or counter.
bit (R) N	bit R N	`R bit, N`	

LAD	FBD	STL	Description
		`SI bit, N`	The Set Immediate and Reset Immediate instructions immediately set (ON) or immediately reset (OFF) the number of points (N), starting at address (bit). You can set or reset from 1 to 255 points immediately. The "I" indicates an immediate reference; when the instruction is executed, the new value is written to both the physical output point and the corresponding process image register location. This differs from the non-immediate references, which write the new value to the process image register only.
		`RI bit, N`	

Non-fatal errors with ENO = 0	SM bits affected
• N = 0 (zero) • 0006H Indirect address • 0091H Operand out of range	None

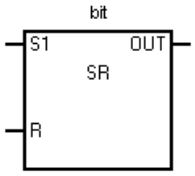
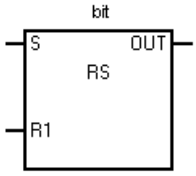
Input / output	Data type	Operand
Bit	BOOL	I, Q, V, M, SM, S, T, C, L
Bit (immediate)	BOOL	Q
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, Constant, *VD, *AC, *LD
Input (LAD)	BOOL	Power flow
Input (FBD)	BOOL	I, Q, V, M, SM, S, T, C, L, Logic flow

See also

Bit logic input examples (Page 253)

Bit logic output examples (Page 255)

10.1.9 Set and reset dominant bistable

LAD/FBD [1]	Description
	The bit parameter assigns the Boolean address that is set or reset. The optional OUT connection reflects the signal state of the Bit parameter. **SR** (Set dominant bistable) is a latch where the set dominates. If the set (S1) and reset (R) signals are both true, the output (OUT) is true.
	**RS** (Reset dominant bistable) is a latch where the reset dominates. If the set (S) and reset (R1) signals are both true, the output (OUT) is false.

[1] Not applicable for STL

Input / outputs	Data type	Operand
bit	BOOL	I, Q, V, M, S
S1, R (LAD SR)	BOOL	Power flow
S, R1 (LAD RS)	BOOL	Power flow
OUT (LAD)	BOOL	Power flow
S1, R (FBD SR)	BOOL	I, Q, V, M, SM, S, T, C, L, Logic flow
S, R1 (FBD RS)	BOOL	I, Q, V, M, SM, S, T, C, L, Logic flow
OUT (FBD)	BOOL	I, Q, V, M, SM, S, T, C, L, Logic flow

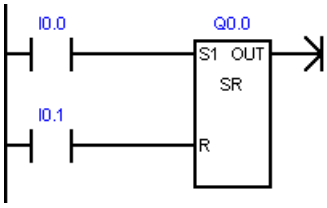
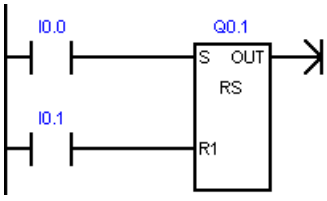
SR truth table

S1	R	Out (bit)
0	0	Previous state
0	1	0
1	0	1
1	1	1

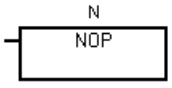
RS truth table

S	R1	Out (bit)
0	0	Previous state
0	1	0
1	0	1
1	1	0

Example SR and RS

LAD	STL
	**NETWORK 1** **LD I0.0** **LD I0.1** **NOT** **A Q0.0** **OLD** **= Q0.0**
	**NETWORK 2** **LD I0.0** **LD I0.1** **NOT** **LPS** **A Q0.1** **= Q0.1** **LPP** **ALD** **O Q0.1** **= Q0.1**

10.1.10 NOP (No operation) instruction

LAD	STL	Description
	**NOP N**	The No Operation (NOP) instruction has no effect on the user program execution. This instruction is not available in FBD mode. The operand N is a number from 0 to 255.

Inputs / Output	Data type	Operand
N (LAD, STL)	BYTE	N: Constant (0 to 255)

10.1.11 Alternating output and alternating output on pulse

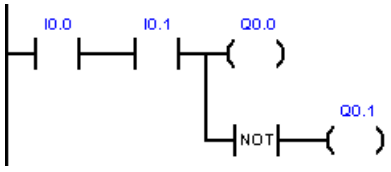
LAD	FBD	STL	Description
bit (ALT)	bit =ALT	**=ALT bit**	The Alternating output instruction (ALT) continues to invert the state of the output bit when the power flow is valid. If the condition is not met, the output bit remains as it is. **LAD/FBD**: The ALT contact instruction changes the state of output bit when power flow is valid. When power flow reaches the ALT contact, the state of the output bit inverts on each scan cycle, from 0 to 1 or from 1 to 0. When power flow does not reach the ALT contact, the output bit remains as it is. **STL**: When the power flow is valid, the ALT instruction keeps switching the value of output bit between 0 and 1.
bit (ALTP)	bit =ALTP	**=ALTP bit**	The Alternating output on pulse instruction (ALTP) inverts the state of the output bit on each rising edge. **LAD/FBD**: The ALTP contact changes the state of output bit on each rising edge. When a rising edge reaches the ALTP contact, the state of the output bit inverts, from 0 to 1 or from 1 to 0. Otherwise the output bit remains as it is. **STL**: The ALTP instruction switches the value of the output bit between 0 and 1 on each rising edge.

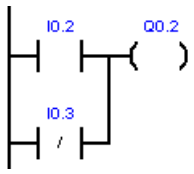
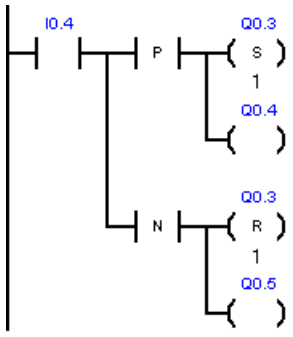
Input / output	Data type	Operand
IN	BOOL	Power flow
OUT	BOOL	Q, V, M

See also

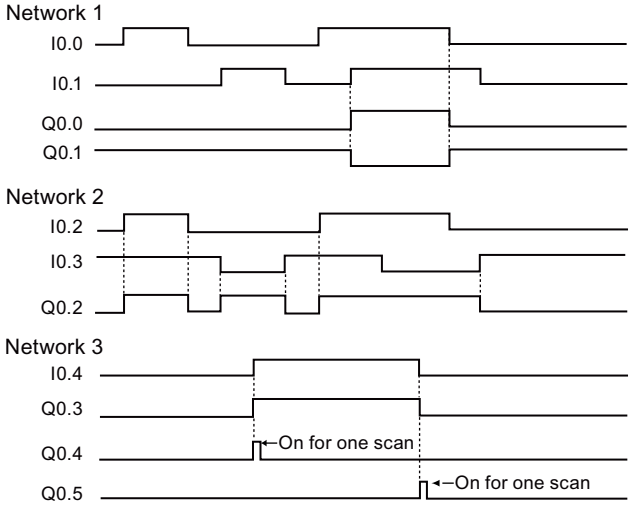
Bit logic output examples (Page 255)

10.1.12 Bit logic input examples

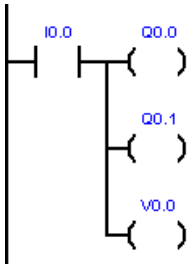
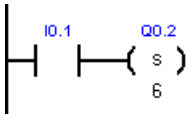
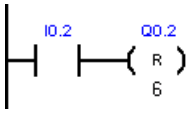
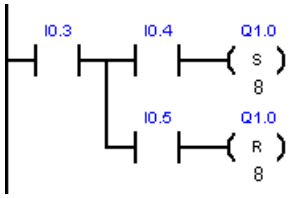
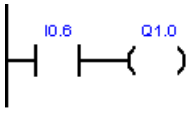
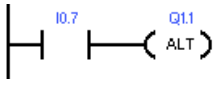
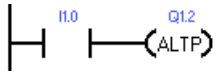
LAD		STL
	Normally-open contacts I0.0 AND I0.1 must be ON (closed) to activate Q0.0. The NOT instruction acts as an inverter. In RUN mode, Q0.0 and Q0.1 have opposite logic states.	**Network 1** **LD I0.0** **A I0.1** **= Q0.0** **NOT** **= Q0.1**

	(Normally-open contact I0.2 must be ON) or (Normally-closed contact I0.3 must be OFF), to activate Q0.2. One or more parallel LAD branches (OR logic) must be true to make the output active.	**Network 2** **LD I0.2** **ON I0.3** **= Q0.2**
	A positive Edge Up input on a P contact or a negative Edge Down input on an N contact outputs a pulse with a 1 scan cycle duration. In RUN mode, the pulsed state changes of Q0.4 and Q0.5 are too fast to be visible in program status view. The Set and Reset outputs latch the pulse state into Q0.3 and make the state change visible in program status view.	**Network 3** **LD I0.4** **LPS** **EU** **S Q0.3, 1** **= Q0.4** **LPP** **ED** **R Q0.3, 1** **= Q0.5**

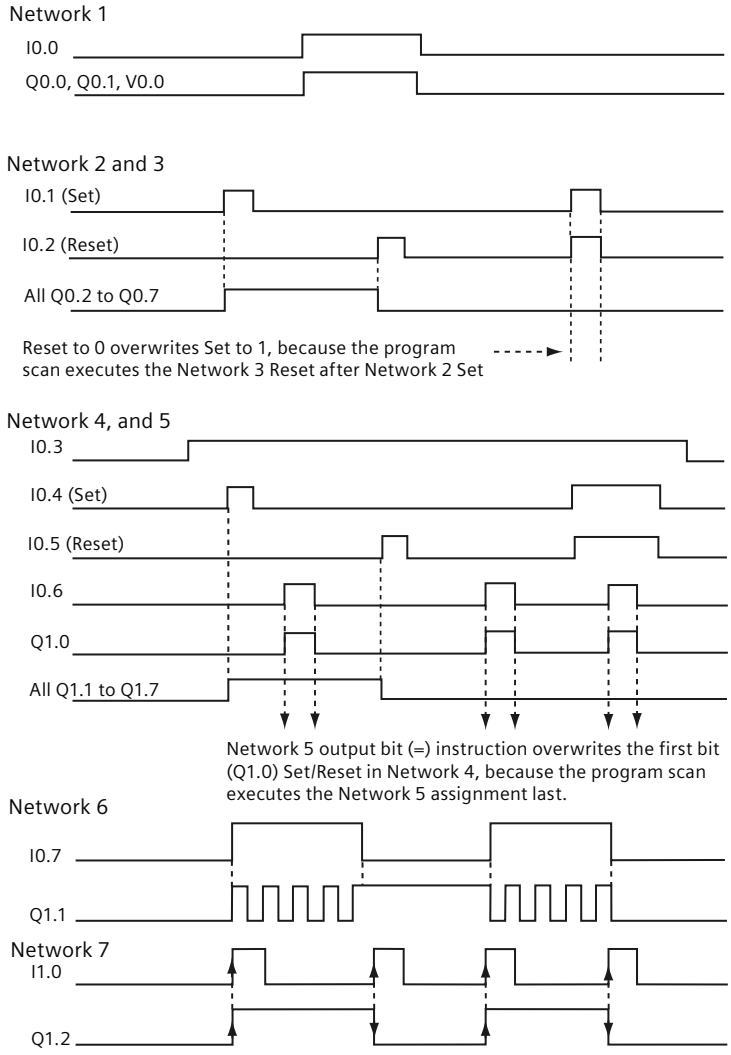
Run-mode timing for input example



10.1.13 Bit logic output examples

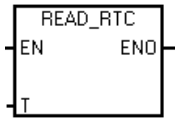
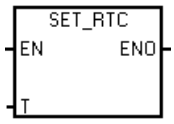
LAD		STL
	Output instructions assign bit values to external I/O (I, Q) and internal memory (M, SM, T, C, V, S, L).	**Network 1** **LD I0.0** **= Q0.0** **= Q0.1** **= V0.0**
	Set a sequential group of 6 bits to a value of 1. Specify a starting bit address and how many bits to set. The program status indicator for Set is ON when the value of the first bit (Q0.2) is 1.	**Network 2** **LD I0.1** **S Q0.2, 6**
	Reset a sequential group of 6 bits to a value of 0. Specify a starting bit address and how many bits to reset. The program status indicator for Reset is ON when the value of the first bit (Q0.2) is 0.	**Network 3** **LD I0.2** **R Q0.2, 6**
	Sets and resets 8 output bits (Q1.0 to Q1.7) as a group.	**Network 4** **LD I0.3** **LPS** **A I0.4** **S Q1.0, 8** **LPP** **A I0.5** **R Q1.0, 8**
	The Set and Reset instructions perform the function of a latched relay. To isolate the Set/Reset bits, make sure they are not overwritten by another assignment instruction. In this example, Network 4 sets and resets eight output bits (Q1.0 to Q1.7) as a group. In RUN mode, Network 5 can overwrite the Q1.0 bit value and control the Set/Reset program status indicators in Network 4.	**Network 5** **LD I0.6** **= Q1.0**
	The Alternating output on pulse instruction (ALT) inverts the state of the output bit on each scan cycle.	**Network 6** **LD I0.7** **=ALT Q1.1**
	The Alternating output on pulse instruction (ALTP) inverts the state of the output bit on each rising edge.	**Network 7** **LD I1.0** **=ALTP Q1.2**

Run-mode timing for output examples



10.2 Clock

10.2.1 Read and set real-time clock

LAD / FBD	STL	Description
	**TODR T**	The Read real-time clock instruction reads the current time and date from the CPU and loads it in an 8 byte Time buffer starting at byte address T.
	**TODW T**	The Set real-time clock instruction writes a new time and date to the CPU using the 8 byte Time buffer data that is assigned by T.

Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address • 0007H T data error	None

Input	Data type	Operand
T	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC

Note

READ_RTC, SET_RTC programming tips

These instructions do not accept Invalid dates. If you enter February 30, for example, a time-of-day non-fatal error occurs (0007H).

Do not use the READ_RTC / SET_RTC instructions in both the main program and in an interrupt routine. A READ_RTC / SET_RTC instruction in an interrupt routine cannot execute while another READ_RTC / SET_RTC instruction is executing. In this case, the CPU sets system flag bit SM4.3, indicating that two simultaneous accesses to the clock were attempted resulting in a T data error (non-fatal error 0007H).

The time-of-day clock in the CPU uses only the least significant two digits for the year, so 00 represents the year 2000. User programs that use the year's value must take into account the two-digit representation.

The CPU handles leap year correctly through year 2099.

Format of 8 byte time buffer, beginning at byte address T

You must assign all date and time values in BCD format (for example, 16#12 for the year 2012). The BCD value range of 00 to 99 can assign years, in 2000 to 2099 range.

T byte	Description	Data value
0	Year	00 to 99 (BCD value) Year 20xx: where xx is the two digit BCD value in T-byte 0
1	Month	01 to 12 (BCD value)
2	Day	01 to 31 (BCD value)
3	Hour	00 to 23 (BCD value)
4	Minute	00 to 59 (BCD value)
5	Second	00 to 59 (BCD value)
6	reserved	Always set to 00
7	Day of week	Value ignored when written with the SET_RTC / TODW instruction. Value reports correct day of week when read with the READ_RTC / TODR instruction based upon current Year/Month/Day values. 1 to 7, 1 = Sunday, 7 = Saturday (BCD value)

Extended power outage effect on the CPU clock

See the S7-200 SMART system manual appendix A CPU specifications, for how long the real-time clock can maintain the correct time during power outages.

A CPU initializes with the time values shown in the following table, after an extended power outage.

Date	Time	Day of week
01-Jan-2000	00:00:00	Saturday

Note

Compact serial (CRs) CPU models do not have a RTC (Real-time Clock)

You can use the READ_RTC and SET_RTC instructions to set the year, date, and time values in compact serial (CRs) CPU models, but the values will be lost on the next CPU power off-on cycle. On power-up, the date and time will initialize to January 1, 2000.

10.2.2 Read and set real-time clock extended

LAD / FBD	STL	Description
READ_RTCX EN ENO T	**TODRX T**	The Read real-time clock extended instruction reads the current time, date, and daylight saving configuration from the PLC and loads it in a 19-byte buffer beginning at the address assigned by T.
SET_RTCX EN ENO T	**TODWX T**	The Set real-time clock instruction writes a new time, date, and daylight saving configuration to the PLC using the 19-byte buffer data that is assigned by byte address T.

Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address • 0007H T data error • 0091H Operand out of range	None

Input	Data type	Operand
T	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC

Note

READ_RTCX, SET_RTCX programming tips

These instructions do not accept Invalid dates. If you enter February 30, for example, a time-of-day non-fatal error occurs (0007H).

Do not use the READ_RTCX / SET_RTCX instructions in both the main program and in an interrupt routine. A READ_RTCX / SET_RTCX instruction in an interrupt routine cannot execute while another READ_RTCX / SET_RTCX instruction is executing. In this case, the CPU sets system flag bit SM4.3, indicating that two simultaneous accesses to the clock were attempted resulting in a T data error (non-fatal error 0007H).

The time-of-day clock in the CPU uses only the least significant two digits for the year, so 00 represents the year 2000. User programs that use the year's value must take into account the two-digit representation.

The CPU handles leap year correctly through year 2099.

Format of 19 byte time buffer, beginning at byte address T

Note

T bytes (9 to18) or (9 to 20) are used only when a time correction mode is assigned in byte 8. Otherwise, the last values written to bytes (9 to18) or (9 to 20) by STEP 7-Micro/WIN SMART or the SET_RTCX instruction are returned.

You must assign all date and time values in BCD format (for example, 16#12 for the year 2012). The BCD value range of 00 to 99 can assign the year in the range of 2000 to 2099.

T byte	Description	Data value
0	Year	00 to 99 (BCD value) Year 20xx: where xx is the two digit BCD value in T-byte 0
1	Month	01 to 12 (BCD value)
2	Day	01 to 31 (BCD value)
3	Hour	00 to 23 (BCD value)
4	Minute	00 to 59 (BCD value)
5	Second	00 to 59 (BCD value)
6	reserved	Always set to 00
7	Day of week	Value ignored when written with the SET_RTCX / TODWX instruction. Value reports correct day of week when read with the READ_RTCX / TODRX instruction based upon current Year/Month/Day values. 1 to 7, 1 = Sunday, 7 = Saturday (BCD value)
8	Correction mode: For Daylight saving time (DST)	00H = correction disabled 01H = EU (time zone offset from UTC = 0 hrs) [1] 02H = EU (time zone offset from UTC = +1 hrs) [1] 03H = EU (time zone offset from UTC = +2 hrs) [1] 04H-07H = reserved 08H = EU (time zone offset from UTC = -1 hrs) [1] 09H-0FH = reserved 10H = US [2] 11H = Australia [3] 12H = reserved 13H = New Zealand [4] 14H-EDH = reserved EEH = user defined (day of week) (using values in bytes 9-20) EFH-FDH reserved FEH = reserved FFH = user defined (day of month) (using values in bytes 9-18)
The following bytes 9-18 are used only for correction mode = FFH (legacy user assigned)		
9	DST correction hours	0 to 23 (BCD value)
10	DST correction minutes	0 to 59 (BCD value)
11	DST beginning month	1 to 12 (BCD value)
12	DST beginning day	1 to 31 (BCD value)
13	DST beginning hour	0 to 23 (BCD value)
14	DST beginning minute	0 to 59 (BCD value)
15	DST ending month	1 to 12 (BCD value)
16	DST ending day	1 to 31 (BCD value)
17	DST ending hour	0 to 23 (BCD value)
18	DST ending minute	0 to 59 (BCD value)
The following bytes 9-20 are used only for correction mode = EEH (extended user assigned)		
9	DST correction hours	0 to 23 (BCD value)
10	DST correction minutes	0 to 59 (BCD value)
11	DST beginning month	1 to 12 (BCD value)

T byte	Description	Data value
12	DST beginning week	1 to 5 (BCD value) [5]
13	DST beginning weekday	1 to 7 (BCD value)
14	DST beginning hour	0 to 23 (BCD value)
15	DST beginning minute	0 to 59 (BCD value)
16	DST ending month	1 to 12 (BCD value)
17	DST ending week	1 to 5 (BCD value) [5]
18	DST ending weekday	1 to 7 (BCD value)
19	DST ending hour	0 to 23 (BCD value)
20	DST ending minute	0 to 59 (BCD value)

[1] EU convention: Adjust time ahead one hour on last Sunday in March at 1:00 a.m. UTC. Adjust time back one hour on last Sunday in October at 2:00 a.m. UTC. (The local time when the correction is made depends upon the time zone offset from UTC).

[2] US convention: 2007 standard - Adjust time ahead one hour on second Sunday in March at 2:00 a.m. local time. Adjust time back one hour on first Sunday in November at 2:00 a.m. local time.

[3] Australia convention: 2007 standard - Adjust time ahead one hour on first Sunday in October at 2:00 a.m. local time. Adjust time back one hour on first Sunday in April at 2:00 a.m. local time (also for Australia - Tasmania).

[4] New Zealand convention: 2007 standard - Adjust time ahead one hour on last Sunday in September at 2:00 a.m. local time. Adjust time back one hour on first Sunday in April at 2:00 a.m. local time.

[5] To assign the last occurrence of the weekday in the month (for example, the last Monday in April), set the week = 5.

Extended power outage effect on the CPU clock

See the S7-200 SMART system manual appendix A CPU specifications, for how long the real-time clock can maintain the correct time during power outages.

A CPU initializes with the time values shown in the following table, after an extended power outage.

Date	Time	Day of week
01-Jan-2000	00:00:00	Saturday

Note

Compact serial (CRs) CPU models do not have a RTC (Real-time Clock)

You can use the READ_RTCX and SET_RTCX instructions to set the year, date, and time values in compact serial (CRs) CPU models, but the values will be lost on the next CPU power off-on cycle. On power-up, the date and time will initialize to January 1, 2000.

10.3 Communications

10.3.1 GET and PUT (Ethernet)

You can use the GET and PUT instructions for communication between S7-200 SMART CPUs through the Ethernet connection.

Note

To use PUT & GET, you must select "Enable Put/Get Server (Page 183)".

Table 10-1 GET and PUT instructions

LAD/FBD	STL	Description
GET EN ENO TABLE	**GET table**	The GET instruction initiates a communications operation on the Ethernet port to gather data from a remote device, as defined in the description table (TABLE). The GET instruction can read up to 222 bytes of information from a remote station.
PUT EN ENO TABLE	**PUT table**	The PUT instruction initiates a communications operation on the Ethernet port to write data to a remote device, as defined in the description table (TABLE). The PUT instruction can write up to 212 bytes of information to a remote station.

You can have any number of GET and PUT instructions in the program, but only a maximum of 16 GET and PUT instructions can be activated at any one time. For example, you can have eight GET and eight PUT instructions, or six GET and ten PUT instructions, active at the same time in a given CPU.

When you execute a GET or PUT instruction, the CPU makes an Ethernet connection to the remote IP address in the GET or PUT table. The CPU maintains a maximum of eight connections at a time. Once a connection is established, that connection is maintained until the CPU goes to STOP mode.

The CPU utilizes a single connection for all GET/PUT instructions that are directed to the same IP address. For example, if there are three GET instructions enabled at the same time when the remote IP address is 192.168.2.10, then the GET instructions execute sequentially on one Ethernet connection to IP address 192.168.2.10.

If you try to create a ninth connection (a ninth IP address), the CPU searches through the connections to find the connection that has been inactive for the longest period of time. The CPU disconnects this connection and then creates a connection to the new IP address.

The GET and PUT instructions require additional communication background time (refer to "Configuring communication" (Page 183)) when they are processing/active/busy and also when they are just maintaining the connection to the other device. The amount of communication background time required depends on the number of GET and PUT instruction that are active/busy, how often the GET and PUT instructions are executed, and

the number of connections that are currently open. You should adjust the communication background time to a higher value if the communication performance is slow.

Table 10-2 Valid operands for the GET and PUT instructions

Inputs/Outputs	Data Type	Operands
TABLE	BYTE	IB, QB, VB, MB, SMB, SB, *VD, *LD, *AC

Error conditions that set ENO = 0:

- 0006 (indirect address)
- If the function returns an error and sets the E bit of table status byte (see the figure below)

The following figure describes the table that is referenced by the TABLE parameter, and the following table lists the error codes.

Table 10-3 Definition of GET and PUT instructions TABLE parameter

Byte off-set	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	D [1]	A [2]	E [3]	0	Error code			
1	Remote station IP Address [4]							
2								
3								
4								
5	Reserved = 0 (Must be set to zero)							
6	Reserved = 0 (Must be set to zero)							
7	Pointer to the data area in the remote station (I, Q, M, V, or DB1) [5]							
8								
9								
10								
11	Data length [6]							
12	Pointer to the data area in the local station (this CPU) (I, Q, M, V, or DB1) [7]							
13								
14								
15								

1 D - Done (function has been completed)

2 A - Active (function has been queued)

3 E - Error (function returned an error)

4 Remote station IP address: The address of the CPU whose data is to be accessed.

5 Pointer to the data area in the remote station: An indirect pointer to the data that is to be accessed in the remote station.

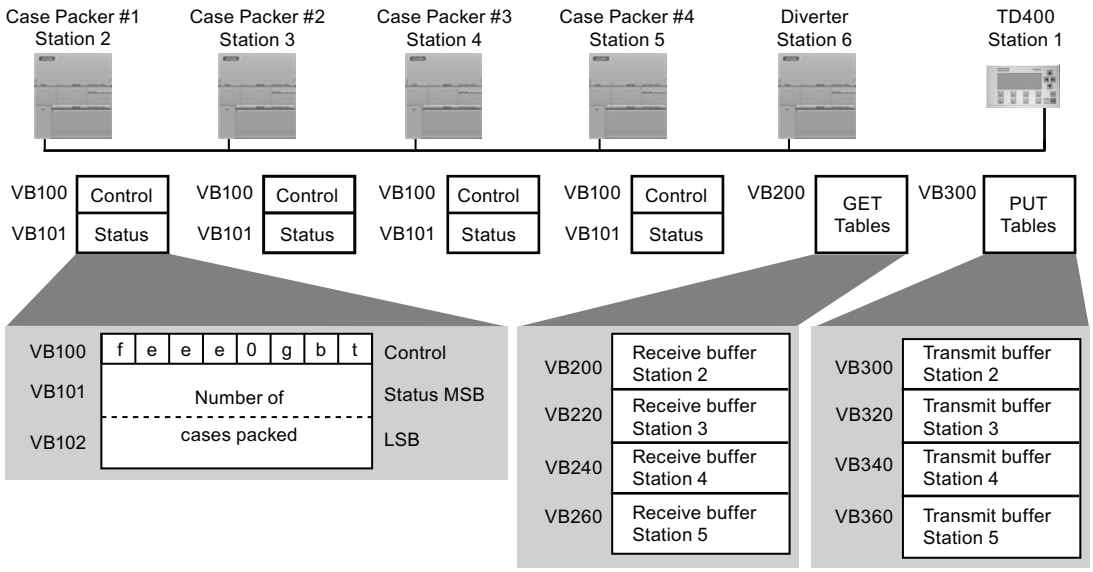
6 Data length: The number of bytes of data that are to be accessed in the remote station (1 to 212 bytes for PUT and 1 to 222 bytes for GET).

7 Pointer to the data area in the local station: An indirect pointer to the data that is to be accessed in the local station (this CPU).

Table 10-4 Error codes for the GET and PUT instructions TABLE parameter

Code	Definition
0	No error
1	Illegal parameter in the PUT/GET table: • Local area is not I, Q, M, or V • Local area is not large enough for the data length requested • Data length is zero or greater than 222 bytes for a GET or greater than 212 bytes for a PUT • Remote area is not I, Q, M, or V • Remote IP address is illegal (0.0.0.0) • Remote IP address is a broadcast address or a multicast address • Remote IP address is the same as the Local IP address • Remote IP address is on a different subnet
2	Too many PUT/GET instructions are currently active (only 16 allowed)
3	No connection available. All connections are currently active with outstanding requests
4	Error returned from remote CPU: • Too much data was requested or sent • Writing to Q memory is not allowed in STOP mode • Memory area is write-protected (see SDB configuration)
5	No connection available to the remote CPU: • Remote CPU does not have an available server connection • Connection to remote CPU was lost (CPU powered off, physical disconnect)
6 to 9, A to F	Not used (Reserved for future use)

The following figure shows an example to illustrate the utility of the GET and PUT instructions. For this example, consider a production line where tubs of butter are being filled and sent to one of four boxing machines (case packers). The case packer packs eight tubs of butter into a single cardboard box. A diverter machine controls the flow of butter tubs to each of the case packers. Four CPUs control the case packers, and a CPU with a TD 400 operator interface controls the diverter.



t	Out of butter tubs to pack; t=1, out of butter tubs
b	Box supply is low; b=1, must add boxes in the next 30 minutes
g	Glue supply is low; g=1, must add glue in the next 30 minutes
eee	Error code identifying the type of fault experienced
f	Fault indicator; f=1, the case packer has detected an error

The following figure shows the GET table (VB200) and PUT table (VB300) for accessing the data in station 2. The Diverter CPU uses a GET instruction to read the control and status information on a continuous basis from each of the case packers. Each time a case packer has packed 100 cases, the diverter notes this and sends a message to clear the status word using a PUT instruction.

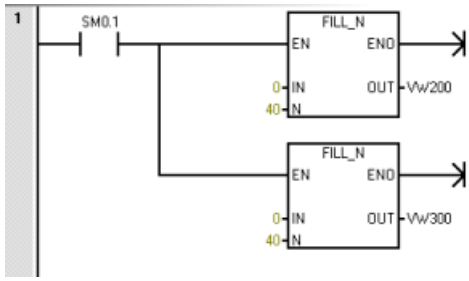
Table 10-5 GET and PUT instructions buffer for reading from and clearing the count of Case Packer 1

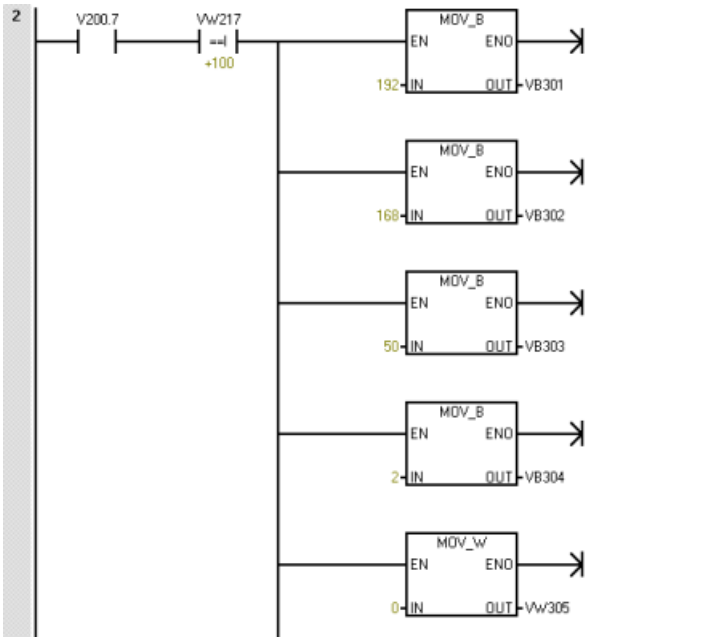
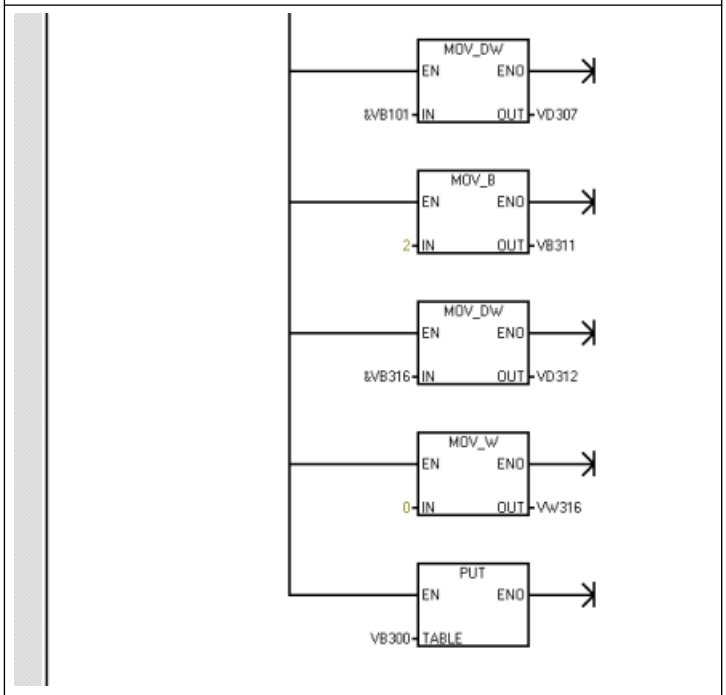
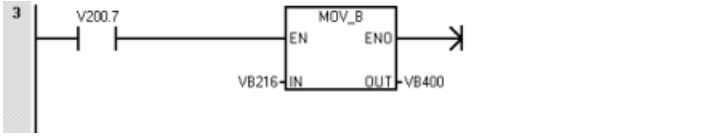
GET_ TABLE buffer	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**PUT_ TABLE buffer**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
VB200	D	A	E	0	Error code				VB300	D	A	E	0	Error code			
VB201	Remote station IP address = 192.								VB301	Remote station IP address = 192.							
VB202	168.								VB302	168.							
VB203	50.								VB303	50.							
VB204	2								VB304	2							
VB205	Reserved = 0 (Must be set to zero)								VB305	Reserved = 0 (Must be set to zero)							
VB206	Reserved = 0 (Must be set to zero)								VB306	Reserved = 0 (Must be set to zero)							
VB207	Pointer to the data								VB307	Pointer to the data							
VB208	area in the								VB308	area in the							
VB209	remote station =								VB309	remote station =							
VB210	(&VB100)								VB310	(&VB101)							
VB211	Data length = 3 bytes								VB311	Data length = 2 bytes							
VB212	Pointer to the data								VB312	Pointer to the data							
VB213	area in the								VB313	area in the							

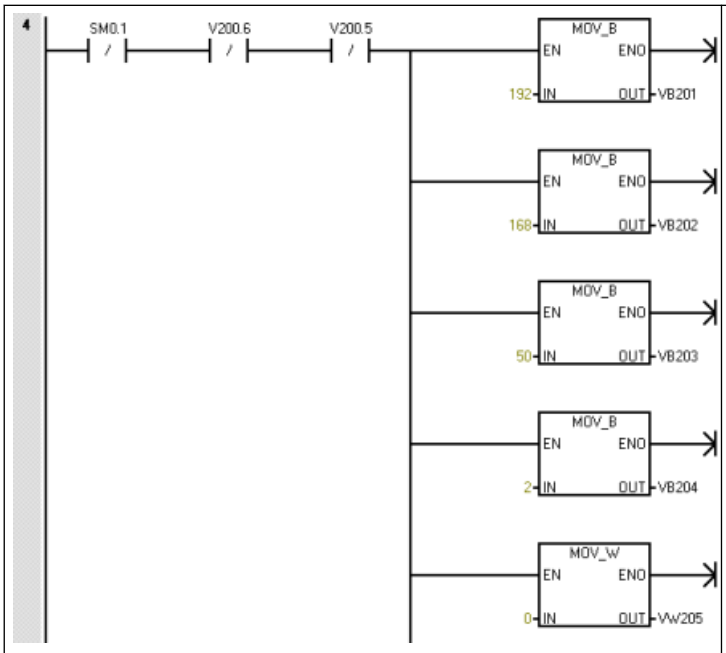
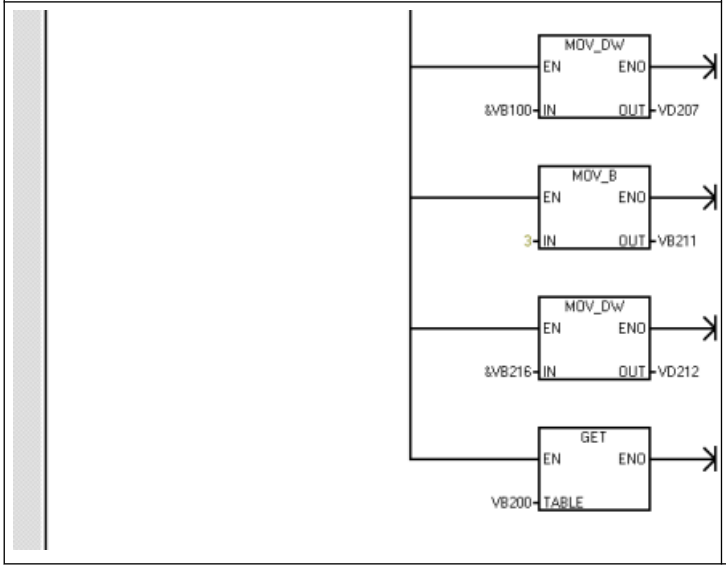
GET_ TABLE buffer	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	PUT_ TABLE buffer	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
VB214	local station (this CPU) =								VB314	local station (this CPU) =							
VB215	(&VB216)								VB315	(&VB316)							
VB216	Control								VB316	0							
VB217	Status MSB								VB317	0							
VB218	Status LSB																

In this example, the data immediately follows the PUT and GET tables. This data can be placed anywhere in the CPU memory since it is pointed to by the local station pointer in a table (for example, VB212 - VB215).

Table 10-6 Example: GET and PUT instructions

	**Network 1** **LD SM0.1** **FILL +0, VW200, 40** **FILL +0, VW300, 40**	On the first scan, clear all receive and transmit buffers.

	**Network 2** **LD V200.7** **AW= VW217, +100** **MOVB 192, VB301** **MOVB 168, VB302** **MOVB 50, VB303** **MOVB 2, VB304** **MOVW 0, VB305** **MOVD &VB101, VD307** **MOVB 2, VB311** **MOVD &VB316, VD312** **MOVW 0, VW316** **PUT VB300**	When the GET "Done" bit (V200.7) is set and 100 cases have been packed: 1. Load the station address of case packer 1. 2. Load a pointer to the data in the remote station. 3. Load the length of data to be transmitted. 4. Load the data to transmit. Reset the number of cases packed by case packer 1
		
	**Network 3** **LD V200.7** **MOVB VB216, VB400**	When the GET "Done" bit is set, save the control data from case packer 1.

	**Network 4** **LDN SM0.1** **AN V200.6** **AN V200.5** **MOVB 192, VB201** **MOVB 168, VB202** **MOVB 50, VB203** **MOVB 2, VB204** **MOVW 0, VB205** **MOVD &VB100, VD207** **MOVB 3, VB211** **MOVD &VB216, VD212** **GET VB200**	If not the first scan and there are no errors: 1. Load the station address of case packer 1. 2. Load a pointer to the data in the remote station. 3. Load the length of data to be received. 4. Read the control and status data in case packer 1.
		

10.3.2 Transmit and receive (Freeport on RS485/RS232)

You can use the Transmit (XMT) and Receive (RCV) instructions for communication between a S7-200 SMART CPU and other devices through the CPU serial port(s). Each S7-200 SMART CPU provides an integrated RS485 port (Port 0). The standard CPUs additionally support two optional CM01 Signal Board (SB) RS232/RS485 port (Port 1 and port 2). The communication protocol must be implemented in the user program.

LAD / FBD	STL	Description
XMT EN ENO TBL PORT	**XMT TBL, PORT**	The Transmit instruction (XMT) is used in Freeport mode to transmit data by means of the communications port(s).
RCV EN ENO TBL PORT	**RCV TBL, PORT**	The Receive instruction (RCV) initiates or terminates the receive message function. You must specify a start and an end condition for the Receive box to operate. Messages received through the specified port (PORT) are stored in the data buffer (TBL). The first entry in the data buffer specifies the number of bytes received.

Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address • 0009H Simultaneous Transmit/ Receive on port 0 • 000BH Simultaneous Transmit/ Receive on port 1 • 0x001F Simultaneous Transmit/ Receive on port 2 • 0090H Port number is invalid • Receive parameter error sets SM86.6, SM186.6 or SM405.6 • CPU is not in Freeport mode	• SM 86.6 Receive message terminated on port 0 • SM 186.6 Receive message terminated on port 1 • SM 405.6 Receive message terminated on port 2

Input / output	Data type	Operand
TBL	BYTE	IB, QB, VB, MB, SMB, SB, *VD, *LD, *AC
PORT	BYTE	Constant: 0, 1 or 2 Note: The three available ports are as follows: • Integrated RS485 port (Port 0), • CM01 Signal Board (SB) RS232/RS485 port (Port 1 and Port 2)

Using Freeport mode to control the serial communications port

You can select the Freeport mode to control the serial communications port of the CPU by means of your user program. When you select Freeport mode, your program controls the operation of the communications port through the use of the receive interrupts, the transmit interrupts, the Transmit instruction, and the Receive instruction and entirely controls the communications protocol while in Freeport mode. You use SMB30, SMB130and SMB31 to select the baud rate and parity.

The CPU assigns three special memory bytes to the two physical ports:

- SMB30 to the integrated RS485 port (Port 0)
- SMB130 to the CM01 RS232/RS485 Signal Board (SB) port (Port 1)
- SMB31 to the CM01 RS232/RS485 Signal Board (SB) port (Port 2)

The Freeport mode is disabled and normal communications are re-established (for example, HMI device access) when the CPU is in STOP mode.

In the simplest case, you can send a message to a printer or a display using only the Transmit (XMT) instruction. Other examples include a connection to a bar code reader, a weigh scale, and a welder. In each case, you must write your program to support the protocol that is used by the device with which the CPU communicates while in Freeport mode.

You can only use Freeport communications when the CPU is in RUN mode. Enable the Freeport mode by setting a value of 01 in the protocol select field of SMB30 (Port 0), SMB130 (Port 1) or SMB31(Port 2). While in Freeport mode, you cannot communicate with an HMI on the same port.

Changing PPI communications to Freeport mode

SMB30, SMB130 and SMB31 configure the communications ports, 0, 1 and 2 respectively, for Freeport operation and provide selection of baud rate, parity, and number of data bits. The following figure describes the Freeport control byte. One stop bit is generated for all configurations.

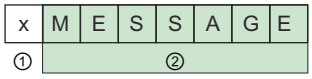
MSB 7							LSB 0
p	p	d	b	b	b	m	m

pp	Parity select		d	Data bits per character	
	00 =	No parity		0 =	8 bits per character
	01 =	Even parity		1 =	7 bits per character
	10 =	No parity			
	11 =	Odd parity			

bbb	Freeport baud rate		mm	Protocol selection	
	000 =	38400		00 =	PPI slave mode
	001 =	19200		01 =	Freeport mode
	010 =	9600		10 =	Reserved (defaults to PPI slave mode)
	011 =	4800		11 =	Reserved (defaults to PPI slave mode)
	100 =	2400			
	101 =	1200			
	110 =	115200			
	111 =	57600			

Transmit data

The Transmit instruction lets you send a buffer of one or more characters, up to a maximum of 255. The following figure shows the format of the Transmit buffer.



① Number of bytes to transmit

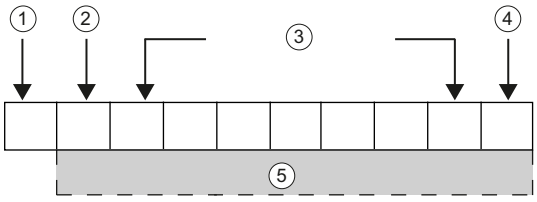
② Characters of the message

If an interrupt routine is attached to the transmit complete event, the CPU generates an interrupt (interrupt event 9 for port 0, interrupt event 26 for port 1, interrupt event 48 for port 2) after the last character of the buffer is sent.

You can transmit without using interrupts (for example, sending a message to a printer) by monitoring SM4.5 (port 0), SM4.6 (port 1) or SM20.0(port 2) to signal when transmission is complete.

You can use the Transmit instruction to generate a BREAK condition by setting the number of characters to zero and then executing the Transmit instruction. This generates a BREAK condition on the line for 16-bit times at the current baud rate. Transmitting a BREAK is handled in the same manner as transmitting any other message, in that a Transmit interrupt is generated when the BREAK is complete and SM4.5, SM4.6 or SM22.0 signals the current status of the Transmit operation.

Receive data



The Receive instruction lets you receive a buffer of one or more characters, up to a maximum of 255. The following figure shows the format of the Receive buffer.

① Number of bytes received (byte field)

② Start character

③ Message

④ End character

⑤ Characters of the message

If an interrupt routine is attached to the receive message complete event, the CPU generates an interrupt (interrupt event 23 for port 0, interrupt event 24 for port 1 and interrupt event 46 for port 2) after the last character of the buffer is received.

You can receive messages without using interrupts by monitoring SMB86 (port 0) or SMB186 (port 1) or SMB405 (port 2). This byte is non-zero when the Receive instruction is inactive or has been terminated. It is zero when a receive is in progress.

As shown in the following table, the Receive instruction allows you to select the message start and message end conditions, using SMB86 through SMB94 for port 0, SMB186 through SMB194 for port 1 and SMB405 throgh SMB413 for port 2.

Note

The receive message function is automatically terminated in case of a framing, parity, overrun, or break error.

You must define a start condition and an end condition (maximum character count) for the receive message function to operate.

Receive buffer format (SMB86 to SMB94, SMB186 to SMB194, and SMB405 to SMB413)

Port 0	Port 1	Port 2	Description
SMB86	SMB186	SMB405	Receive message status byte MSB 7 … LSB 0 n r e 0 0 t c p n: 1 = Receive message function terminated; user issued disable command. r: 1 = Receive message function terminated; error in input parameters or missing start or end condition. e: 1 = End character received. t: 1 = Receive message function terminated; timer expired. c: 1 = Receive message function terminated; maximum character count achieved. p: 1 = Receive message function terminated; a parity error.
SMB87	SMB187	SMB406	Receive message control byte MSB 7 … LSB 0 en sc ec il c/m tmr bk 0 en: 0 = Receive message function is disabled. 1 = Receive message function is enabled. The enable/disable receive message bit is checked each time the RCV instruction is executed. sc: 0 = Ignore SMB88, SMB188 or SMB407. 1 = Use the value of SMB88 or SMB188 to detect start of message. ec: 0 = Ignore SMB89, SMB189 or SMB408. 1 = Use the value of SMB89 or SMB189 to detect end of message. il: 0 = Ignore SMB90, SMB190 or SMB409. 1 = Use the value of SMB90 or SMB190 to detect start of message. c/m: 0 = Timer is an inter-character timer. 1 = Timer is a message timer. tmr: 0 = Ignore SMW92, SMW192 or SMW411. 1 = Terminate receive if the time period in SMW92 or SMW192 is exceeded. bk: 0 = Ignore break conditions. 1 = Use break condition as start of message detection.
SMB88	SMB188	SMB407	Start of message character.
SMB89	SMB189	SMB408	End of message character.
SMW90	SMW190	SMW409	Idle line time period given in milliseconds. The first character received after idle line time has expired is the start of a new message.

Port 0	Port 1	Port 2	Description
SMW92	SMW192	SMW411	Inter-character/message timer time-out value given in milliseconds. If the time period is exceeded, the receive message function is terminated.
SMB94	SMB194	SMB413	Maximum number of characters to be received (1 to 255 bytes). This range must be set to the expected maximum buffer size, even if the character count message termination is not used.

Start and End conditions for the Receive instruction

The Receive instruction uses the bits of the receive message control byte (SMB87 or SMB187 or SMB406) to define the message start and end conditions.

Note

If there is traffic present on the communications port from other devices when the Receive instruction is executed, the receive message function could begin receiving a character in the middle of that character, resulting in a possible parity or framing error and termination of the receive message function. If parity is not enabled the received message could contain incorrect characters. This situation can occur when the start condition is specified to be a specific start character or any character, as described in item 2 and item 6 below.

The Receive instruction supports several message start conditions. Specifying a start condition involving a break or an idle line detection avoids the problem of starting a message in the middle of a character by forcing the receive message function to synchronize the start of the message with the start of a character before placing characters into the message buffer.

The Receive instruction supports several start conditions:

1. *Idle line detection:* The idle line condition is defined as a quiet or idle time on the transmission line. A receive is started when the communications line has been quiet or idle for the number of milliseconds specified in SMW90 or SMW190 or SMW409. When the Receive instruction in your program is executed, the receive message function initiates a search for an idle line condition. If any characters are received before the idle line time expires, the receive message function ignores those characters and restarts the idle line timer with the time from SMW90 or SMW190 or SMW409. See the following figure. After the idle line time expires, the receive message function stores all subsequent characters received in the message buffer.
   The idle line time should always be greater than the time to transmit one character (start bit, data bits, parity and stop bits) at the specified baud rate. A typical value for the idle line time is three character times at the specified baud rate.
   You use idle line detection as a start condition for binary protocols, protocols where there is not a particular start character, or when the protocol specifies a minimum time between messages.
   Setup: il = 1, sc = 0, bk = 0, SMW90/SMW190/SMW409 = idle line timeout in milliseconds

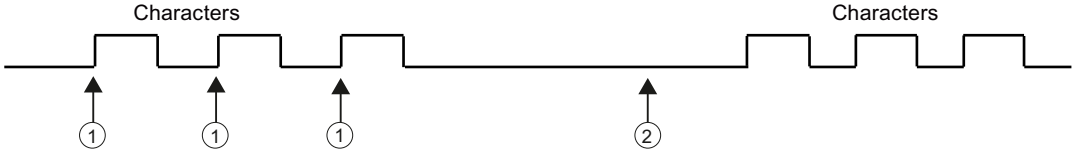
① Receive instruction is executed: Starts the idle time
② Restarts the idle time
③ Idle time is detected: Starts the Receive Message function
④ First character is placed in the message buffer

2. *Start character detection:* The start character is any character which is used as the first character of a message. A message is started when the start character specified in SMB88, SMB188 or SMB407 is received. The receive message function stores the start character in the receive buffer as the first character of the message. The receive message function ignores any characters that are received before the start character. The start character and all characters received after the start character are stored in the message buffer.
   Typically, you use start character detection for ASCII protocols in which all messages start with the same character.
   Setup: il = 0, sc = 1, bk = 0, SMW90/SMW190/SMW409 = don't care, SMB88/SMB188/SMB407 = start character

3. *Idle line and start character:* The Receive instruction can start a message with the combination of an idle line and a start character. When the Receive instruction is executed, the receive message function searches for an idle line condition. After finding the idle line condition, the receive message function looks for the specified start character. If any character but the start character is received, the receive message function restarts the search for an idle line condition. All characters received before the idle line condition has been satisfied and before the start character has been received are ignored. The start character is placed in the message buffer along with all subsequent characters.
   The idle line time should always be greater than the time to transmit one character (start bit, data bits, parity and stop bits) at the specified baud rate. A typical value for the idle line time is three character times at the specified baud rate.
   Typically, you use this type of start condition when there is a protocol that specifies a minimum time between messages, and the first character of the message is an address or something which specifies a particular device. This is most useful when implementing a protocol where there are multiple devices on the communications link. In this case the Receive instruction triggers an interrupt only when a message is received for the specific address or devices specified by the start character.
   Setup: il = 1, sc = 1, bk = 0, SMW90/SMW190/SMW409 > 0, SMB88/SMB188/SMB407 = start character
4. *Break detection:* A break is indicated when the received data is held to a zero value for a time greater than a full character transmission time. A full character transmission time is defined as the total time of the start, data, parity and stop bits. If the Receive instruction is configured to start a message on receiving a break condition, any characters received after the break condition are placed in the message buffer. Any characters received before the break condition are ignored.
   Typically, you use break detection as a start condition only when a protocol requires it.
   Setup: il = 0, sc = 0, bk = 1, SMW90/SMW190/SMW409 = don't care, SMB88/SMB188/SMB407 = don't care
5. *Break and a start character:* The Receive instruction can be configured to start receiving characters after receiving a break condition, and then a specific start character, in that sequence. After the break condition, the receive message function looks for the specified start character. If any character but the start character is received, the receive message function restarts the search for a break condition. All characters received before the break condition has been satisfied and before the start character has been received are ignored. The start character is placed in the message buffer along with all subsequent characters.
   Setup: il = 0, sc = 1, bk = 1, SMW90/SMW190/SMW409 = don't care, SMB88/SMB188/SMB407 = start character
6. *Any character:* The Receive instruction can be configured to immediately start receiving any and all characters and placing them in the message buffer. This is a special case of the idle line detection. In this case the idle line time (SMW90, SMW190 or SMW409) is set to zero. This forces the Receive instruction to begin receiving characters immediately upon execution.
   Setup: il = 1, sc = 0, bk = 0, SMW90/SMW190/SMW409 = 0, SMB88/SMB188/SMB407 = don't care
   Starting a message on any character allows the message timer to be used to time out the receiving of a message. This is useful in cases where Freeport is used to implement the master or host portion of a protocol and there is a need to time out if no response is received from a slave device within a specified amount of time. The message timer starts when the Receive instruction executes because the idle line time was set to zero. The message timer times out and terminates the receive message function if no other end condition is satisfied.
   Setup: il = 1, sc = 0, bk = 0, SMW90/SMW190/SMW409 = 0, SMB88/SMB188/SMB407 = don't care, c/m = 1, tmr = 1, SMW92/SMW192/SMW411 = message timeout in milliseconds

The Receive instruction supports several ways to terminate a message. The message can be terminated on one or a combination of the following:

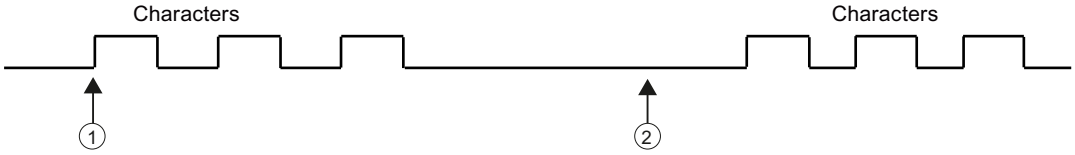
1. *End character detection:* The end character is any character which is used to denote the end of the message. After finding the start condition, the Receive instruction checks each character received to see if it matches the end character. When the end character is received, it is placed in the message buffer and the receive is terminated.
   Typically, you use end character detection with ASCII protocols where every message ends with a specific character. You can use end character detection in combination with the intercharacter timer, the message timer or the maximum character count to terminate a message.
   Setup: ec = 1, SMB89/SMB189/SMB408 = end character
2. *Intercharacter timer:* The intercharacter time is the time measured from the end of one character (the stop bit) to the end of the next character (the stop bit). If the time between characters (including the second character) exceeds the number of milliseconds specified in SMW92 or SMW192 or SMW411, the receive message function is terminated. The intercharacter timer is restarted on each character received. See the following figure.
   You can use the intercharacter timer to terminate a message for protocols which do not have a specific end-of-message character. This timer must be set to a value greater than one character time at the selected baud rate since this timer always includes the time to receive one entire character (start bit, data bits, parity and stop bits).
   You can use the intercharacter timer in combination with the end character detection and the maximum character count to terminate a message.
   Setup: c/m = 0, tmr = 1, SMW92/SMW192/SMW411 = timeout in milliseconds



① Restarts the intercharacter timer

② The intercharacter timer expires: Terminates the message and generates the Receive message interrupt

3. *Message timer:* The message timer terminates a message at a specified time after the start of the message. The message timer starts as soon as the start condition(s) for the receive message function have been met. The message timer expires when the number of milliseconds specified in SMW92, SMW192 or SMW411 has passed. See the following figure.
   Typically, you use a message timer when the communications devices cannot guarantee that there will not be time gaps between characters or when operating over modems. For modems, you can use a message timer to specify a maximum time allowed to receive the message after the message has started. A typical value for a message timer would be about 1.5 times the time required to receive the longest possible message at the selected baud rate. You can use the message timer in combination with the end character detection and the maximum character count to terminate a message.
   Setup: c/m = 1, tmr = 1, SMW92/SMW192/SMW411 = timeout in milliseconds



① Start of the message: Starts the message timer

② The message timer expires: Terminates the message and generates the Receive message interrupt

4. *Maximum character count:* The Receive instruction must be told the maximum number of characters to receive (SMB94 or SMB194 or SMB413). When this value is met or exceeded, the receive message function is terminated. The Receive instruction requires that the user specify a maximum character count even if this is not specifically used as a terminating condition. This is because the Receive instruction needs to know the maximum size of the receive message so that user data placed after the message buffer is not overwritten.
   The maximum character count can be used to terminate messages for protocols where the message length is known and always the same. The maximum character count is always used in combination with the end character detection, intercharacter timer, or message timer.

5. *Parity errors:* The Receive instruction automatically terminates when the hardware signals a parity, framing, or overrun error; or if a break condition is detected after the start of a message. Parity errors occur only if parity is enabled in SMB30 or SMB130 or SMB31. Framing errors occur if the stop bit is not correct. Overrun errors occur if characters come in to quickly for the hardware to handle. A break condition terminates a message because it resembles a parity or framing error to the hardware. There is no way to disable this function.

6. *User termination:* The user program can terminate a receive message function by executing another Receive instruction with the enable bit (EN) in SMB87 or SMB187 or SMB406 set to zero. This immediately terminates the receive message function.

Using character interrupt control to receive data

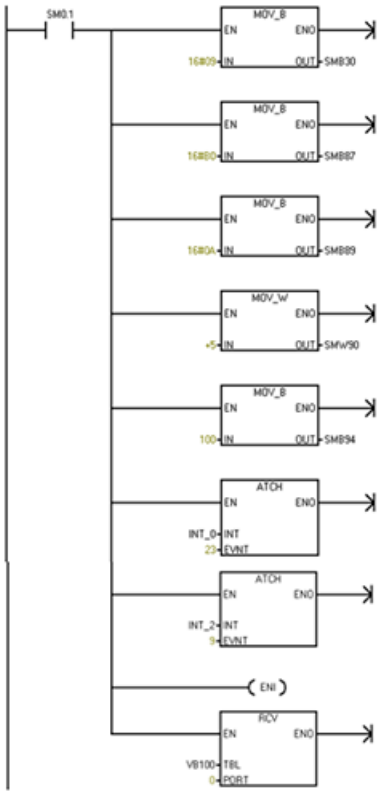
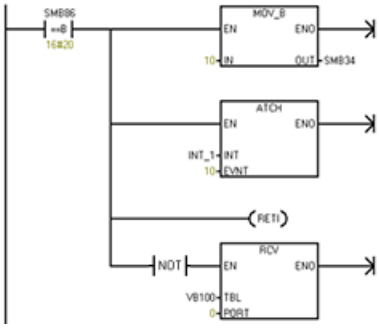
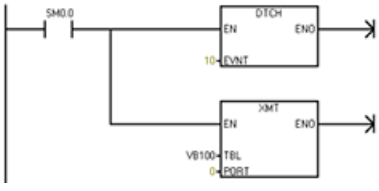
To allow complete flexibility in protocol support, you can also receive data using character interrupt control. Each character received generates an interrupt. The received character is placed in SMB2, and the parity status (if enabled) is placed in SM3.0 just prior to execution of the interrupt routine attached to the receive character event. SMB2 is the Freeport receive character buffer. Each character received while in Freeport mode is placed in this location for easy access from the user program. SMB3 is used for Freeport mode and contains a parity error bit that is turned on when a parity, framing, overrun, or break error is detected on a received character. All other bits of the byte are reserved. Use the parity bit either to discard the message or to generate a negative acknowledgement to the message.

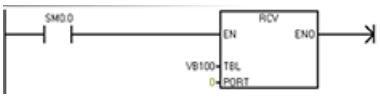
When the character interrupt is used at high baud rates (38.4 Kbps to 115.2 Kbps), the time between interrupts is very short. For example, the character interrupt for 38.4 Kbps is 260 microseconds, for 57.6 Kbps is 173 microseconds, and for 115.2 Kbps is 86 microseconds. Ensure that you keep the interrupt routines very short to avoid missing characters, or else use the Receive instruction.

Note

SMB2 and SMB3 are shared between Port 0, Port 1 and Port 2. When the reception of a character on Port 0 results in the execution of the interrupt routine attached to that event (interrupt event 8), SMB2 contains the character received on Port 0, and SMB3 contains the parity status of that character. When the reception of a character on Port 1 results in the execution of the interrupt routine attached to that event (interrupt event 25), SMB2 contains the character received on Port 1 and SMB3 contains the parity status of that character. When the reception of a character on Port 2 results in the execution of the interrupt routine attached to that event (interrupt event 47), SMB0 contains the character received on Port 1 and SMB3 contains the parity status of that character.

Example: Transmit and Receive instructions

MAIN	Network 1	Network 1 //This program receives a string of characters until a line feed character is received. The message is then transmitted back to the sender.	
		LD SM0.1 MOVB 16#09, SMB30	On the first scan: 1. Initialize Freeport: - Select 9600 baud. - Select 8 data bits. - Select no parity.
		MOVB 16#B0, SMB87	2. Initialize RCV message control byte: - RCV enabled. - Detect end of message character. - Detect idle line condition as the message start condition.
		MOVB 16#0A, SMB89	3. Set end of message character to hex 0A (line feed).
		MOVW +5, SMW90	4. Set idle line timeout to 5 ms.
		MOVB 100, SMB94	5. Set maximum number of characters to 100.
		ATCH INT_0, 23	6. Attach interrupt 0 to the Receive Complete event.
		ATCH INT_2, 9	7. Attach interrupt 2 to the Transmit Complete event.
		ENI	8. Enable user interrupts.
		RCV VB100, 0	9. Enable receive box with buffer at VB100.
INT 0	Network 1	Network 1	
		LDB= SMB86, 16#20 MOVB 10, SMB34 ATCH INT_1, 10 CRETI NOT RCV VB100, 0	Receive complete interrupt routine: 1. If receive status shows receive of end character, then attach a 10 ms timer to trigger a transmit and return. 2. If the receive completed for any other reason, then start a new receive.
INT 1	Network 1	Network 1	
		LD SM0.0 DTCH 10 XMT VB100, 0	10-ms Timer interrupt: 1. Detach timer interrupt. 2. Transmit message back to user on port.

INT 2	Network 1	Network 1	
		LD SM0.0 RCV VB100, 0	Transmit Complete interrupt: Enable another receive.

10.3.3 Get port address and set port address (PPI protocol on RS485/RS232)

You can use the GET_ADDR and SET_ADDR instructions to read and set the PPI network address of the selected port.

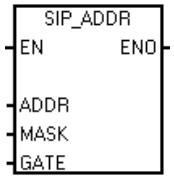
LAD / FBD	STL	Description
GET_ADDR EN ENO ADDR PORT	**GPA ADDR, PORT**	The GET_ADDR instruction reads the station address of the CPU port specified in PORT and places the value in the address specified in ADDR.
SET_ADDR EN ENO ADDR PORT	**SPA ADDR, PORT**	The SET_ADDR instruction sets the port station address (PORT) to the value specified in ADDR. The new address is not saved permanently. After a power cycle, the affected port returns to the network address downloaded in the system block.

Non-fatal error conditions with ENO = 0	SM bits affected
• 006H Indirect address • 0004H Attempted to perform a SET_ADDR instruction in an interrupt routine • 0090H Port number is invalid • 0091H Port address is invalid	None

Input / output	Data type	Operand
ADDR	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant (A constant value is valid only for the Set Port Address instruction.)
PORT	BYTE	Constant: 0, 1 or 2 Note: The two available ports are as follows: • Integrated RS485 port (Port 0), • CM01 Signal Board (SB) RS232/RS485 port (Port 1) • CM01 Signal Board (SB) RS232/RS485 port (Port 2)

10.3.4 Get IP address and set IP address (Ethernet)

You can use the GIP_ADDR and SIP_ADDR instructions to read and set the Ethernet IP address, the subnet mask, and gateway address for the Ethernet port.

LAD / FBD	STL	Description
	GIP ADDR, MASK, GATE	The GIP_ADDR instruction copies the CPU's IP address into ADDR, the CPU's subnet mask into MASK, and the CPU's gateway into GATE.
	**SIP ADDR, MASK, GATE**	The SIP_ADDR instruction sets the CPU's IP address to the value found in ADDR, the CPU's subnet mask to the value found in MASK, and the CPU's gateway to the value found in GATE.

Non-fatal errors with ENO = 0	SM bits affected
• 006H Indirect address • 0004H Attempted to execute a SIP_ADDR instruction in an interrupt routine • IP address cannot be changed (see following note) • IP address is invalid for the current subnet	None

Input / output	Data type	Operand
ADDR	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant (For SIP_ADDR only)
MASK	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant (For SIP_ADDR only)
GATE	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant (For SIP_ADDR only)

Note

To use the SIP_ADDR instruction, do not select the **"IP address data is fixed to the values below and cannot be changed by other means"** option for the Ethernet Port in the Communication section of the System Block.

Execution of the SIP_ADDR instruction causes the CPU to store the IP address, subnet mask, and gateway values in persistent memory.

Example

Note that STEP 7-Micro/WIN SMART displays the outputs for the GIP_ADDR instruction, ADDR, MASK, and GATE, as string values. For the SIP_ADDR instruction, however, you provide the ADDR, MASK, and GATE inputs as hexadecimal values. For the SIP_ADDR input values, think of each octet of the IP address, MASK and GATE as a hexadecimal number.

For the SIP_ADDR instruction, consider the octets of the IP address "192.168.2.150":

Octet decimal value	Hexadecimal value
192	C0
168	A8
2	02
150	96

You would use the combined hexadecimal values of the octets to form the ADDR input to the SIP_ADDR instruction: 16#C0A80296. (You could convert this number to a decimal value, but the hexadecimal value is representative of the values of the octets.)

Similarly, consider the octets of the subnet mask "255.255.255.0":

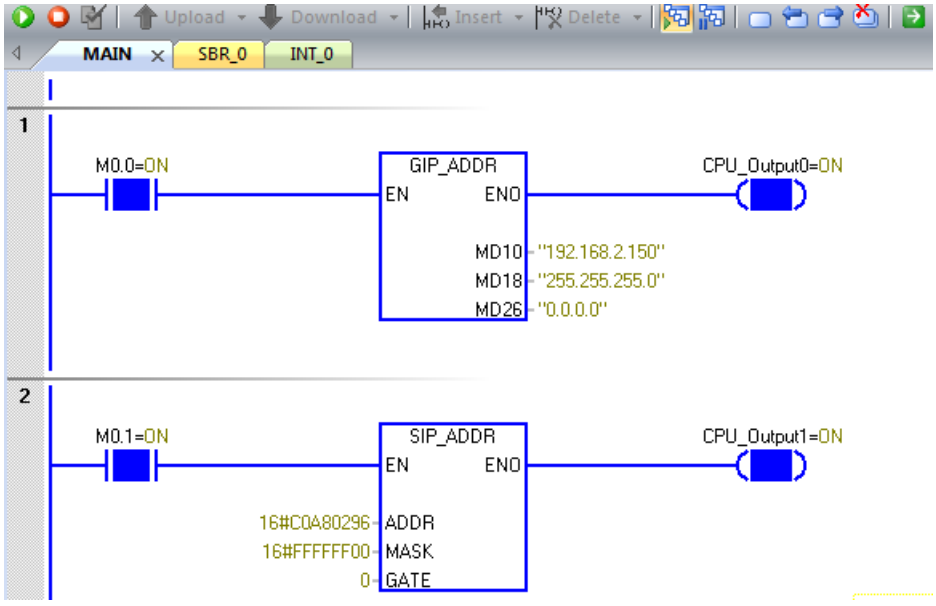
Octet decimal value	Hexadecimal value
255	FF
255	FF
255	FF
0	00

You would use the combined hexadecimal values of the octets to form the MASK input to the SIP_ADDR instruction: 16#FFFFFF00. You could also use the decimal equivalent, but not a string representation.

The following program status display shows two networks:

- Network 1: The GIP_ADDR reads the IP address of 192.168.2.150 with a subnet mask of 255.255.255.0.
- Network 2: The SIP_ADDR sets the IP address to 192.168.2.150 (16#C0A80296) and sets the subnet mask to 255.255.255.0 (16#FFFFFF00).

Note that the default gateway is 0.



10.3.5 Open user communication

The Open User Communication (OUC) instructions give your program a way to communicate over an Ethernet network to another Ethernet capable device. The other Ethernet device can be another S7-200 SMART CPU or another third party device that supports either UDP, TCP, or ISO-on-TCP protocol. Your program controls all aspects of the communication from selecting the protocol, initiating the connection, sending data, receiving data, and terminating the connection.

10.3.5.1 OUC instructions

There are four Open User Communications (OUC) instructions to control the communication process:

- TCON opens the UDP, TCP, or ISO-on-TCP (RFC 1006) connection between the S7-200 SMART CPU and the remote device.
- TSEND and TRCV send and receive data.
- TDCON closes the connection.

Table 10-7 OUC instructions

LAD/FBD	STL	Description
TCON EN ENO TABLE	**TCON table**	TCON initiates a UDP, TCP, or ISO-on-TCP communications connection from the CPU to a communication partner.
TSEND EN ENO TABLE	**TSEND table**	TSEND sends data to another device.
TRECV EN ENO TABLE	**TRECV table**	TRECV retrieves data over an existing communication connection.
TDCON EN ENO TABLE	**TDCON table**	TDCON terminates a UDP, TCP, or ISO-on-TCP communications connection.

The OUC instructions maintain information about the connections so that your program does not need to permanently allocate V memory for the OUC Tables. The data in the tables must be kept constant while the OUC instruction is active.

The OUC instructions require additional communication background time when they are processing/active/busy and also when they are just maintaining the connection to the other device. The amount of communication background time required depends on the number of OUC instructions that are active/busy, how often the OUC instructions are executed, and the number of connections that are currently open. You should adjust the communication background time to a higher value if the communication performance is slow. Refer to "Configuring communication" (Page 183) for further information.

All of the OUC instructions use a table to store the parameters for the instructions. The content of the tables for each instruction is described below.

The S7-200 SMART CPU uses the input table parameters to determine the instance of the OUC instructions. The table parameters must be kept the same during an operation so that the S7-200 SMART CPU knows that the particular instruction (instance) is the same as during the previous scans.

Note

Siemens also offers the Open User Communication (OUC) library instructions for your convenience. The OUC Library instructions build the tables for you based upon the inputs to the library instructions. The Library instructions also retrieve the response information from the tables and provide this information on the outputs of the library instructions. Refer to "Open user communication library" (Page 616) for further information.

Table 10-8 Valid operands for the OUC instructions

Inputs/Outputs	Data Type	Operands
TABLE	BYTE	IB, QB, VB, MB, SMB, SB, *VD, *LD, *AC

Error conditions that set ENO = 0:

- 0006 (indirect address)
- If the function returns an error and sets the E bit of table status byte (see the figure below)

TCON instruction

You use the TCON instruction to set up and establish a communication connection. Once the CPU establishes the connection, it is automatically maintained and monitored by the CPU. The TCON instruction has only one parameter which is the address of the TCON table. The TCON table contains the connection parameters. There are two formats for the TCON table based upon the protocol selected for the connection. UDP and TCP share a common table format. ISO-on-TCP has a special TCON table format. Refer to the TCON instruction tables below for further information.

Set the REQ bit in the table to TRUE to initiate a connection. The CPU ignores the REQ bit while the TCON instruction is active and the connection is initializing and the Active bit is TRUE. The TCON instruction set the Done bit when the CPU has established the connection. The Error bit is set if there is a problem with the connection parameters or if the CPU cannot establish a connection to the remote device. The Error Code describes the reason for the connection failure if the Error bit is set.

The TCON instruction is asynchronous and can take several scans to complete. The Active bit will be set while the connection operation is pending.

The TCON instruction creates either an active (client) connection or a passive (server) connection. The CPU initiates contact with the remote devices for an active connection. Passive connections cause the CPU to wait on the remote device to contact the CPU.

You can use the TCON instruction to determine the current status of a connection. If your program calls the TCON instruction with the REQ bit set to FALSE, the CPU reports the status of the connection:

- The instruction sets the Done bit (without Error) if the CPU establishes the connection and the connection is operational.
- The instruction sets the Active bit if the connection is still in the process of being connected.
- The instruction sets the Done and Error bits if no connection can be established. The Error Code gives the reason for the connection failure.

The REQ bit in the table is level-triggered. It is recommended that you put a positive edge trigger on the REQ input to initiate a connection so that the CPU only requests the connection establishment one time.

During the connection process (the call to the TCON instruction), your program assigns a Connection ID to the given connection. The Connection ID is a user-selected, 16-bit value passed into the TCON instruction. The Connection ID can be any number 0 to 65534 inclusive. The CPU does not allow the Connection ID to be 65535 (0xFFFF). The Connection ID value is an input to all of the OUC instructions to identify the connection to be utilized for the given operation.

You can select your own Connection ID, allowing you to use a number that is logical for your situation. For example, you can use part of your IP address as your connection ID. You can name your connection to IP address 192.168.2.10, connection ID 10.

Note that the S7-200 SMART does not automatically attempt to reconnect to a device after the connection has been closed. After a connection has been disconnected, your program must execute another TCON instruction to reconnect the device. This is true for both active and passive connections.

TCON instruction tables

The following tables contain the format and definitions for the TCON instruction. Refer to "OUC instruction error codes" (Page 296) for the error code listing. Refer to "Ports and TSAPs" (Page 493) for port number restrictions and further information:

- Status: The first byte of the table returns the status of the operation to the user. The OUC instructions ignore the value of the status byte as an input. The status byte is valid on the return of the instruction. These are the status bit definitions:
  - D = Done (Complete)
  - A = Active (In progress, in other words, Busy)
  - E = Error (Complete with error)
  - Error Code

  If there is an error, the Done and Error bits are both set. The error codes are listed in "OUC instruction error codes" (Page 296).
- REQ: You use the REQ bit to initiate a new operation. The REQ bit is a level-triggered value. Your program code must provide the one shot operation if required (a positive edge contact). If the operation is not busy, a REQ value of TRUE initiates a new operation. For example, if there is not currently a TSEND instruction in progress, a TRUE value in the REQ bit causes the program to initiate a new TSEND instruction operation.
- Connection ID: The Connection ID is a 16-bit value that you select to pass into the function. The range is 0 to 65534 (65535 is reserved). The Connection ID parameter is an input to the OUC instructions. The TSEND, TRECV, and TDCON instructions use the Connection ID that you select for the TCON instruction as a reference.

Table 10-9 Definition of TCON instruction TABLE parameter structure for UDP and TCP

Byte offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	D	A	E	Error code (5 bits)				
1	A/P [1]							REQ
2	Connection ID (2 bytes)							
3								
4	Connection Type [2]							

Byte offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
5	Remote							
6	IP							
7	Address [3]							
8								
9	Remote port [4]							
10								
11	Local port [5]							
12								

1 A/P: Active/Passive selection (1 = Active, 0 = Passive)

2 Connection Type: The Connection Type informs the TCON instruction of the desired type of connection: UDP = 19 and TCP = 11

3 Remote IP address: This is the IP address of the remote device in the case of an active connection. You should set the Remote IP Address to 0.0.0.0 for UDP connections. The IP address must be different than that of the local CPU and cannot be a multicast or broadcast address. Since the S7-200 SMART supports routing, the IP address may be on a different subnet than the local CPU.
If you set the IP address for a passive (server) connection, then the CPU only accepts a connection from the specified IP address. If you set the IP address to 0.0.0.0 for a passive connection, the CPU accepts a connection from any IP address.

4 Remote Port: This is the port number in the remote device. You do not use the remote port number for UDP or passive connections, and you should set the Remote Port to zero.

5 Local Port: This is the port number for the connection in the local CPU.

Table 10-10 Definition of TCON instruction TABLE parameter structure for ISO-on-TCP

Byte offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	D	A	E	Error code (5 bits)				
1	A/P [1]							REQ
2	Connection ID							
3	(2 bytes)							
4	Connection Type [2]							
5	Remote							
6	IP							
7	Address [3]							
8								
9	Remote TSAP [4]							
to	String of 2 to 16 characters (3 to 17 bytes)							
25								

Byte offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
26	Local TSAP [5]							
to	String of 2 to 16 characters (3 to 17 bytes)							
42								

[1] A/P: Active/Passive selection (1 = Active, 0 = Passive)

[2] Connection Type: The Connection Type informs the TCON instruction of the desired type of connection: ISO-on-TCP = 12

[3] Remote IP address: This is the IP address of the remote device in the case of an active connection. The IP address must be different than that of the local CPU and cannot be a multicast or broadcast address. Since the S7-200 SMART supports routing, the IP address may be on a different subnet than the local CPU.
If you set the IP address for a passive (server) connection, then the CPU only accepts a connection from the specified IP address. If you set the IP address to 0.0.0.0 for a passive connection, the CPU accepts a connection from any IP address.

[4] Remote TSAP: This is the Transport Service Access Point (TSAP) of the remote device. You use the remote TSAP for ISO-on-TCP connections only. The remote TSAP is a string of 2 to 16 ASCII characters.

[5] Local TSAP: This is the Transport Service Access Point (TSAP) for the connection in local CPU. You only use the local TSAP for ISO-on-TCP connections. The local TSAP is a string of 2 to 16 ASCII characters. If you use two characters, the TSAP must start with a hex "E0" character ($E0), followed by another hex character (for example, "$E0$01"). You cannot use the string "SIMATIC-".

TSEND

You use the TSEND instruction to send data over an existing communication connection. The TSEND table contains the connection parameters. There are two formats for the TSEND table based upon the protocol selected for the connection. TCP and ISO-on-TCP share a common table format. UDP has a special TSEND table format. Refer to the TSEND and TRECV instruction tables below for further information.

The TSEND instruction initiates sending the specified number of bytes when your program calls the TSEND instruction with the REQ bit set and the connection is not currently busy with some other operation.

The REQ bit is level-triggered. It is recommended that you put a positive edge trigger on the REQ input so that the CPU does not initiate unintended send operations. The CPU ignores the REQ bit while the TSEND is Active. The status bits and error code show the status of the TSEND for each call:

- Done without Error means the TSEND instruction completed with no errors.
- Active means that the TSEND instruction is still busy.
- Done with Error means there is a problem with the TSEND instruction. The Error Code contains the reason for the failure.

The Done/Active/Error status is shown for one call of the TSEND instruction after the send operation is complete. After that, the TSEND instruction responds with error code 24, which means no operation pending, if your program calls the instruction with REQ set to FALSE. If the REQ bit remains set, the TSEND instruction initiates another send operation.

The maximum amount of data that you can send in one message is 1024 bytes. Only one TSEND instruction can be active at a time on a given connection. The program copies the data from your send buffer in user memory to an internal buffer when the TSEND instruction

executes with REQ set, so you can change your send buffer after the TSEND instruction executes.

TRECV

You use the TRECV instruction to retrieve data that the CPU has received over an existing communication connection. You assign the receive area/buffer and the maximum length of the receive area so that there is no possibility of a buffer overrun. The TRECV table contains the parameters needed for the TRECV instruction. There are two formats for the TRECV table based upon the protocol selected for the connection. TCP and ISO-on-TCP share a common table format. UDP has a special TRECV table format. Refer to the TSEND and TRECV tables below for further information.

The TRECV instruction does not have a REQ bit. After the first execution of the TRECV instruction, the status bits show the instruction as Active. All subsequent calls to the TRECV instruction show an Active status if no data has been received by the CPU for this connection.

After a successful receipt of data, the instruction sets the Done bit in the status byte of the table, and the returned Data Length value is the actual number of bytes received. The TRECV instruction copies the received data from an internal buffer to the your receive buffer only when the TRECV instruction executes and the Done bit is set to TRUE.

The maximum amount of data that you can receive in one message is 1024 bytes. Because TCP acts as a "streaming" protocol, the program can collect multiple messages in one receive message if the TRECV instruction is not called frequently. The UDP and ISO-on-TCP protocols guarantee that each message is delineated as a separate message.

For example, let us suppose that there is a TCP client that sends four 20-byte messages to the S7-200 SMART in rapid succession, and your program is not calling the TRECV instruction. If your program calls the TRECV instruction after all four messages have been accepted by the CPU, the program sees this as one receive message of 80 bytes. Your program is responsible for calling the TRECV instruction as often as needed to receive each message as it is sent.

Assuming the same client and the same messages as in the above example, ISO-on-TCP and UDP delivers the four messages during four subsequent calls to the TRECV instruction. These protocols delineate the messages and keep them separate in the CPU until your program calls the TRECV instruction to retrieve them.

If the CPU receives more bytes than will fit into the user buffer, the TRECV instruction copies in the maximum number of bytes allowed (Data Length in the table) and discards the rest of the received bytes. In this situation, the TRECV instruction completes with an error to tell the user that bytes were discarded.

TSEND and TRECV instruction tables

The following tables contain the format and definitions for the TSEND and TRECV instructions. Refer to "OUC instruction error codes" (Page 296) for the error code listing. Refer to "Ports and TSAPs" (Page 493) for port number restrictions and further information:

- Status: The first byte of the table returns the status of the operation to the user. The OUC instructions ignore the value of the status byte as an input. The status byte is valid on the return of the instruction. These are the status bit definitions:
  - D = Done (Complete)
  - A = Active (In progress, in other words, Busy)
  - E = Error (Complete with error)
  - Error Code

  If there is an error, the Done and Error bits are both set. The error codes are listed in "OUC instruction error codes" (Page 296).
- REQ: You use the REQ bit to initiate a new operation. The REQ bit is a level-triggered value. Your program code must provide the one shot operation if required (a positive edge contact). If the operation is not busy, a REQ value of TRUE initiates a new operation. For example, if there is not currently a TSEND instruction in progress, a TRUE value in the REQ bit causes the program to initiate a new TSEND instruction operation.
- Connection ID: The Connection ID is a 16-bit value that you select to pass into the function. The range is 0 to 65534 (65535 is reserved). The Connection ID parameter is an input to the OUC instructions. The TSEND, TRECV, and TDCON instructions use the Connection ID that you select for the TCON instruction as a reference.

Table 10-11 Definition of TSEND and TRECV instruction TABLE parameter structure for TCP and ISO-on-TCP

Byte offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	D	A	E	Error code (5 bits)				
1								REQ [1]
2	Connection ID (2 bytes)							
3								
4	Data Length [2]							
5								

Byte offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
6	Data Pointer [3]							
7								
8								
9								

[1] REQ: You set the REQ bit to TRUE to initiate a new TSEND instruction operation. The TRECV instruction ignores the REQ status bit. The REQ bit is only used for the TSEND instruction.
For the TRECV instruction, the Done bit means that the CPU received data (New Data Ready) and the Data_Length value returns the actual number of bytes received. If there is no data available when called, the TRECV instruction returns with the Active flag set and a Data_Length value of zero. If the number of received bytes exceeds the size of the receive buffer (data length input), the program copies the maximum number of bytes into the buffer and returns an error to the TRECV instruction.

[2] Data Length: The Data Length in the TRECV instruction table is both an input and output parameter. The input value is the maximum size of the receive buffer. The output value is the number of bytes actually received.
The Data Length is an input value only for the TSEND instruction.

[3] Data Pointer: An S7-200 SMART pointer to the data in the local CPU.

Table 10-12 Definition of TSEND and TRECV instruction TABLE parameter structure for UDP

Byte offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	D	A	E	Error code (5 bits)				
1								REQ [1]
2	Connection ID (2 bytes)							
3								
4	Data Length [2]							
5								
6	Data Pointer [3]							
7								
8								
9								
10	Remote IP Address [4]							
11								
12								
13								

Byte offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
14	Remote port [5]							
15								

[1] REQ: You set the REQ bit to TRUE to initiate a new TSEND instruction operation. The TRECV instruction ignores the REQ status bit. The REQ bit is only used for the TSEND instruction.
For the TRECV instruction, the Done bit means that the CPU received data (New Data Ready) and the Data_Length value returns the actual number of bytes received. If there is no data available when called, the TRECV instruction returns with the Active flag set and a Data_Length value of zero. If the number of received bytes exceeds the size of the receive buffer (data length input), the program copies the maximum number of bytes into the buffer and returns an error to the TRECV instruction.

[2] Data Length: The Data Length in the TRECV instruction structures is both an input and output parameter. The input value is the maximum size of the receive buffer. The output value is the number of bytes actually received.
The Data Length is an input value only for the TSEND instruction.

[3] Data Pointer to the data area: An S7-200 SMART pointer to the data in the local CPU.

[4] Remote IP address: This is the IP address of the remote device for a TSEND instruction. The IP address must be different than that of the local CPU and cannot be a multicast or broadcast address. Since the S7-200 SMART supports routing, the IP address may be on a different subnet than the local CPU. (The IP address must be supplied for each UDP send operation.)
The IP address is a returned value for a UDP receive operation. The IP address is the address of the sender of UDP message.

[5] Remote Port: This is the port number in the remote device.
The remote port is a returned value for a UDP receive operation. The port is the port number of the sender of the UDP message.
UDP requires the remote port number for each TSEND instruction message.

TDCON

You use the TDCON instruction to terminate an existing communication connection. The instruction terminates the connection when the REQ bit is set. It is recommended that you put a positive edge trigger on the REQ input. If the your program calls the TDCON instruction and the connection is already disconnected, then the instruction responds with error code 24, which means no operation pending.

TDCON instruction table

The following table contains the format and definitions for the TDCON instruction. Refer to "OUC instruction error codes" (Page 296) for the error code listing. Refer to "Ports and TSAPs" (Page 493) for port number restrictions and further information:

- Status: The first byte of the table returns the status of the operation to the user. The OUC instructions ignore the value of the status byte as an input. The status byte is valid on the return of the instruction. These are the status bit definitions:
  - D = Done (Complete)
  - A = Active (In progress, in other words, Busy)
  - E = Error (Complete with error)
  - Error Code

  If there is an error, the Done and Error bits are both set. The error codes are listed in "OUC instruction error codes" (Page 296).
- REQ: You use the REQ bit to initiate a new operation. The REQ bit is a level-triggered value. Your program code must provide the one shot operation if required (a positive edge contact). If the operation is not busy, a REQ value of TRUE initiates a new operation. For example, if there is not currently a TSEND instruction in progress, a TRUE value in the REQ bit causes the program to initiate a new TSEND instruction operation.
- Connection ID: The Connection ID is a 16-bit value that you select to pass into the function. The range is 0 to 65534 (65535 is reserved). The Connection ID parameter is an input to the OUC instructions. The TSEND, TRECV, and TDCON instructions use the Connection ID that you select for the TCON instruction as a reference.

Table 10-13 Definition of TDCON instruction TABLE parameter structure

Byte offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	D	A	E	Error code (5 bits)				
1								REQ
2	Connection ID (2 bytes)							
3								

10.3.5.2 OUC instruction error codes

The following table lists the Open User Communication (OUC) error codes:

Error code	Description	TCON	TSEND	TRECV	TDCON
0	No error	X	X	X	X
1	The data length parameter is greater than the maximum allowed (1024 bytes).		X	X	
2	The data buffer is not in I, Q, M, or V memory areas.		X	X	
3	The data buffer does not fit in the memory area.		X	X	
4	The table parameter does not fit into the memory area.	X	X	X	X
5	The connection is locked in another context. You are attempting to access the same connection in both the background (the Main) and in an interrupt routine at the same time.	X	X	X	X
6	A UDP IP address or port error		X		
7	An instance mismatch: The connection is busy with another instance or the input data does not match the data stored for the requested connection ID when the request was initiated.	X	X	X	X
8	The Connection ID does not exist because the connection has never been created, or the connection was terminated at your request (using the TDCON instruction).	X	X	X	X
9	A TCON operation is in progress with this Connection ID.		X	X	X
10	A TDCON operation is in progress with this Connection ID.	X	X	X	
11	A TSEND instruction is in progress with this Connection ID.		X		X
12	A temporary communication error has occurred. The connection cannot be started at this time. Try again later.	X	X	X	
13	The connection partner refused or actively dropped the connection (the partner issued a disconnect to this CPU).	X	X	X	
14	The connection partner cannot be reached (no answer to the connect request).	X	X	X	
15	The connection aborted due to inconsistencies. Disconnect and reconnect to correct the situation.	X	X	X	X
16	The Connection ID is already in use with a different IP address, port, or TSAP combination.	X			
17	No connection resource is available. All connections of the requested type (active/passive) are in use.	X			
18	The local or remote port number is reserved or the port number is already in use for another server (passive) connection.	X			

Error code	Description	TCON	TSEND	TRECV	TDCON
19	One of the following IP address errors have occurred: • The IP address is invalid (for example, address 0.0.0.0). • This IP address is the IP address of this CPU. • This CPU has IP address 0.0.0.0. • The IP address is a broadcast or multicast address.	X			
20	A local or remote TSAP error (ISO-on-TCP only)	X			
21	An invalid connection ID (65535 is reserved)	X			
22	An active/passive error (UDP only allows passive)	X			
23	The connection type is not one of the allowed types.	X			
24	There is no operation pending so there is no status to report.		X		X
25	The receive buffer is too small: The CPU received more bytes than the buffer length supports. The CPU discards the extra bytes.			X	
31	Unknown error	X	X	X	X

10.4 Compare

10.4.1 Compare number values

The compare instructions can compare two number values with the same data type. You can compare bytes, integers, double integers, and real numbers.

For **LAD and FBD**: When the comparison is TRUE, the compare instruction sets ON a contact (LAD network power flow), or output (FBD logic flow).

For **STL**: When the comparison is TRUE, the compare instructions can load, AND, or OR a 1 with the value on the top of the logic stack.

Types of comparison

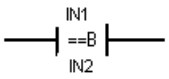
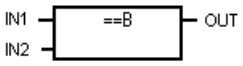
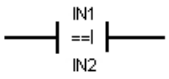
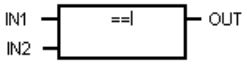
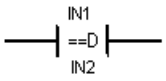
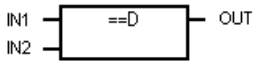
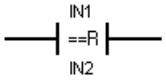
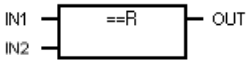
Six comparison types are available:

Comparison type	The output is TRUE only if
== (LAD/FBD) = (STL)	IN1 is equal to IN2
<>	IN1 is not equal to IN2
>=	IN1 is greater than or equal to IN2
<=	IN1 is less than or equal to IN2
>	IN1 is greater than IN2
<	IN1 is less than IN2

Selecting the data types to be compared

The data type identifier that you choose determines the required data type for the IN1 and IN2 parameters.

Data type identifier	Required IN1, IN2 data type
B	Unsigned byte
W	Signed word integer
D	Signed double word integer
R	Signed real

LAD contacts, FBD boxes	STL	Comparison result
 	**LDB= IN1, IN2** **OB= IN1, IN2** **AB= IN1, IN2**	Compare two unsigned byte values: The result is TRUE, if IN1 = IN2
 	**LDW= IN1, IN2** **OW= IN1, IN2** **AW= IN1, IN2**	Compare two signed integer values: The result is TRUE, if IN1 = IN2
 	**LDD= IN1, IN2** **OD= IN1, IN2** **AD= IN1, IN2**	Compare two signed double integer values: The result is TRUE, if IN1 = IN2
 	**LDR= IN1, IN2** **OR= IN1, IN2** **AR= IN1, IN2**	Compare two signed real values: The result is TRUE, if IN1 = IN2

Note

The following conditions cause a non-fatal error, set power flow to OFF (ENO bit = 0), and use value 0 as the result of the comparison

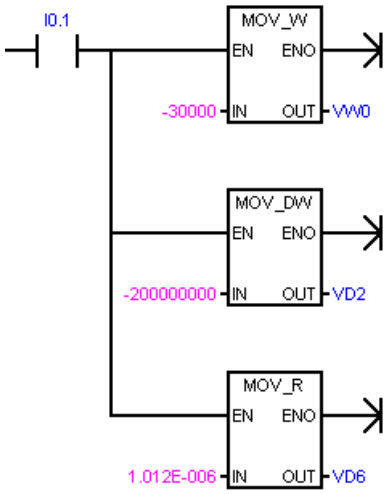
- Illegal indirect address is encountered (any compare instruction)
- Illegal real number (for example, NaN) is encountered for compare real instruction

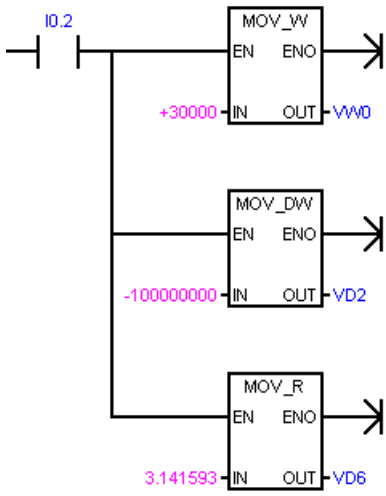
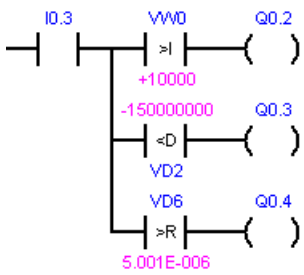
To prevent these conditions from occurring, ensure that you properly initialize pointers and values that contain real numbers before executing compare instructions that use these values.

Compare instructions are executed regardless of the state of power flow.

Input / output	Data type	Operand
IN1, IN2	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
OUT	BOOL	LAD: Power flow FBD: I, Q, V, M, SM, S, T, C, L, Logic Flow

Example compare values

LAD		STL
	Activate I0.1 to load V memory addresses with low values that make the comparisons FALSE and that set the status indicators OFF.	**Network 1** **LD I0.1** **MOVW -30000, VW0** **MOVD -200000000, VD2** **MOVR 1.012E-006, VD6**

	Activate I0.2 to load V memory addresses with high values that make the comparisons TRUE and that set the status indicators ON.	**Network 2** **LD I0.2** **MOVW +30000, VW0** **MOVD -100000000, VD2** **MOVR 3.141593, VD6**
	Activate I0.3 to perform comparisons. The Integer Word comparison tests to find if VW0 > +10000 is TRUE. You can also compare two values stored in variable memory like VW0 > VW100.	**Network 3** **LD I0.3** **LPS** **AW> VW0, +10000** **= Q0.2** **LRD** **AD< -150000000, VD2** **= Q0.3** **LPP** **AR> VD6, 5.001E-006** **= Q0.4**

See also

Constants (Page 120)

10.4.2 Compare character strings

The compare string instructions can compare two ASCII character strings.

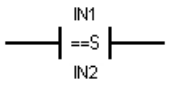
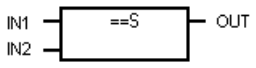
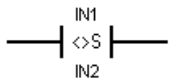
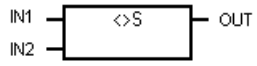
For **LAD** and **FBD:** When the comparison is TRUE, the compare instruction turns ON the contact (LAD), or output (FBD).

For **STL**: When the comparison is TRUE, the compare instruction loads, ANDs, or ORs a 1 with the value on the top of the logic stack.

Comparisons can be made between two variables, or between a constant and a variable. If a constant is used in a comparison, then it must be the top parameter (LAD contact / FBD box) or the first parameter (STL).

In the program editor, a constant string parameter assignment must begin and end with a double quote character. The maximum length of a constant string entry is 126 characters (bytes).

In contrast, a variable string is referenced by the byte address of the initial length byte with the character bytes stored the next byte addresses. A variable string has a maximum length of 254 characters (bytes) and can be initialized in the data block editor (with beginning and ending double quote character).

LAD contact FBD box	STL	Description
 	**LDS= IN1, IN2** **OS= IN1, IN2** **AS= IN1, IN2**	Compare two character strings of STRING data type: The result is TRUE, if string IN1 **equals** string IN2.
 	**LDS<> IN1, IN2** **OS<> IN1, IN2** **AS<> IN1, IN2**	Compare two character strings of STRING data type: The result is TRUE, if string IN1 **does not equal** string IN2.

Note

The following conditions cause a non-fatal error, set power flow to OFF (ENO bit = 0), and use value 0 as the result of the comparison:

- Illegal indirect address is encountered (any compare instruction)
- A variable string with a length greater than 254 characters is encountered (Compare String instruction)
- A variable string whose starting address and length are such that it will not fit in the specified memory area (Compare String instruction)

To prevent these conditions from occurring, ensure that you properly initialize pointers and memory locations that are intended to hold ASCII strings prior to executing compare instructions that use these values. Ensure that the buffer reserved for an ASCII string can reside completely within the specified memory area.

Compare instructions are executed regardless of the state of power flow.

Input / output	Data type	Operand
IN1	STRING	VB, LB, *VD, *LD, *AC, Constant string
IN2	STRING	VB, LB, *VD, *LD, *AC
OUT	BOOL	LAD: Power flow FBD: I, Q, V, M, SM, S, T, C, L, Logic Flow

Format of the STRING data type

A string variable is a sequence of characters, with each character stored as a byte. The first byte of the STRING data type defines the length of the string, which is the number of character bytes.

The diagram below shows the STRING data type stored as a variable in memory. The string can have a length of 0 to 254 characters. The maximum storage requirement for a variable string is 255 bytes (the length byte plus 254 characters).

Length	Character 1	Character 2	Character 3	Character 4	. . .	Character 254
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4		Byte 254

If a constant string parameter is entered directly in the program editor (126 characters maximum) or a variable string is initialized in the data block editor (254 characters maximum), the string assignment must begin and end with double quote characters.

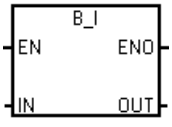
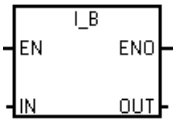
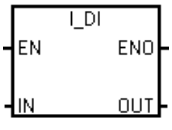
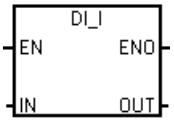
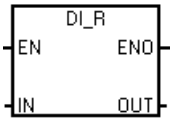
See also Constants (Page 120)

10.5 Convert

10.5.1 Standard conversion instructions

These instructions convert an input value IN to the assigned format and store the output value in the memory location assigned by OUT. For example, you can convert a double integer value to a real number. You can also convert between integer and BCD formats.

Standard conversions

LAD / FBD	STL	Description
	**BTI IN, OUT**	**Byte to integer**: Convert the byte value IN to an integer value and place the result at the address assigned to OUT. The byte is unsigned; therefore, there is no sign extension.
	**ITB IN, OUT**	**Integer to byte**: Convert the word value IN to a byte value and place the result at the address assigned to OUT. Values 0 to 255 are converted. All other values result in overflow and the output is not affected. **Note:** To change an integer to a real number, execute the Integer to Double Integer instruction and then the Double Integer to Real instruction.
	**ITD IN, OUT**	**Integer to double integer**: Convert the integer value IN to a double integer value and place the result at the address assigned to OUT. The sign is extended.
	**DTI IN, OUT**	**Double Integer to integer**: Convert the double integer value IN to an integer value and place the result at the address assigned to OUT. If the value that you convert is too large to be represented in the output, then the overflow bit is set and the output is not affected.
	**DTR IN, OUT**	**Double integer to real**: Convert a 32-bit, signed integer IN into a 32-bit real number and place the result at the address assigned to OUT.

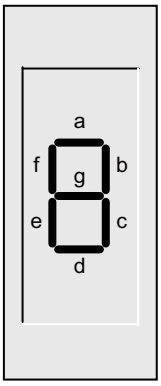
LAD / FBD	STL	Description
BCD_I EN ENO IN OUT I_BCD EN ENO IN OUT	**BCDI OUT** **IBCD OUT**	**BCD to Integer**: Convert the binary-coded decimal WORD data type value IN to an integer WORD data type value and load the result in the address assigned to OUT. The valid range for IN is 0 to 9999 BCD. **Integer to BCD**: Convert the input integer WORD data type value IN to a binary-coded decimal WORD data type and load the result at the address assigned to OUT. The valid range for IN is 0 to 9999 integer. For STL, the IN and OUT parameters use the same address.
ROUND EN ENO IN OUT TRUNC EN ENO IN OUT	**ROUND IN, OUT** **TRUNC IN, OUT**	**Round**: Convert the 32-bit real-number value IN to a double integer value and place the rounded result at the address assigned to OUT. If the fraction portion is 0.5 or greater, the number is rounded up. **Truncate**: Convert the 32-bit real-number value IN into a double integer value and place the result at the address assigned to OUT. Only the whole number portion of the real number is converted, and the fraction is discarded. **Note:** If the value that you are converting is not a valid real number or is too large to be represented in the output, then the overflow bit is set and the output is not affected.
SEG EN ENO IN OUT	**SEG IN, OUT**	**SEG:** To illuminate the segments of a seven-segment display, the Segment instruction converts the character byte specified by IN to generate a bit pattern byte at the address assigned to OUT. The illuminated segments represent the character in the least significant digit of the input byte.
DF_R EN ENO IN OUT	**DFTR IN, OUT**	**Double float to real:** Convert the double float value IN to a real value and place the result at the address assigned to OUT. If the value that you convert is too large to be represented in the output, the overflow bit is set and the output is not affected.
BCD_DI EN ENO IN OUT DI_BCD EN ENO IN OUT	**BCDDI IN, OUT** **DIBCD IN, OUT**	**BCD to Double Integer**: Convert the binary-coded decimal DINT data type value IN to an double integer DINT data type value and load the result in the address assigned to OUT. The valid range for IN is from 0 to 99999999 BCD. **Double Integer to BCD:** Convert the input double integer DINT data type value IN to a binary-coded decimal DINT data type and load the result at the address assigned to OUT. The valid range for IN is from 0 to 99999999 integer.

Non-fatal error conditions with ENO = 0	SM bits affected
• 0006H Indirect address • SM1.1 Overflow • SM1.6 Invalid BCD	• SM1.1 Overflow • SM1.6 Invalid BCD

Input / output	Data type	Operand
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
	WORD (BCD_I, I_BCD), INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AIW, AC, *VD, *LD, *AC, Constant
	DINT	ID, QD, VD, MD, SMD, SD, LD, HC, AC, *VD, *LD, *AC, Constant
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC
	WORD (BCD_I, I_BCD)	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
	INT (B_I, DI_I)	IW, QW, VW, MW, SMW, SW, T, C, LW, AC,, AQW, *VD, *LD, *AC
	DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

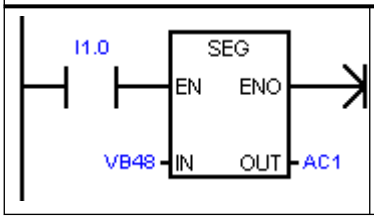
Coding for a seven-segment display

(IN) LSD	Segment Display	(OUT) - gfe dcba
0		0011 1111
1		0000 0110
2		0101 1011
3		0100 1111
4		0110 0110
5		0110 1101
6		0111 1101
7		0000 0111

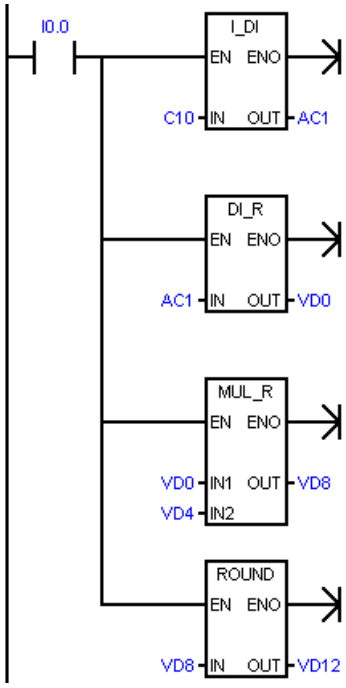
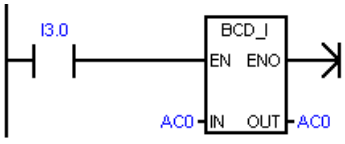


(IN) LSD	Segment Display	(OUT) - gfe dcba
8		0111 1111
9		0110 0111
A		0111 0111
B		0111 1100
C		0011 1001
D		0101 1110
E		0111 1001
F		0111 0001

Example: Using SEG to display the numeral 5 on a seven-segment display

LAD		STL
	05 SEG 6D VB48 AC1 (display character)	**Network 1** **LD I1.0** **SEG VB48, AC1**

Examples: I_DI, DI_R, and BCD_I

LAD		STL
	Convert inches to centimeters: 1. Load a counter value (inches) into AC1 (ex. C10=101). 2. Convert the value to a real number (ex. VD0=101.0). 3. Multiply by 2.54 to convert to centimeters (ex. VD4=2.54, VD8=256.54). 4. Convert the value back to an integer (ex. VD12=257).	**Network 1** **LD I0.0** **ITD C10, AC1** **DTR AC1, VD0** **MOVR VD0, VD8** ***R VD4, VD8** **ROUND VD8, VD12**
	Convert a BCD value to an integer (ex. AC0=1234, execute BCD_I, then AC0=04D2).	**Network 2** **LD I0.3** **BCDI AC0**

See also

Assigning a constant value for instructions (Page 120)

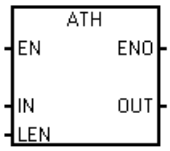
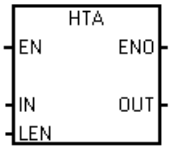
10.5.2 ASCII character array conversion

Converting from or to ASCII character byte arrays

The ASCII character array instructions use the BYTE data type for character input or output. An array of ASCII characters is referenced a sequence of byte addresses.

This is not the STRING data type, as no length byte is used. Use the ASCII string instructions to work with the variables of the STRING data type.

ASCII to Hex and Hex to ASCII

LAD / FBD	STL	Description
 	**ATH IN, OUT, LEN** **HTA IN, OUT, LEN**	ATH converts a number LEN of ASCII characters, starting at IN, to hexadecimal digits starting at OUT. The maximum number of ASCII characters that can be converted is 255 characters. HTA converts the hexadecimal digits, starting with the input byte IN, to ASCII characters starting at OUT. The number of hexadecimal digits to be converted is assigned by length LEN. The maximum number of ASCII characters or hexadecimal digits that can be converted is 255. Valid ASCII input characters are the alphanumeric characters 0 to 9 with a hexadecimal code value of 30 to 39, and uppercase characters A to F with a hex code value of 41 to 46.

Non-fatal error conditions with ENO = 0	SM bits affected
• 0006H Indirect address • 0091H Operand out of range • SM1.7 ATH: Illegal ASCII value	• SM1.7 ATH: Illegal ASCII value

Input / output	Data type	Operand
IN, OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC
LEN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant

Converting number values to the ASCII character representation (ITA, DTA, and RTA)

ASCII character output number format:

- Positive values are written to the output buffer without a sign.
- Negative values are written to the output buffer with a leading minus sign (-).
- Leading zeros to the left of the decimal point (except the digit adjacent to the decimal point) are suppressed.
- Values are right-justified in the output buffer.
- Real numbers: Values to the right of the decimal point are rounded to fit in the assigned number of digits to the right of the decimal point.
- Real numbers: The size of the output buffer must be a minimum of three bytes more than the number of digits to the right of the decimal point.

Integer to ASCII

LAD / FBD	STL	Description
ITA EN ENO IN OUT FMT	`ITA IN, OUT, FMT`	The Integer to ASCII instruction converts the integer value IN to an array of ASCII characters. The format parameter FMT assigns the conversion precision to the right of the decimal, and whether the decimal point is to be shown as a comma or a period. The resulting conversion is placed in 8 consecutive bytes beginning with the address assigned by OUT.

Non-fatal error conditions with ENO = 0	SM bits affected
• 0006H Indirect address • 0091H Operand out of range • FMT bit is not zero for 4 most significant bits of the FMT byte • nnn > 5	None

Input / output	Data type	Operand
IN	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
FMT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC

The size of the output buffer is always 8 bytes. The number of digits to the right of the decimal point in the output buffer is assigned by the nnn field. The valid range of the nnn field is 0 to 5. If you assign 0 digits to the right of the decimal point, then the value is converted with no decimal point. For values of nnn greater than 5, the output buffer is filled with ASCII space characters. The c bit specifies the use of either a comma (c=1) or a decimal point (c=0) as the separator between whole number and fraction. The most significant 4 bits must always be zero.

The following figure shows examples of values that are formatted using a decimal point (c=0) with three digits to the right of the decimal point (nnn=011).

FMT operand for the integer to ASCII (ITA) instruction

FMT

MSB							LSB
7	6	5	4	3	2	1	0
0	0	0	0	c	n	n	n

c = comma (1) or decimal point (0)
nnn= digits to right of decimal point

	Out	Out +1	Out +2	Out +3	Out +4	Out +5	Out +6	Out +7
in = 12				0	.	0	1	2
in = -123			-	0	.	1	2	3
in = 1234				1	.	2	3	4
in = -12345		-	1	2	.	3	4	5

Double integer to ASCII

LAD / FBD	STL	Description
DTA EN ENO IN OUT FMT	**DTA IN, OUT, FMT**	The **Double Integer to ASCII** instruction converts a double word IN to an array of ASCII characters. The format parameter FMT specifies the conversion precision to the right of the decimal. The resulting conversion is placed in 12 consecutive bytes beginning with OUT.

Non-fatal error conditions with ENO = 0	SM bits affected
• 0006H Invalid indirect address • 0091H Operand out of range • FMT bit is not zero for 4 most significant bits, of the FMT byte • nnn > 5	• None

Input / output	Data type	Operand
IN	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
FMT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC

The size of the output buffer is always 12 bytes. The number of digits to the right of the decimal point in the output buffer is assigned by the nnn field. The valid range of the nnn field is 0 to 5. If you assign 0 digits to the right of the decimal point, then the value is converted with no decimal point. For values of nnn bigger than 5, the output buffer is filled with ASCII spaces. The c bit specifies the use of either a comma (c=1) or a decimal point (c=0) as the separator between whole number and fraction. The most significant 4 bits must always be zero.

The following figure shows examples of values that are formatted using a decimal point (c=0) with four digits to the right of the decimal point (nnn=100).

FMT operand for the double integer to ASCII (DTA) instruction

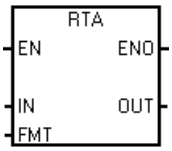
FMT

MSB							LSB
7	6	5	4	3	2	1	0
0	0	0	0	c	n	n	n

c = comma (1) or decimal point (0)
nnn= digits to right of decimal point

	Out	Out +1	Out +2	Out +3	Out +4	Out +5	Out +6	Out +7	Out +8	Out +9	Out +10	Out +11
in = 12						-	0	.	0	0	1	2
in = 1234567					1	2	3	.	4	5	6	7

Real to ASCII

LAD / FBD	STL	Description
	**RTA IN, OUT, FMT**	The **Real to ASCII** instruction converts a real-number value IN to ASCII characters. The format parameter FMT specifies the conversion precision to the right of the decimal, whether the decimal point is shown as a comma or a period, and the output buffer size. The resulting conversion is placed in an output buffer beginning with OUT.

Non-fatal error conditions with ENO = 0	SM bits affected
• 0006H Invalid indirect address • 0091H Operand out of range • nnn > 5 • ssss < 3 • ssss < number of characters in OUT	None

Input / output	Data type	Operand
IN	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
FMT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC

The number (or length) of the resulting ASCII characters is the size of the output buffer and can be assigned from 3 to 15 bytes or characters.

The real-number format supports a maximum of 7 significant digits. Attempting to display more than 7 significant digits produces a rounding error.

The following figure describes the format operand (FMT) for the RTA instruction. The size of the output buffer is assigned by the ssss field. A size of 0, 1, or 2 bytes is not valid. The number of digits to the right of the decimal point in the output buffer is assigned by the nnn field. The valid range of the nnn field is 0 to 5. If you assign 0 digits to the right of the decimal point, then the value is converted without a decimal point. The output buffer is filled with ASCII spaces for values of nnn greater than 5 or when the assigned output buffer is too small to store the converted value. The c bit specifies the use of either a comma (c=1) or a decimal point (c=0) as the separator between whole number and fraction.

The following figure also shows examples of values that are formatted using a decimal point (c=0) with one digit to the right of the decimal point (nnn=001) and a buffer size of six bytes (ssss=0110).

FMT Operand for the Real to ASCII (RTA) instruction

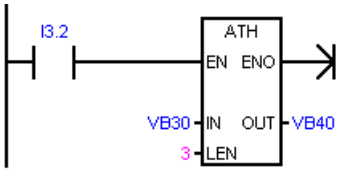
FMT

MSB							LSB
7	6	5	4	3	2	1	0
s	s	s	s	c	n	n	n

ssss = size of the output buffer
c =comma (1) or decimal point (0)
nnn= digits to right of decimal point

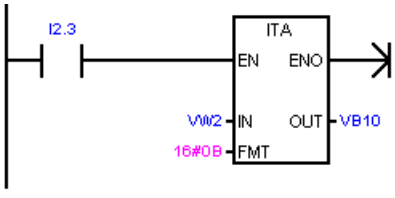
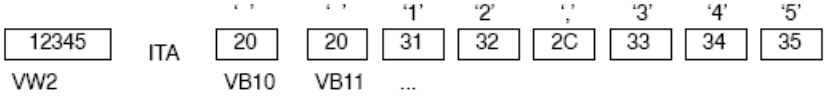
	Out	Out +1	Out +2	Out +3	Out +4	Out +5
in = 1234.5	1	2	3	4	.	5
in = -0.0004				0	.	0
in = -3.67526			-	3	.	7
in = 1.95				2	.	0

Example: ASCII to Hexadecimal

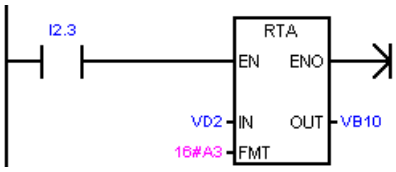
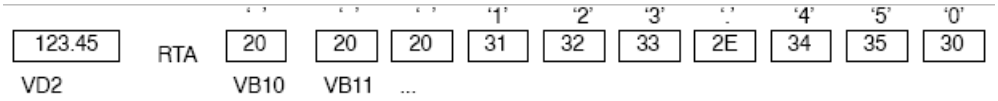
LAD	STL
	**Network 1** **LD I3.2** **ATH VB30, VB40, 3**

[1] The "x" indicates that the "nibble" (half of a byte) is unchanged.

Example: Integer to ASCII

LAD		STL
	Convert the integer value at VW2 to 8 ASCII characters starting at VB10, using a format of 16#0B (a comma for the decimal point, followed by 3 digits).	**Network 1** **LD I2.3** **ITA VW2, VB10, 16#0B**
		

Example: Real to ASCII

LAD		STL
	Convert the real value at VD2 to 10 ASCII characters starting at VB10, using a format of 16#A3 (a period for the decimal point, followed by 3 digits).	**Network 1** **LD I2.3** **RTA VD2, VB10, 16#A3**
		

See also

Assigning a constant value for instructions (Page 120)

10.5.3 Number value to ASCII string conversion

Format of the STRING data type

A string variable is a sequence of characters, with each character stored as a byte. The first byte of the STRING data type defines the length of the string, which is the number of character bytes.

The diagram below shows the STRING data type stored as a variable in memory. The string can have a length of 0 to 254 characters. The maximum storage requirement for a variable string is 255 bytes (the length byte plus 254 characters).

Length	Character 1	Character 2	Character 3	Character 4	. . .	Character 254
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4		Byte 254

If a constant string parameter is entered directly in the program editor (126 characters maximum) or a variable string is initialized in the data block editor (254 characters maximum), the string assignment must begin and end with double quote characters.

ASCII output number format

- Positive values are written to the output buffer without a sign.
- Negative values are written to the output buffer with a leading minus sign (-).
- Leading zeros to the left of the decimal point (except the digit adjacent to the decimal point) are suppressed.
- Values are right-justified in the output string.

- Real numbers: Values to the right of the decimal point are rounded to fit in the specified number of digits to the right of the decimal point.
- Real numbers: The size of the output string must be a minimum of three bytes more than the number of digits to the right of the decimal point.

Integer to string conversion

LAD / FBD	STL	Description
I_S EN ENO IN OUT FMT	`ITS IN, OUT, FMT`	The Integer to String instruction converts an integer word IN to an ASCII string with a length of 8 characters. The format (FMT) assigns the conversion precision to the right of the decimal, and whether the decimal point is to be shown as a comma or a period. The resulting string is written to 9 consecutive bytes starting at OUT.

Non-fatal error conditions with ENO = 0	SM bits affected
• 0006H indirect address • 0091H operand out of range • Illegal format (nnn > 5) • FMT bit is not zero for the four most significant bits of the FMT byte	None

Input / output	Data type	Operand
IN	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
FMT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
OUT	STRING	VB, LB, *VD, *LD, *AC

The length of the output string is always 8 characters. The number of digits to the right of the decimal point in the output buffer is assigned by the nnn field. The valid range of the nnn field is 0 to 5. If you assign 0 digits to the right of the decimal point, then the value is converted without a decimal point. For values of nnn greater than 5, the output is a string of 8 ASCII space characters. The c bit specifies the use of either a comma (c=1) or a decimal point (c=0) as the separator between whole number and fraction. The most significant 4 bits of the format must be zero.

The following figure also shows examples of values that are formatted using a decimal point (c= 0) with three digits to the right of the decimal point (nnn = 011). The value at OUT is the length of the string stored in the next byte addresses.

FMT parameter for the integer to string instruction

FMT

MSB							LSB
7	6	5	4	3	2	1	0
0	0	0	0	c	n	n	n

c = comma (1) or decimal point (0)
nnn= digits to right of decimal point

	Out	Out +1	Out +2	Out +3	Out +4	Out +5	Out +6	Out +7	Out +8
in = 12	8				0	.	0	1	2
in = -123	8			-	0	.	1	2	3
in = 1234	8				1	.	2	3	4
in = -12345	8		-	1	2	.	3	4	5

Double integer to string conversion

LAD / FBD	STL	Description
DI_S EN ENO IN OUT FMT	**DTS IN, OUT, FMT**	The Double Integer to String instruction converts a double integer IN to an ASCII string with a length of 12 characters. The format (FMT) assigns the conversion precision to the right of the decimal, and whether the decimal point is to be shown as a comma or a period. The resulting string is written to 13 consecutive bytes starting at OUT.

Non-fatal error conditions with ENO = 0	SM bits affected
• 0006H indirect address • 0091H operand out of range • Illegal format (nnn > 5) • FMT bit is not zero for the four most significant bits of the FMT byte	None

Input / output	Data type	Operand
IN	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
FMT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
OUT	STRING	VB, LB, *VD, *LD, *AC

The length of the output string is always 12 characters. The number of digits to the right of the decimal point in the output buffer is specified by the nnn field. The valid range of the nnn field is 0 to 5. If you assign 0 digits to the right of the decimal point causes, then the value is displayed without a decimal point. For values of nnn greater than 5, the output is a string of 12 ASCII space characters. The c bit specifies the use of either a comma (c=1) or a decimal point (c=0) as the separator between the whole number and the fraction. The upper 4 bits of the format must be zero.

The following figure also shows examples of values that are formatted using a decimal point (c= 0) with four digits to the right of the decimal point (nnn = 100). The value at OUT is the length of the string stored in the next byte addresses.

FMT operand for the double integer to string instruction

FMT

MSB							LSB
7	6	5	4	3	2	1	0
0	0	0	0	c	n	n	n

c = comma (1) or decimal point (0)
nnn= digits to right of decimal point

	Out	Out +1	Out +2	Out +3	Out +4	Out +5	Out +6	Out +7	Out +8	Out +9	Out +10	Out +11	Out +12
in = 12	12							0	.	0	0	1	2
in = 1234567	12					1	2	3	.	4	5	6	7

Real to string conversion

LAD / FBD		Description
R_S EN ENO IN OUT FMT	**RTS IN, OUT, FMT**	The Real to String instruction converts a real value IN to an ASCII string. The format (FMT) assigns the conversion precision to the right of the decimal, whether the decimal point is to be shown as a comma or a period and the length of the output string. The resulting conversion is placed in a string beginning with OUT. The length of the resulting string is specified in the format and can be 3 to 15 characters.

Non-fatal error conditions with ENO = 0	SM bits affected
• 0006H indirect address • 0091H operand out of range • Illegal format – (nnn > 5) – ssss < 3 – ssss < number of characters required	None

Input / output	Data type	Operand
IN	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
FMT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
OUT	STRING	VB, LB, *VD, *LD, *AC

The real-number format used by the CPU supports a maximum of 7 significant digits. An attempt to display more than the 7 significant digits produces a rounding error.

The length of the output string is specified by the ssss field. A size of 0, 1, or 2 bytes is not valid. The number of digits to the right of the decimal point in the output buffer is assigned by the nnn field. The valid range of the nnn field is 0 to 5. If you assign 0 digits to the right of the decimal point, then the value is displayed without a decimal point. The output string is filled with ASCII space characters when nnn is greater than 5 or when the assigned length of the output string is too small to store the converted value. The c bit specifies the use of either a comma (c=1) or a decimal point (c=0) as the separator between the whole number and the fraction.

The following figure also shows examples of values that are formatted using a decimal point (c= 0) with one digit to the right of the decimal point (nnn = 001) and an output string length of 6 characters (ssss = 0110). The value at OUT is the length of the string stored in the next byte addresses.

FMT operand for the real to string instruction

FMT

MSB							LSB
7	6	5	4	3	2	1	0
s	s	s	s	c	n	n	n

ssss = length of output string
c = comma (1) or decimal point (0)
nnn= digits to right of decimal point

	Out	Out +1	Out +2	Out +3	Out +4	Out +5	Out +6
in = 1234.5	6	1	2	3	4	.	5
in = -0.0004	6				0	.	0
in = -3.67526	6			-	3	.	7
in = 1.95	6				2	.	0

See Also

Assigning a constant value for instructions (Page 120)

10.5.4 ASCII sub-string to number value conversion

LAD / FBD	STL	Description
S_I EN ENO IN OUT INDX	**STI IN, INDX, OUT**	ASCII sub-string to integer value conversion
S_DI EN ENO IN OUT INDX	**STD IN, INDX, OUT**	ASCII sub-string to double integer value conversion
S_R EN ENO IN OUT INDX	**STR IN, INDX, OUT**	ASCII sub-string to real value conversion

Non-fatal error conditions with ENO = 0	SM bits affected
• 0006H Indirect address • 0091H Operand out of range • 009BH Index = 0 • SM1.1 Overflow or illegal value	• SM1.1 Overflow or illegal value

Input / output	Data type	Operand
IN	STRING	VB, LB, *VD, *LD, *AC, Constant string
INDX	BYTE	VB, IB, QB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant

Input / output	Data type	Operand
OUT	INT	VW, IW, QW, MW, SMW, SW, T, C, LW, AC, AQW, *VD, *LD, *AC
	DINT, REAL	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

String input format for S_I (integer number) and S_DI (double integer number)

[spaces] [+ or -] [digits 0 - 9]

String input format for S_R (real number)

[spaces] [+ or -] [digits 0 - 9] [. or ,] [digits 0 - 9]

INDX parameter

The INDX value is normally set to 1, which starts the conversion with the first character of the string. The INDX value can be set to other values to start the conversion at different points within the string. This can be used when the input string contains text that is not part of the number to be converted. For example, if the input string is "Temperature: 77.8", you set INDX to a value of 13 to skip over the word "Temperature: " at the start of the string.

The Substring to Real instruction does not convert strings using scientific notation or exponential forms of real numbers. The instruction does not produce an overflow error (SM1.1) but converts the string to a real number up to the exponential and then terminates the conversion. For example, the string '1.234E6' converts without errors to a real value of 1.234.

The conversion is terminated when the end of the string is reached or when the first invalid character is found. An invalid character is any character that is not a digit (0 - 9), or one of the following characters: plus (+), minus (-), comma (,), or period (.).

The overflow error (SM1.1) is set whenever the conversion produces an integer value that is too large for the output value. For example, the Substring to Integer instruction sets the overflow error if the input string produces a value greater than 32767 or less than -32768.

The overflow error (SM1.1) is also set if no conversion is possible when the input string does not contain a valid value. For example, if the input string contains 'A123', the conversion instruction sets SM1.1 (overflow) and the output value remains unchanged.

Examples of valid and invalid input strings

Valid Input Strings
for Integer and Double Integer

Input String	Output Integer
'123'	123
'-00456'	-456
'123.45'	123
'+2345'	2345
'000000123ABCD'	123

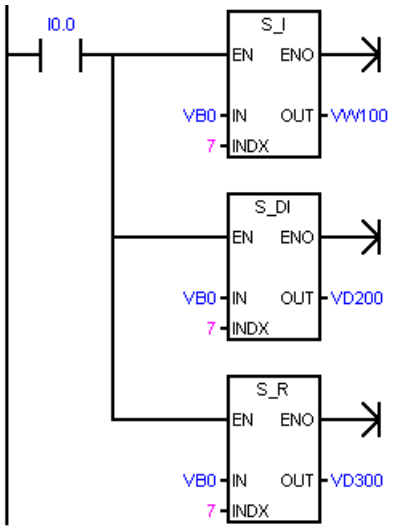
Valid Input Strings
for Real Numbers

Input String	Output Real
'123'	123.0
'-00456'	-456.0
'123.45'	123.45
'+2345'	2345.0
'00.000000123'	0.000000123

Invalid Input Strings

Input String
'A123'
' '
'++123'
'+-123'
'+ 123'

Example string conversion: Substring to integer, double integer, and real

LAD		STL
	S_I converts the numeric string to an integer value. S_DI converts the numeric string to a double integer value. S_R converts the numeric string to a real value.	**Network 1** **LD I0.0** **STI VB0, 7, VW100** **STD VB0, 7, VD200** **STR VB0, 7, VD300**

VB0 VB11

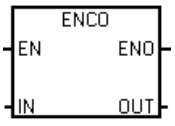
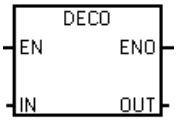
11	'T'	'e'	m'	'p'	' '	' '	'9'	'8'	'.'	'6'	'F'

After executing the network:
VW100 (integer) = 98
VD200 (double integer) = 98
VD300 (real) = 98.6

See also

Assigning a constant value for instructions (Page 120)

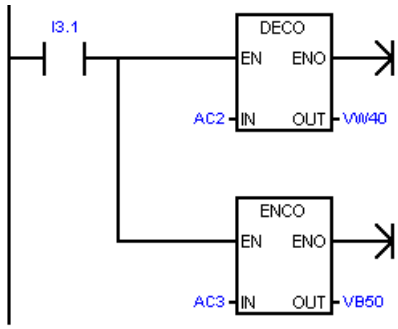
10.5.5 Encode and decode

LAD / FBD	STL	Description
	**ENCO IN, OUT**	Encode writes the bit number of the least significant bit set in the input word IN, to the least significant "nibble" (4 bits) of the output byte OUT.
	**DECO IN, OUT**	Decode sets the bit in the output word OUT that corresponds to the bit number represented by the least significant "nibble" (4 bits) of the input byte IN. All other bits of the output word are set to 0.

Non-fatal error conditions with ENO = 0	SM bits affected
• 0006H Indirect address	None

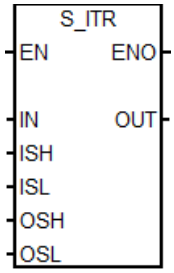
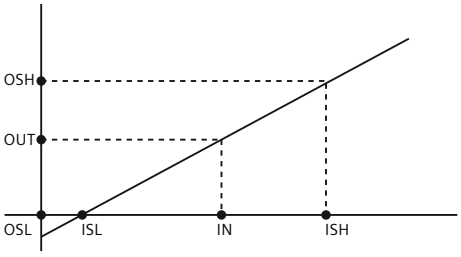
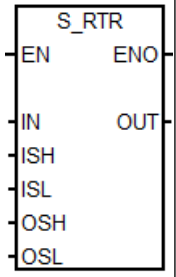
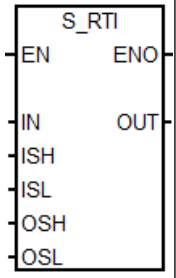
Input / output	Data type	Operand
IN	WORD (ENCO)	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
	BYTE (DECO)	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
OUT	BYTE (ENCO)	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC
	WORD (DECO)	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AQW, *VD, *LD, *AC

Example: Encode and decode

LAD		STL
	If AC2 contains error bits: 1. The DECO instruction sets the bit in VW40 that corresponds to this error code 2. The ENCO instruction converts the least significant bit set to an error code that is stored in VB50.	**Network 1** **LD I3.1** **DECO AC2, VW40** **ENCO AC3, VB50**

10.5.6 Scale

You can use the "Scale" instruction to scale the value at the IN input by mapping it to a specified value range.

LAD / FBD	STL	Description	
	**SITR IN, ISH, ISL, OSH, OSL, OUT**	You can use this instruction to convert analog input to internal data of S7-200 SMART. When this instruction is executed, the integer value at input IN is scaled to the value range, which is defined by the parameters OSH and OSL. The result of the scaling is an Real, which is stored at output OUT.	The following figure shows an example of how values can be converted:  OUT = [(OSH - OSL) * (IN - ISL)/(ISH - ISL)] + OSL • OUT: Conversion result • OSH: Convert result high limit • OSL: Covert result low limit • IN: Conversion object • ISH: Covert object high limit • ISL: Covert object low limit
	**SRTR IN, ISH, ISL, OSH, OSL, OUT**	You can use this instruction to convert a REAL to another REAL. When this instruction is executed, the Real value at input IN is scaled to the value range, which is defined by the parameters OSH and OSL. The result of the scaling is an Real, which is stored at output OUT.	
	**SRTI IN, ISH, ISL, OSH, OSL, OUT**	You can use this instruction to convert internal data of S7-200 SMART to analog output. When this instruction is executed, the Real value at input IN is scaled to the value range, which is defined by the parameters OSH and OSL. The result of the scaling is an integer, which is stored at output OUT.	

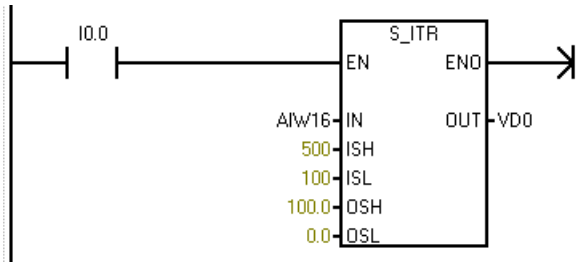
Non-fatal errors with ENO = 0

- 0X0090: invalid parameter, when (ISH ≤ ISL) or (OSH ≤ OSL) or IN is out of range [ISL, ISH]).

	input/output	Data type	Operand
S_ITR	IN	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	ISH	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	ISL	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	OSH	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
	OSL	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
	OUT	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

	input/output	Data type	Operand
S_RTR	IN	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
	ISH	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
	ISL	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
	OSH	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
	OSL	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
	OUT	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC
S_RTI	IN	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
	ISH	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
	ISL	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
	OSH	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	OSL	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	OUT	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AQW, *VD, *AC, *LD

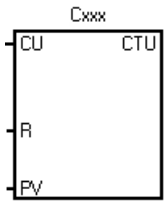
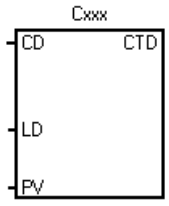
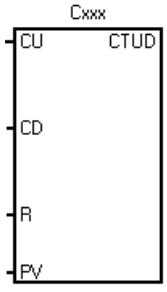
Example:

LAD	STL
 If the "I0.0" operand has signal state "1", the instruction is executed. The value at the "IN" input ranges from "ISH" and "ISL" is scaled to the range of values defined by the values at the "OSH" and "OSL" inputs. The result is stored in the "OUT" output. If the instruction is executed without errors, the ENO enable output has signal state "1" and the "OUT" output is set.	**Network 1** **LD I0.0** **SITR AIW16, 500, 100, 100.0, 0.0, VD0**

IN	ISH	ISL	OSH	OSL	OUT
350	500	100	100.0	0.0	62.5

10.6 Counters

10.6.1 Counter instructions

LAD / FBD	STL	Description
	**CTU Cxxx, PV**	**LAD/FBD:** The **CTU** count up instruction counts up from the current value each time the count up CU input makes the transition from OFF to ON. When the current value Cxxx is greater than or equal to the preset value PV, the counter bit Cxxx is set ON. The current count value is reset when the reset input R is set ON, or when the reset instruction is executed for the Cxxx address. The counter stops counting when it reaches the maximum value 32,767. **STL:** R reset input is the top of stack value**.** CU count up input is loaded in the second stack level
	**CTD Cxxx, PV**	**LAD/FBD:** The **CTD** count down instruction counts down from the current value of that counter each time the CD count down input makes the transition from OFF to ON. When the current value Cxxx is equal to 0, the counter bit Cxxx turns ON. The counter resets the counter bit Cxxx and loads the current value with the preset value PV when the LD load input is set ON. The counter stops upon reaching zero, and the counter bit Cxxx is set ON. **STL:** LD load input is the top of stack value. CD count down input value is loaded in the second stack level
	**CTUD Cxxx, PV**	**LAD/FBD:** The **CTUD** count up/down instruction counts up each time the CU count up input makes the transition from OFF to ON, and counts down each time the CD count down input makes the transition from OFF to ON. The current value Cxxx of the counter maintains the current count. The PV preset value is compared to the current value each time the counter instruction is executed. Upon reaching maximum value 32,767, the next rising edge at the count up input causes the current count to wrap around to the minimum value -32,768. On reaching the minimum value -32,768, the next rising edge at the count down input causes the current count to wrap around to the maximum value 32,767. When the current value Cxxx is greater than or equal to the PV preset value, the counter bit Cxxx is set ON. Otherwise, the counter bit is OFF. The counter is reset when the R reset input is set ON, or when the Reset instruction is executed for the Cxxx address. **STL:** R reset input is the top of stack value. CD count down input value is loaded in the second stack level. CU count Up input value is loaded in the third stack level

Input / output	Data type	Operand
Cxxx	WORD	Constant (C0 to C511)
CU, CD (LAD)	BOOL	Power flow
CU, CD (FBD)	BOOL	I, Q, V, M, SM, S, T, C, L, Logic flow

Input / output	Data type	Operand
R (LAD)	BOOL	Power Flow
R (FBD)	BOOL	I, Q, V, M, SM, S, T, C, L, Logic flow
LD (LAD)	BOOL	Power Flow
LD (FBD)	BOOL	I, Q, V, M, SM, S, T, C, L, Logic flow
PV	INT	IW, QW, VW, MW, SMW, SW, LW, T, C, AC, AIW, *VD, *LD, *AC, Constant

Note

Since there is one current value for each counter, do not assign the same counter number to more than one counter. (Up Counters, Up/Down Counters, and Down counters with the same number access the same current value.)

When you reset a counter using the Reset instruction, the counter bit is reset and the counter current value is set to zero. Use the counter number to reference both the current value and the counter bit of that counter.

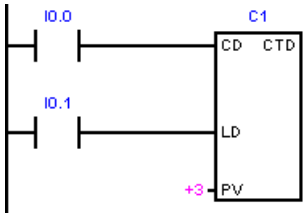
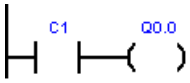
See also Configuring the retentive ranges - system block configuration (Page 190)

Counter operation

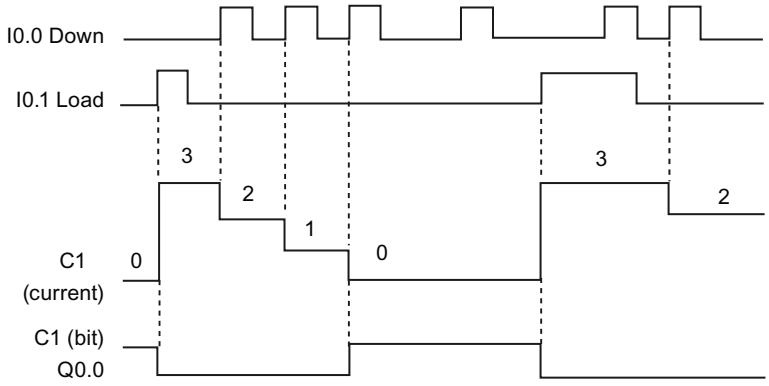
Type	Operation	Counter bit	Power cycle / first scan
CTU	• CU increments the current value. • Current value continues to increment until it reaches 32,767.	The counter bit is set ON when: Current value >= Preset	• Counter bit is OFF. • Current value can be retained [1]
CTD	• CD decrements the current value until the current value reaches 0.	The counter bit is set ON when: Current value = 0	• Counter bit is OFF. • Current value can be retained [1]
CTUD	• CU increments the current value. • CD decrements the current value. • Current value continues to increment or decrement until the counter is reset.	The counter bit is set ON when: Current value >= Preset	• Counter bit is OFF. • Current value can be retained [1]

[1] You can select the current value for the counter to be retentive, but not the counter bit value.

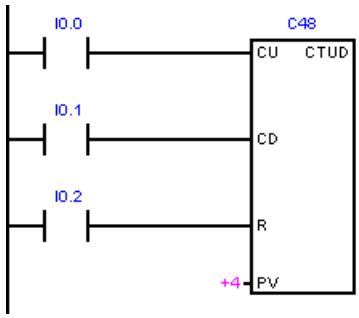
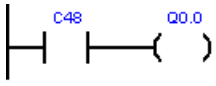
Example CTD count down

LAD		STL
	Count down counter C1 current value counts from 3 to 0 With I0.1 OFF, I0.0 OFF-ON decrements C1 current value I0.1 ON loads countdown preset value 3	**Network 1** **LD I0.0** **LD I0.1** **CTD C1, +3**
	C1 bit is ON when counter C1 current value = 0	**Network 2** **LD C1** **= Q0.0**

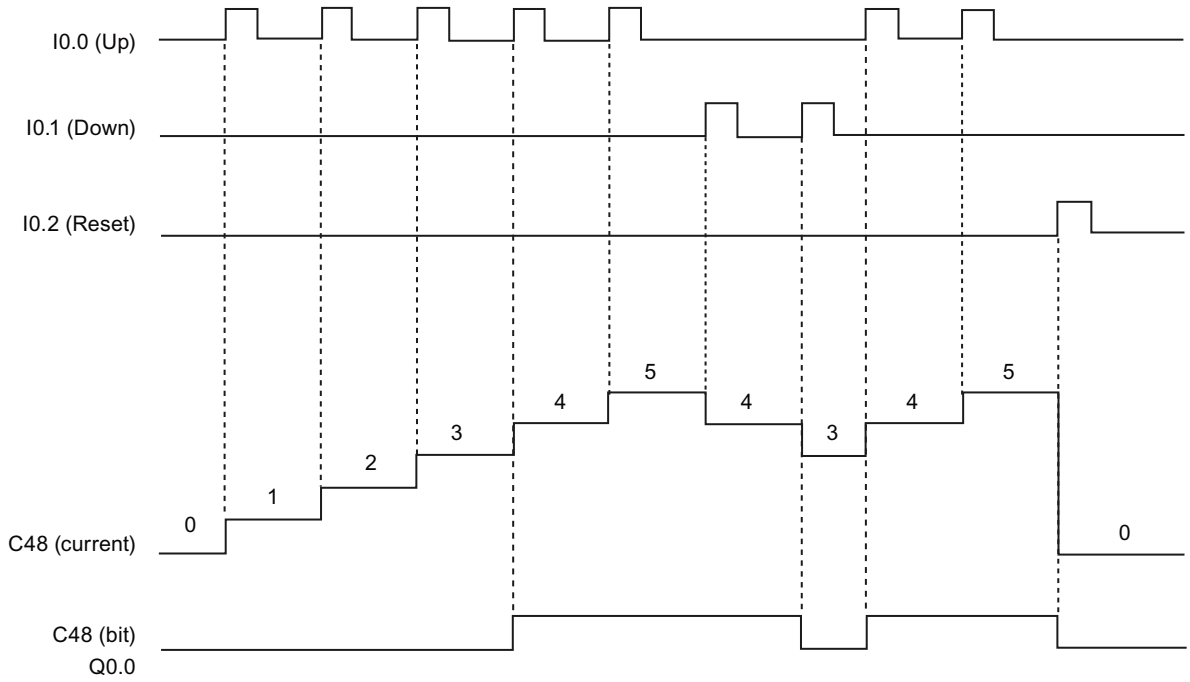
Timing diagram



Example CTUD count up/down

LAD		STL
	I0.0 counts up I0.1 counts down I0.2 resets current value to 0	**Network 1** **LD I0.0** **LD I0.1** **LD I0.2** **CTUD C48, +4**
	Count Up/Down counter C48 turns on C48 bit when current value >= 4	**Network 2** **LD C48** **= Q0.0**

Timing diagram



10.6.2 High-speed counter instructions

High-speed counters can count high-speed events that cannot be controlled by standard counters. Standard counters operate at a slower rate that is limited by the PLC scan time. You can use the HDEF and HSC instructions and create your own HSC routines, or you can simplify the programming tasks by using the High Speed Counter wizard.

LAD / FBD	STL	Description
HDEF EN ENO HSC MODE	**HDEF HSC, MODE**	The High-Speed Counter Definition instruction (HDEF) selects the operating mode of a specific high-speed counter (HSC0-7). The mode selection defines the clock, direction, and reset functions of the high-speed counter. You must use one High-Speed Counter Definition instruction for each of up to eight active high-speed counters.
HSC EN ENO N	**HSC N**	The High-Speed Counter (HSC) instruction configures and controls the high-speed counter, based on the state of the HSC special memory bits. The parameter N specifies the high-speed counter number. The high-speed counters can be configured for up to eight different modes of operation. Each counter has dedicated inputs for clocks, direction control, and reset where these functions are supported. In AB quadrature phase, you can select one times (1x) or four times (4x) the maximum counting rate. All counters run at maximum rates without interfering with one another.

Error conditions with ENO = 0		SM bits affected
HDEF: • 0003H Input point conflict • 0004H Illegal instruction in interrupt • 000AH HSC redefinition • 0016H Attempted to use HSC or Edge Interrupt on Input that is allocated for use by Motion Functionality • 0090H Invalid HSC number	HSC: • 0001H HSC before HDEF • 0005H Simultaneous HSC/PLS • 0090H Invalid HSC number	None

Input / output	Data type	Operand
HSC	BYTE	HSC number constant (0, 1, 2, 3, 4, 5, 6 or 7)
MODE	BYTE	Counter • Mode number constant: Eight possible modes (0, 1, 3, 4, 6, 7, 9, or 10) Frequency • Mode number constant: Four possible modes (0, 3, 6, 9)
N	WORD	HSC number constant (0, 1, 2, 3, 4, 5, 6 or 7)

HSC operation

A high-speed counter can be used as the drive for a drum timer, where a shaft rotating at a constant speed is fitted with an incremental shaft encoder. The shaft encoder provides a specified number of counts per revolution and a reset pulse that occurs once per revolution. The clock(s) and the reset pulse from the shaft encoder provide the inputs to the high-speed counter.

The high-speed counter is loaded with the first of several presets, and the desired outputs are activated for the time period where the current count is less than the current preset. The counter is set up to provide an interrupt when the current count is equal to preset and also when reset occurs.

As each current-count-value-equals-preset-value interrupt event occurs, a new preset is loaded and the next state for the outputs is set. When the reset interrupt event occurs, the first preset and the first output states are set, and the cycle is repeated.

Since the program interrupts occur at a much lower rate than the counting rates of the high-speed counters, precise control of high-speed operations can be implemented with relatively minor impact to the overall PLC scan cycle time. The method of interrupt attachment allows each load of a new preset to be performed in a separate interrupt routine for easy state control. (Alternatively, all interrupt events can be processed in a single interrupt routine.)

HSC input assignments and capabilities

All high-speed counters function the same way for the same mode of operation. The HSC input connections (clock, direction, and reset) must use the CPU's integrated input channels as shown in the High-speed counter summary (Page 327) table. Input channels located on a signal board or an expansion module cannot be used for high-speed counters.

Note

You must ensure that high-speed counter inputs are correctly filtered and wired, for counting high frequency signals.

In an S7-200 SMART CPU, all high-speed counter inputs are connected to internal input filter circuits. The S7-200 SMART default input filter setting is 6.4 ms, which limits the maximum counting rate to 78 Hz. You must change the filter settings to count higher frequencies.

Refer to "Noise reduction for high-speed inputs (Page 327)" for details about system block filter options, maximum counting frequencies, shielding requirements, and external pull-down circuits.

Available HSC counter types

Mode	Description	Input		
0	Single-phase clock counter with internal direction control	Clock		
1		Clock		Reset
3	Single-phase clock counter with external direction control	Clock	Direction	
4		Clock	Direction	Reset
6	Two-phase clock counter with 2 clock inputs (clock-up and clock-down)	Clock-up	Clock-down	
7		Clock-up	Clock-down	Reset
9	AB quadrature phase counter	Phase A clock	Phase B Clock	
10		Phase A clock	Phase B Clock	Reset

HSC operating rules

- Before you use a high-speed counter, you must execute the HDEF instruction (High-Speed Counter Definition) to select a counter mode. Use the first scan memory bit, SM0.1 (this bit is ON for the first scan and OFF for subsequent scans) to execute HDEF directly, or call a subroutine that contains the HDEF instruction.
- You can use all counter types with or without a reset input.
- When you activate the reset input, it clears the current value and holds it clear until you deactivate the reset input.

Reference information

Refer to the following sections for further information:

- High-speed counter programming (Page 330)
- High-speed counter summary (Page 327)
- Example initialization sequences for high-speed counters (Page 340)
- Noise reduction for high-speed inputs (Page 330)

10.6.3 High-speed counter summary

	Clock A	Direction / Clock B	Reset	Single phase / Dual phase maximum clock / input rate	AB quadrature phase maximum clock / input rate
HSC0, HSC1, HSC2, HSC3, HSC4, HSC5, HSC6, HSC7	I0.0-I1.7 [1]			• 200 kHz	• 200 kHz • A/B phase (4X) frequency support up to 800 kHZ

[1] For the same channel HSC, Clock A and Direction / Clock B can only be selected simultaneously in IB0（I0.0 ~I0.7） or simultaneously in IB1（I1.0 ~I1.7）.

10.6.4 Noise reduction for high-speed inputs

Counting high-speed pulses with HSC inputs

Note

High-speed input wiring must use shielded cables

Use shielded cable with a maximum length of 50 m, when connecting HSC input channels I0.0-I1.7.

One or both of the following actions may be necessary to correctly operate a high-speed counter.

- Adjust the System Block digital input filter time of the input channels used by the HSC channel. The S7-200 SMART CPU applies input filtering before the counting of pulses by the HSC channel. This means that if an HSC input pulse occurs at a rate that is filtered out by the input filtering, then the HSC does not detect any pulses on the input. You must make sure that you configure the filter time of each input of the HSC to a value that allows counting at the rate your application requires. This includes direction and reset inputs. The following table shows the maximum input frequency that an HSC can detect for each input filter configuration:

Input filter time	Maximum detectable frequency
0.2 µs	200 kHz
0.4 µs	200 kHz
0.8 µs	200 kHz
1.6 µs	200 kHz
3.2 µs	156 kHz
6.4 µs	78 kHz
12.8 µs	39 kHz
0.2 ms	2.5 kHz
0.4 ms	1.25 kHz
0.8 ms	625 Hz
1.6 ms	312 Hz
3.2 ms	156 Hz
6.4 ms	78 Hz
12.8 ms	39 Hz

- If the device generating the HSC input signals does not drive the input signals both high and low, then signal distortion can occur at high speeds. This can occur if the output of the device is an open-collector transistor. When the transistor turns off, there is nothing driving the signal to a low state. The signal transitions to a low state, but the time to do so is dependent on the input resistance and capacitance of the circuitry. This condition can result in missed pulses. You can prevent this condition by wiring a pull-down resistor to the input signals, as seen in the following figure. Since the input voltage of the CPU is 24 V DC, the resistor has to be rated for a high wattage. A 200 ohm, 5 Watt resistor is a suitable choice.

Note

The signal line and the common wire need to be routed in pairs, and it is recommended to use twisted pair for better signal quality.

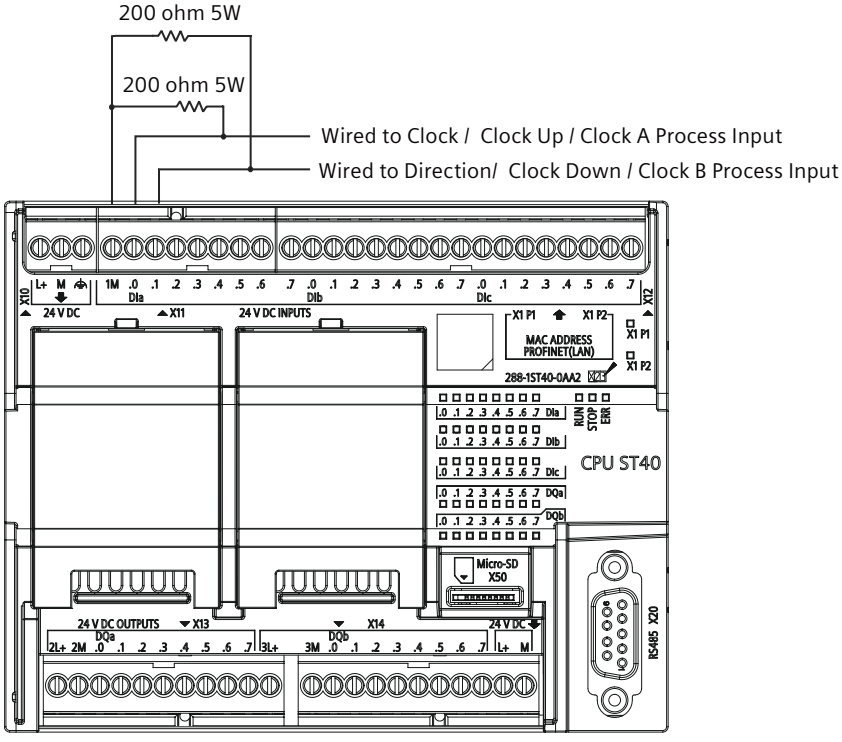


Figure 10-1 Pull-down resistor wiring for open-collector HSC input drivers

10.6.5 High-speed counter programming

You can use the high-speed counter wizard to simplify HSC programming tasks. The wizard helps you select the counter type/mode, preset/current values, counter options, and generates the necessary special memory assignments, subroutines, and interrupt routines.

Note

You must ensure that high-speed counter inputs are correctly filtered and wired for counting high frequency signals.

In an S7-200 SMART CPU, all high-speed counter inputs are connected to internal input filter circuits. The S7-200 SMART CPU default input filter setting is 6.4 ms, which limits the maximum counting rate to 78 Hz. You must change the filter settings to count higher frequencies.

Refer to "Noise reduction for high-speed inputs (Page 327)" for details about system block filter options, maximum counting frequency, shielding requirements, and external pull-down circuits.

Configuring high-speed counters

Use one of the following actions to configure the high-speed counter wizard:

- Open the wizard: Select "High-Speed Counter" in the wizards area of the Tools menu ribbon strip.
- Open the wizard: Double-click "High-Speed Counter" node in the wizards folder, from the project tree.

With the wizard open, assign the HSC setup values. You can navigate through the wizard setup pages, modify parameters, and then generate new wizard program code.

Your program must perform the following basic tasks to use a high-speed counter:

- Define the counter and mode (execute the HDEF instruction exactly once for each counter).
- Set the control byte in SM memory.
- Set the current value (starting value) in SM memory.
- Set the preset value (target value) in SM memory.
- Assign and enable appropriate interrupt routines.
- Activate the high-speed counter (execute the HSC instruction).

HDEF instruction sets the counting mode

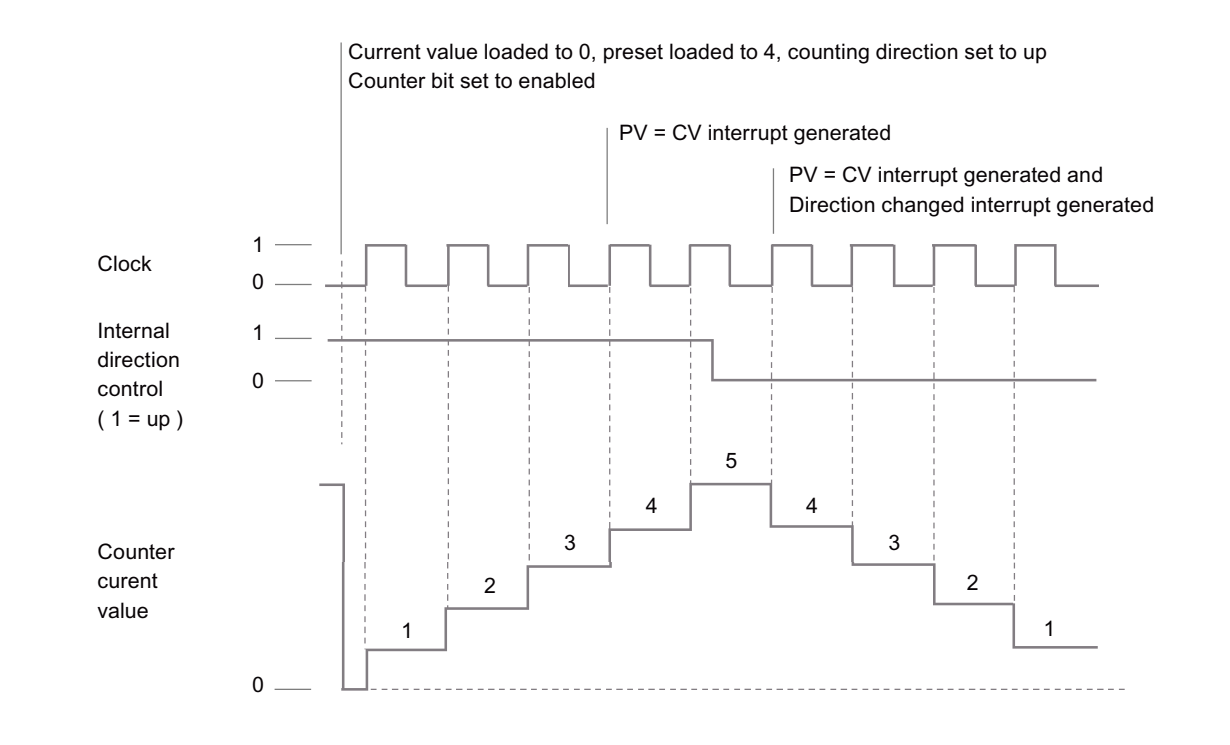
The HDEF instruction assigns HSC counter mode. The following table shows the physical inputs assigned for clock, direction control, and reset functions. The same input cannot be used for two different functions, but you can use any input not being used by the present mode of its high-speed counter for another purpose. For example, if the present mode of HSC0 is mode 1, which uses I0.0 and I0.4; then you can use I0.1, I0.2, and I0.3 for edge interrupts, HSC3, or motion control inputs.

Mode	Description	Input assignment		
0	Single-phase counter with internal direction control	Clock		
1		Clock		Reset

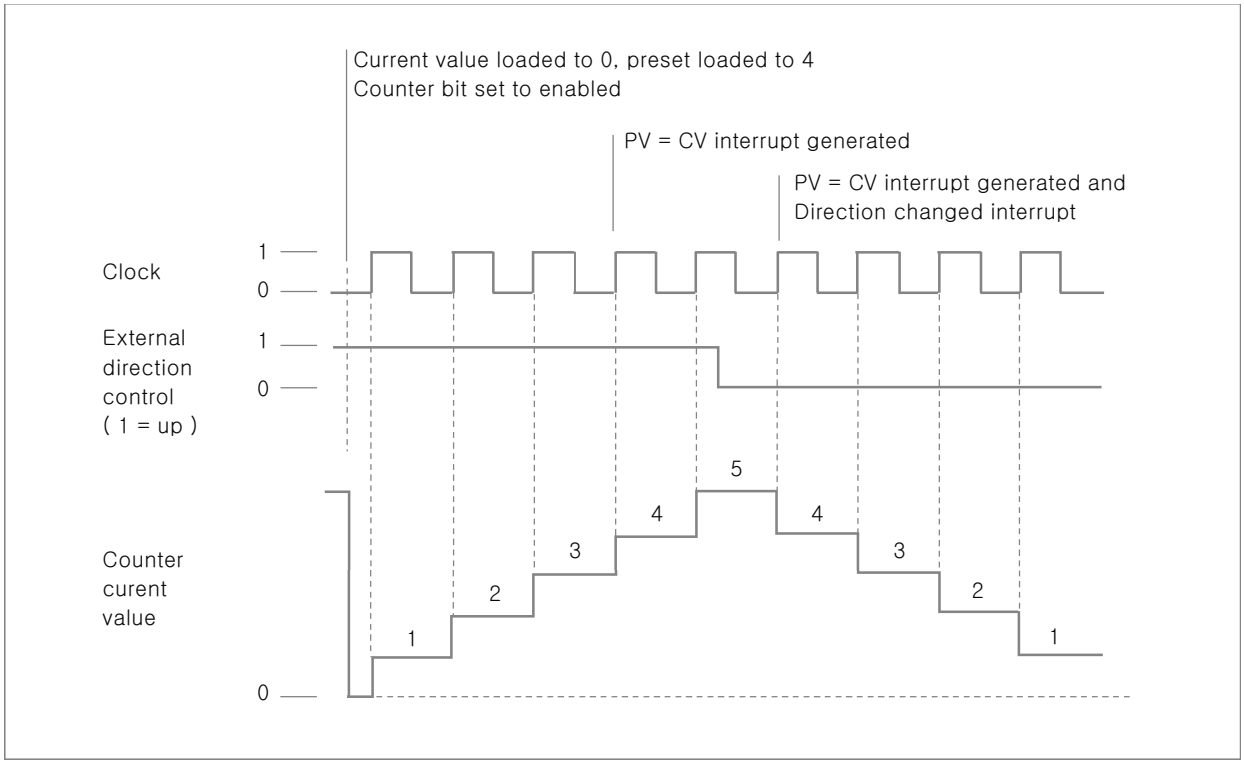
Mode	Description	Input assignment		
3	Single-phase counter with external direction control	Clock	Direction	
4		Clock	Direction	Reset
6	Two-phase counter with 2 clock inputs	Clock Up	Clock Down	
7		Clock Up	Clock Down	Reset
9	AB quadrature phase counter	Clock A	Clock B	
10		Clock A	Clock B	Reset

How mode selection affects counter operation

HSC modes 0 and 1

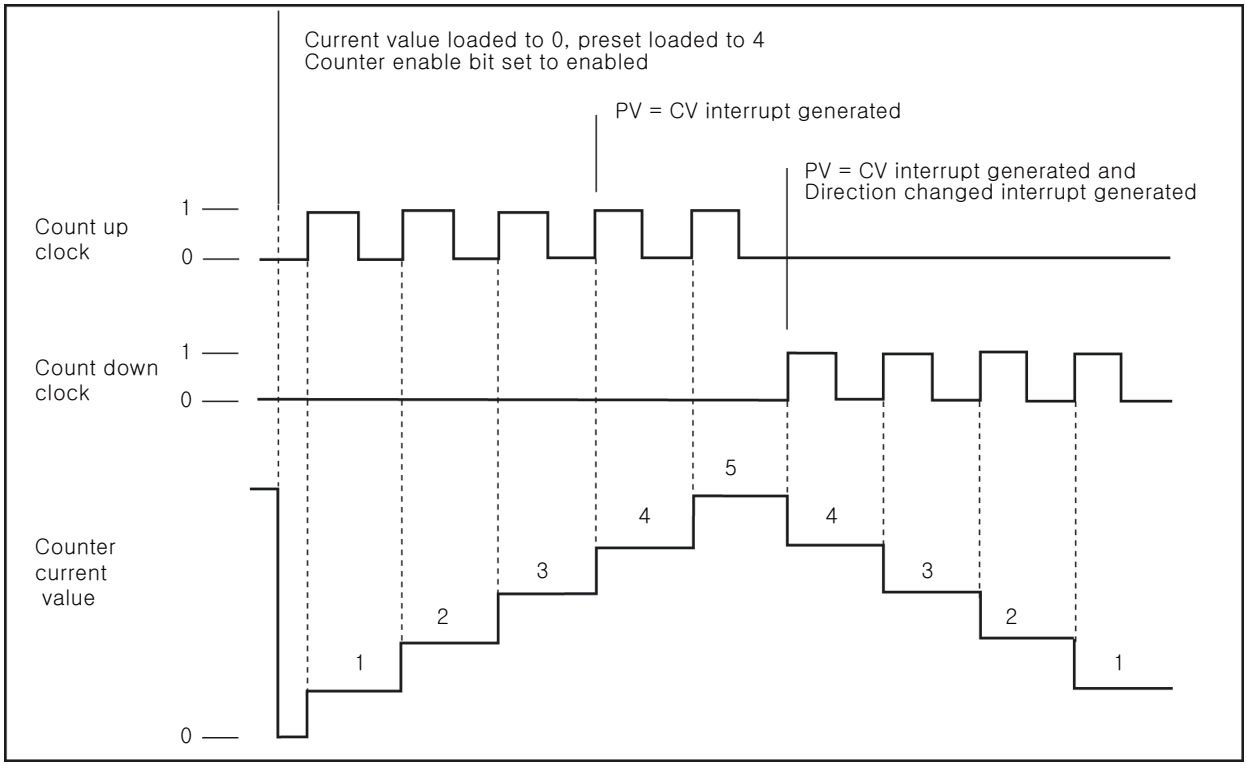


HSC modes 3 and 4

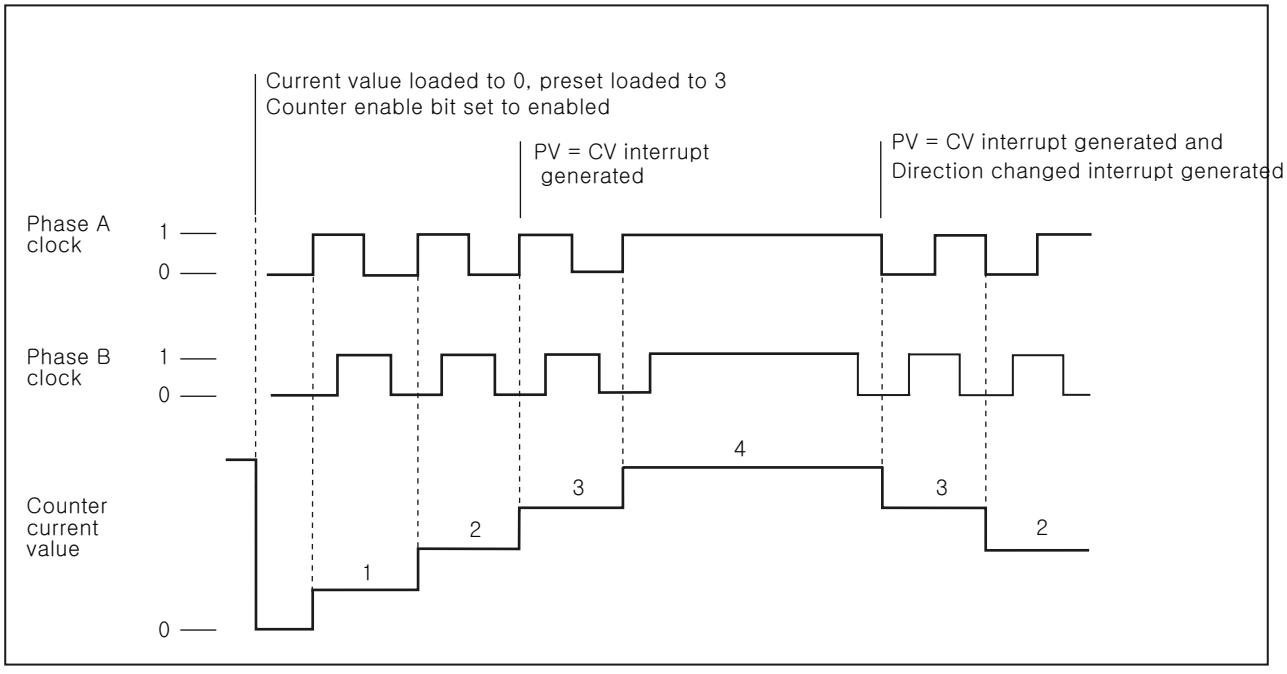


HSC modes 6 and 7

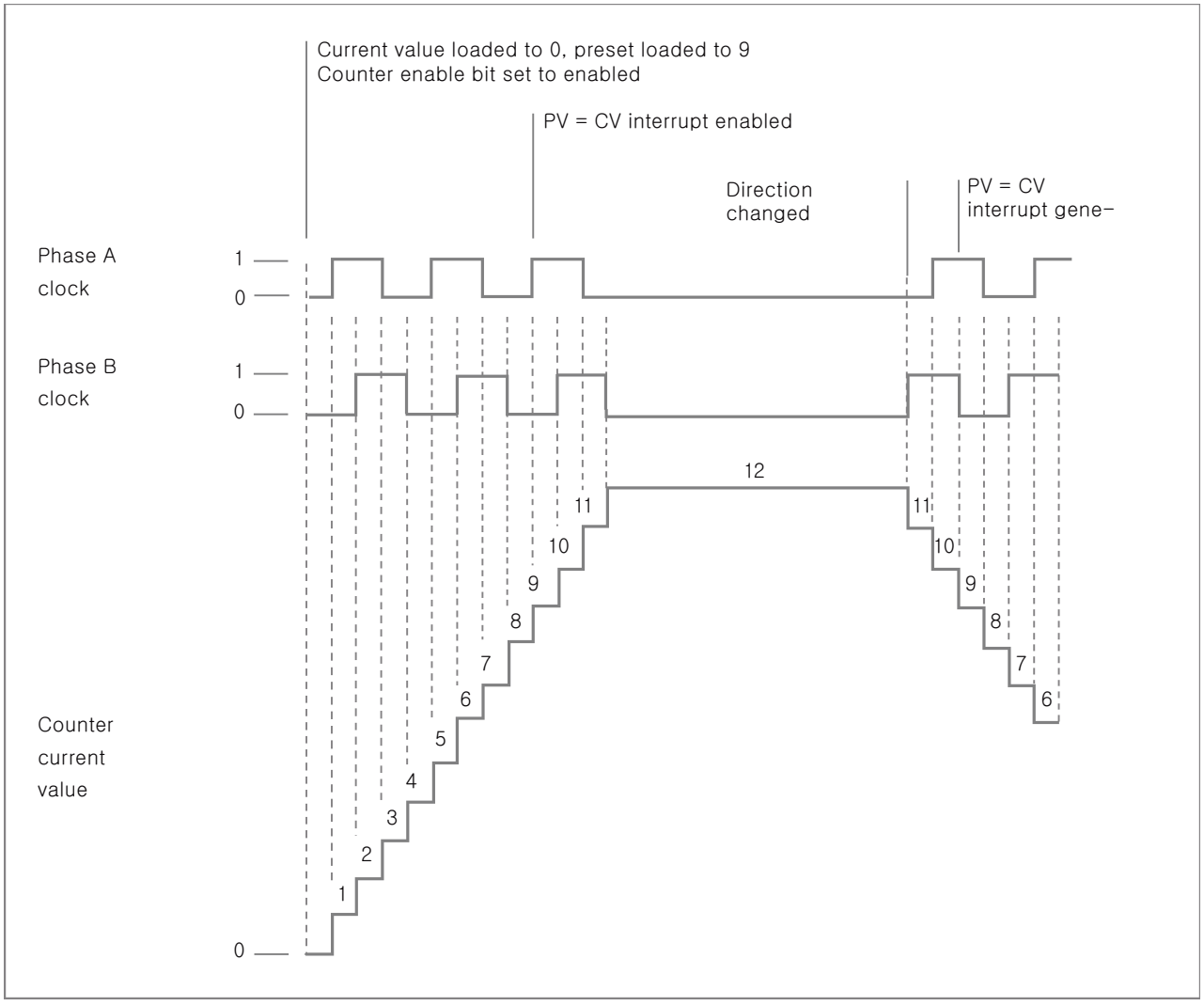
When you use counting modes 6 or 7, and rising edges on both the up clock and down clock inputs occur within 0.3 microseconds of each other, the high-speed counter could see these events as happening simultaneously. If this happens, the current value is unchanged and no change in counting direction is indicated. As long as the separation between rising edges of the up and down clock inputs is greater than this time period, the high-speed counter captures each event separately. In either case, the program generates no error, and the counter maintains the correct count value.



HSC modes 9 and 10 (AB quadrature phase 1x)



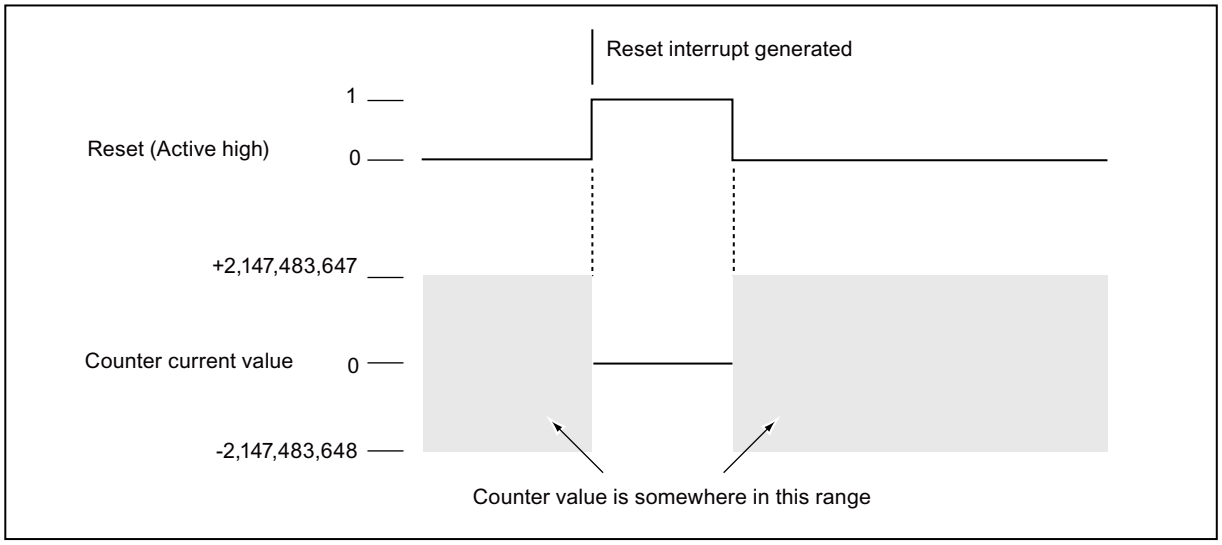
HSC modes 9 and 10 (AB quadrature phase 4x)



Reset operation

The operation of reset shown in the following figure applies to all modes that use the reset input. In the figure below, the reset operation is shown with the active state assigned as the high level.

HSC reset



HDEF instruction sets the reset active level and counting rate

Note

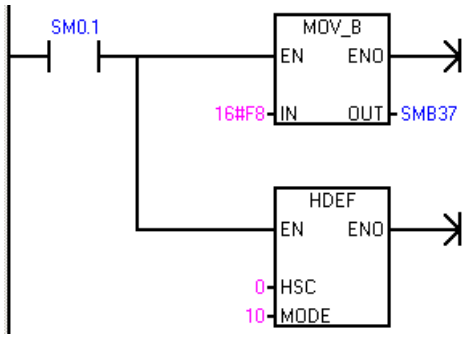
You must set these two control bits to the desired state before the HDEF instruction is executed. Otherwise, the counter takes on the default configuration for the counter mode selected.

Once the HDEF instruction has been executed, you cannot change the counter setup unless you first place the CPU in STOP mode.

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7	Description (used only when HDEF is executed)
SM37.0	SM47.0	SM57.0	SM137.0	SM147.0	SM157.0	SM361.0	SM371.0	**Active level control bit for Reset:*** • 0 = Reset is active high • 1 = Reset is active low
SM37.2	SM47.2	SM57.2	SM137.2	SM147.2	SM157.2	SM361.2	SM371.2	**Counting rate selection for AB quadrature phase counters:*** • 0 = 4X counting rate • 1 = 1X counting rate

* The default setting of the reset input is active high, and the AB quadrature phase counting rate is 4x (or four times the input clock frequency).

Example: High-speed counter definition

LAD			STL
MAIN		On the first scan: 1. Select the reset input to be active high and select 4x mode. 2. Configure HSC0 for AB quadrature phase with reset input (mode 10).	**Network 1** **LD SM0.1** **MOVB 16#F8, SMB37** **HDEF 0, 10**

HSC instruction enables counters, sets counting direction, and loads preset/current count values

The HSC instruction uses the control byte during execution. After you assign the counter and the counter mode, you can program the dynamic parameters of the counter. Each high-speed counter has a control byte in SM memory that allows the following actions:

- Enabling or disabling the counter
- Controlling the direction (modes 0 and 1 only)
- Loading the current value
- Loading the preset value

HSC Control bytes

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7	Description
SM37.3	SM47.3	SM57.3	SM137.3	SM147.3	SM157.3	SM361.3	SM371.3	Counting direction control bit: • 0 = Count down • 1 =Count up
SM37.4	SM47.4	SM57.4	SM137.4	SM147.4	SM157.4	SM361.4	SM371.4	Write the counting direction to the HSC: • 0 = No update • 1 =Update direction
SM37.5	SM47.5	SM57.5	SM137.5	SM147.5	SM157.5	SM361.5	SM371.5	Write the new preset value to the HSC: • 0 = No update • 1 = Update preset
SM37.6	SM47.6	SM57.6	SM137.6	SM147.6	SM157.6	SM361.6	SM371.6	Write the new current value to the HSC: • 0 = No update • 1 =Update current value

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7	Description
SM37.7	SM47.7	SM57.7	SM137.7	SM147.7	SM157.7	SM361.7	SM371.7	Enable the HSC: • 0 = Disable the HSC • 1 =Enable the HSC
SMB195	SMB197	SMB199	SMB201	SMB203	SMB205	SMB207	SMB209	HSC Type: • 0 = Counting mode • 1 = Frequency measurement mode
SMB196	SMB198	SMB200	SMB202	SMB204	SMB206	SMB208	SMB210	HSC frequency measurement sampling period, working in frequency measurement mode: • 0 = Sampling period 1 s • 1 = Sampling period 100 ms • 2 = Sampling period 10 ms

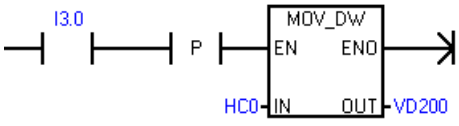
Read the HSC current value with your program

You can only read the current value of each high-speed counter using the data type HC (High-Speed-Counter Current) followed by the counter identifier number (0, 1, 2, 3, 4, 5, 6 or 7) as shown in the following table. Use the HC data type whenever you wish to read the current count, either in a status chart or in the user program. The HC data type is read-only double word value; you cannot write a new current count to the high-speed counter using the HC data type.

Current values of HSC0, HSC1, HSC2, HSC3, HSC4, HSC5, HSC6 and HSC7

Value to be read	HSC0 address	HSC1 address	HSC2 address	HSC3 address	HSC4 address	HSC5 address	HSC6 address	HSC7 address
CV (counter current value)	HC0	HC1	HC2	HC3	HC4	HC5	HC6	HC7

Example: Reading and saving the current count value

LAD			STL
MAIN		Save the value of HSC0 into VD200 when I3.0 transitions from OFF to ON.	**Network 1** **LD I3.0** **EU** **MOVD HC0, VD200**

Set current values and preset values with your program

Each high-speed counter has a 32-bit current value (CV) and a 32-bit preset value (PV) stored internally. The current value is the actual count value of the counter, while the preset value is a comparison value optionally used to trigger an interrupt when the current value reaches the preset value. You can read the current value using the HC data type as described in the previous section. You cannot read the preset value directly. To load a new current or preset value into the high-speed counter, you must set up the control byte and the special memory double-word(s) that hold the desired new current and/or new preset values, and also execute the HSC instruction to cause the new values to be transferred to the high-speed counter. The table below lists the special memory double-words used to hold the desired new current and preset values.

Use the following steps to write a new current value and/or new preset value to the high-speed counter (steps 1 and 2 can be done in either order):

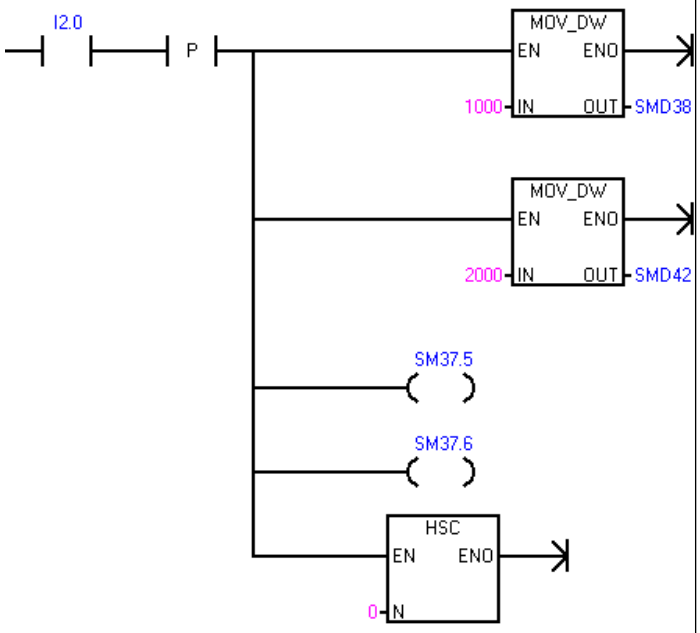
1. Load the value to be written into the appropriate SM new current value and/or new preset value (see the table below). Loading these new values does not affect the high-speed counter yet.
2. Set or clear the appropriate bits in the appropriate control byte to indicate whether to update the current and/or preset values (bit x.5 for preset and x.6 for current). Manipulating these bits does not affect the high-speed counter yet.
3. Execute the HSC instruction referencing the appropriate high-speed counter number. Executing this instruction causes the control byte to be examined. If the control byte specifies an update for the current, the preset, or both, then the appropriate values are copied from the SM new current value and/or new preset value locations into the high-speed counter internal registers.

Value to be loaded	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7
New current value (new CV)	SMD38	SMD48	SMD58	SMD138	SMD148	SMD158	SMD362	SMD372
New preset value (new PV)	SMD42	SMD52	SMD62	SMD142	SMD152	SMD162	SMD366	SMD376

Note

Changes to the control byte and the SM locations for new current value and new preset value does not affect the high-speed counter until the corresponding HSC instruction is executed.

Example: Updating the current and preset values

LAD		STL
MAIN program network  	Update the current count to 1000 and the preset value to 2000 for HSC0 when I2.0 transitions from OFF to ON.	**Network 1** **LD I2.0** **EU** **MOVD 1000, SMD38** **MOVD 2000, SMD42** **= SM37.5** **= SM37.6** **HSC 0**

Attaching HSC interrupt routines in your program

All high-speed counter modes support an interrupt event when the current value of the HSC is equal to the loaded preset value. Counter modes that use an external reset input support an interrupt on activation of the external reset. All counter modes except modes 0 and 1 support an interrupt on a change in counting direction. Each of these interrupt conditions can be enabled or disabled separately. For a complete discussion on the use of interrupts, see the section about Interrupt instructions (Page 393).

HSC status byte

A status byte for each high-speed counter provides status memory bits that indicate the current counting direction and whether the current value is greater than or equal to the preset value. The following table defines these status bits for each high-speed counter.

Note

Status bits are valid only while the high-speed counter interrupt routine executes. The purpose of monitoring the state of the high-speed counter is to enable interrupts for the events that are of consequence to the operation being performed.

Table 10-14 Status bits for HSC0, HSC1, HSC2, HSC3, HSC4, HSC5, HSC6 and HSC7

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7	Description
SM36.5	SM46.5	SM56.5	SM136.5	SM146.5	SM156.5	SM360.5	SM370.5	**Current counting direction status bit:** • 0 = Counting down • 1 = Counting up
SM36.6	SM46.6	SM56.6	SM136.6	SM146.6	SM156.6	SM360.6	SM370.6	**Current value equals preset value status bit:** • 0 = Not equal • 1 = Equal
SM36.7	SM46.7	SM56.7	SM136.7	SM146.7	SM156.7	SM360.7	SM370.7	**Current value greater than preset value status bit:** • 0 = Less than or equal • 1 = Greater than

Reference information

Refer to the following sections for further information:

- High-speed counter instructions (Page 325)
- High-speed counter summary (Page 327)
- Example initialization sequences for high-speed counters (Page 340)

10.6.6 Example initialization sequences for high-speed counting mode

HSC0 is used as the counter in the following descriptions of the initialization and operation sequences.

The initialization descriptions assume that you have just placed the CPU in RUN mode, and, for that reason, the first scan memory bit is true. If this is not the case, remember that you can execute the HDEF instruction only one time for each high-speed counter, after entering RUN mode. Executing HDEF for a high-speed counter a second time generates a run-time error and does not change the counter setup from the way it was set up on the first execution of HDEF for that counter.

Note

Although the following sequences show how to change direction, current value, and preset value individually, you can change all or any combination of them in the same sequence by setting the value of SMB37 appropriately and then executing the HSC0 instruction.

Initialization of modes 0 and 1

The following steps describe how to initialize HSC0 for single-phase up/down counter with internal direction (modes 0 and 1):

1. Use the first scan memory bit to call a subroutine in which the initialization operation is performed. Since you use a subroutine call, subsequent scans do not make the call to the subroutine, which reduces scan time execution and provides a more structured program.
2. In the initialization subroutine, load SMB195 and SMB37 according to the desired control operation.
   For example: SMB195=0 indicates counting mode
   SMB37 = 16#F8 produces the following results:
   - Enables the counter
   - Writes a new current value
   - Writes a new preset value
   - Sets the direction to count up
   - Sets the reset input to be active high
3. Execute the HDEF instruction with the HSC input set to 0 and the MODE input set to one of the following:
   - Mode 0 for no external reset
   - Mode 1 for external reset
4. Load SMD38 (double-word-sized value) with the desired current value (load with 0 to clear it).
5. Load SMD42 (double-word-sized value) with the desired preset value.
6. In order to capture the current value equal to preset event, program an interrupt by attaching the CV = PV interrupt event (event 12) to an interrupt routine. See the section that discusses the Interrupt instructions for complete details on interrupt processing.
7. In order to capture an external reset event, program an interrupt by attaching the external reset interrupt event (event 28) to an interrupt routine.
8. Execute the global interrupt enable instruction (ENI) to enable the interrupts.
9. Execute the HSC instruction to cause the CPU to program HSC0.
10. Exit the subroutine.

Initialization of modes 3 and 4

The following steps describe how to initialize HSC0 for single-phase up/down counter with external direction control (modes 3 and 4):

1. Use the first scan memory bit to call a subroutine in which the initialization operation is performed. Since you use a subroutine call, subsequent scans do not make the call to the subroutine, which reduces scan time execution and provides a more structured program.
2. In the initialization subroutine, load SMB195 and SMB37 according to the desired control operation.
   For example: SMB195=0 indicates counting mode
   SMB37 = 16#F8 produces the following results:
   - Enables the counter
   - Writes a new current value
   - Writes a new preset value
   - Sets the initial direction of the HSC to count up
   - Sets the reset input to be active high
3. Execute the HDEF instruction with the HSC input set to 0 and the MODE input set to one of the following:
   - Mode 3 for no external reset
   - Mode 4 for external reset
4. Load SMD38 (double-word-sized value) with the desired current value (load with 0 to clear it).
5. Load SMD42 (double-word-sized value) with the desired preset value.
6. In order to capture the current-value-equal-to-preset event, program an interrupt by attaching the CV = PV interrupt event (event 12) to an interrupt routine. See the section that discusses the Interrupt instructions for complete details on interrupt processing.
7. In order to capture direction changes, program an interrupt by attaching the direction changed interrupt event (event 27) to an interrupt routine.
8. In order to capture an external reset event, program an interrupt by attaching the external reset interrupt event (event 28) to an interrupt routine.
9. Execute the global interrupt enable instruction (ENI) to enable interrupts.
10. Execute the HSC instruction to cause the CPU to program HSC0.
11. Exit the subroutine.

Initialization of modes 6 and 7

The following steps describe how to initialize HSC0 for two-phase up/down counter with up/down clocks (modes 6 and 7):

1. Use the first scan memory bit to call a subroutine in which the initialization operations are performed. Since you use a subroutine call, subsequent scans do not make the call to the subroutine, which reduces scan time execution and provides a more structured program.
2. In the initialization subroutine, load SMB195 and SMB37 according to the desired control operation.
   For example: SMB195=0 indicates counting mode
   SMB37 = 16#F8 produces the following results:
   - Enables the counter
   - Writes a new current value
   - Writes a new preset value
   - Sets the initial direction of the HSC to count up
   - Sets the reset input to be active high
3. Execute the HDEF instruction with the HSC input set to 0 and the MODE set to one of the following:
   - Mode 6 for no external reset
   - Mode 7 for external reset
4. Load SMD38 (double-word-sized value) with the desired current value (load with 0 to clear it).
5. Load SMD42 (double-word-sized value) with the desired preset value.
6. In order to capture the current-value-equal-to-preset event, program an interrupt by attaching the CV = PV interrupt event (event 12) to an interrupt routine. See the section on interrupts.
7. In order to capture direction changes, program an interrupt by attaching the direction changed interrupt event (event 27) to an interrupt routine.
8. In order to capture an external reset event, program an interrupt by attaching the external reset interrupt event (event 28) to an interrupt routine.
9. Execute the global interrupt enable instruction (ENI) to enable interrupts.
10. Execute the HSC instruction to cause the CPU to program HSC0.
11. Exit the subroutine.

Initialization of modes 9 and 10

The following steps describe how to initialize HSC0 as an AB quadrature phase counter (modes 9 and 10):

1. Use the first scan memory bit to call a subroutine in which the initialization operations are performed. Since you use a subroutine call, subsequent scans do not make the call to the subroutine, which reduces scan time execution and provides a more structured program.
2. In the initialization subroutine, load SMB195 and SMB37 according to the desired control operation.
   For example (1x counting mode): SMB195=0 indicates counting mode
   SMB37 = 16#FC produces the following results:
   - Enables the counter
   - Writes a new current value
   - Writes a new preset value
   - Sets the initial direction of the HSC to count up
   - Sets the reset input to be active high

   Example (4x counting mode): SMB37 = 16#F8 produces the following results:
   - Enables the counter
   - Writes a new current value
   - Writes a new preset value
   - Sets the initial direction of the HSC to count up
   - Sets the reset input to be active high
3. Execute the HDEF instruction with the HSC input set to 0 and the MODE input set to one of the following:
   - Mode 9 for no external reset
   - Mode 10 for external reset
4. Load SMD38 (double-word-sized value) with the desired current value (load with 0 to clear it).
5. Load SMD42 (double-word-sized value) with the desired preset value.
6. In order to capture the current-value-equal-to-preset event, program an interrupt by attaching the CV = PV interrupt event (event 12) to an interrupt routine. See the section on enabling interrupts (ENI) for complete details on interrupt processing.
7. In order to capture direction changes, program an interrupt by attaching the direction changed interrupt event (event 27) to an interrupt routine.
8. In order to capture an external reset event, program an interrupt by attaching the external reset interrupt event (event 28) to an interrupt routine.
9. Execute the global interrupt enable instruction (ENI) to enable interrupts.
10. Execute the HSC instruction to cause the CPU to program HSC0.
11. Exit the subroutine.

Change direction in modes 0 and 1

The following steps describe how to configure HSC0 for change direction for single-phase counter with internal direction (modes 0 and 1):

1. Load SMB37 to write the desired direction:
   SMB37 = 16#90
   - Enables the counter
   - Sets the direction of the HSC to count down

   SMB37 = 16#98
   - Enables the counter
   - Sets the direction of the HSC to count up
2. Execute the HSC instruction to cause the CPU to program HSC0.

Loading a new current value (any mode)

The following steps describe how to change the counter current value of HSC0 (any mode):

1. Load SMB37 to write the desired current value:
   SMB37 = 16#C0
   - Enables the counter
   - Writes the new current value
2. Load SMD38 (double-word-sized value) with the desired current value (load with 0 to clear it).
3. Execute the HSC instruction to cause the CPU to program HSC0.

Loading a new preset value (any mode)

The following steps describe how to change the preset value of HSC0 (any mode):

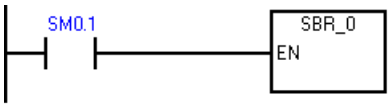
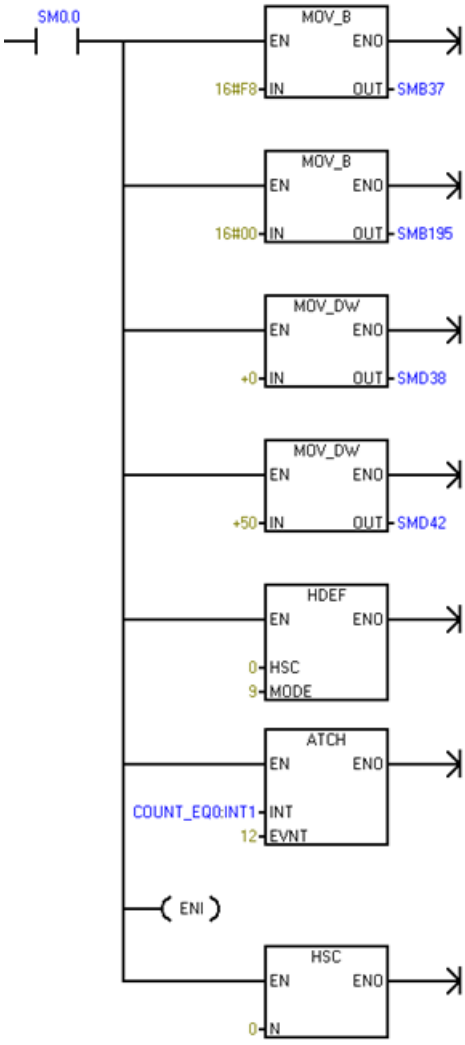
1. Load SMB37 to write the desired preset value:
   SMB37 = 16#A0
   - Enables the counter
   - Writes the new preset value
2. Load SMD42 (double-word-sized value) with the desired preset value.
3. Execute the HSC instruction to cause the CPU to program HSC0.

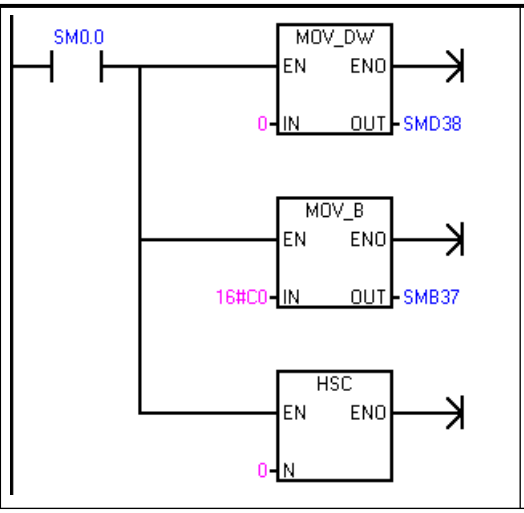
Disabling a high-speed counter (any mode)

The following steps describe how to disable the HSC0 high-speed counter (any mode):

1. Load SMB37 to disable the counter:
   SMB37 = 16#00
   - Disables the counter
2. Execute the HSC instruction to disable the counter.

Example: high-speed counter instruction

LAD			STL
MAIN		On the first scan, call SBR_0.	**Network 1** **LD SM0.1** **CALL SBR_0**
SBR0		On the first scan, configure HSC0: 1. Enable the counter – Write a new current value. – Write a new preset value. – Set the initial direction to count up. – Select the reset input to be active high. – Select 4x mode. 2. Configure HSC0 for AB quadrature phase with reset input. 3. Clear the current value of HSC0. 4. Set the HSC0 preset value to 50. 5. Attach event 12 to interrupt routine INT_0. The interrupt is executed when HSC0 current value = preset value. 6. Global interrupt enable 7. Configure HSC0.	**Network 1** **LD SM0.0** **MOVB 16#F8, SMB37** **MOVB 16#00, SMB195** **MOVD +0, SMD38** **MOVD +50, SMD42** **HDEF 0, 9** **ATCH** **COUNT_EQ0:INT1, 12** **ENI** **HSC 0**

LAD			STL
INT0		Program HSC0: 1. Clear the current value of HSC0. 2. Select to write only a new current and leave HSC0 enabled. 3. Configure HSC0.	**Network 1** **LD SM0.0** **MOVD +0, SMD38** **MOVB 16#C0, SMB37** **HSC 0**

Reference information

Refer to the following sections for further information:

- High-speed counter instructions (Page 325)
- High-speed counter summary (Page 327)
- High-speed counter programming (Page 330)
- Interrupt instructions (Page 393)

10.6.7 Example initialization sequences for high-speed frequency mode

Initialization of modes 0

The following steps describe how to initialize HSC0 for single-phase up/down frequency meter with internal direction (modes 0):

1. Use the first scan memory bit to call a subroutine in which the initialization operation is performed. Since you use a subroutine call, subsequent scans do not make the call to the subroutine, which reduces scan time execution and provides a more structured program.
2. In the initialization subroutine, load SMB195 and SMB37 according to the desired control operation.
   For example: SMB195 = 1 indicates frequency measurement mode
   SMB37 = 16#F8 produces the following results:
   - Enables the counter
   - Sets the direction to count up
3. Execute the HDEF instruction with the HSC input set to 0 and the MODE input set to 0 for no external reset.
4. Execute the HSC instruction to cause the CPU to program HSC0.
5. Exit the subroutine.

Initialization of modes 3

The following steps describe how to initialize HSC0 for single-phase up/down counter with external direction control (modes 3):

1. Use the first scan memory bit to call a subroutine in which the initialization operation is performed. Since you use a subroutine call, subsequent scans do not make the call to the subroutine, which reduces scan time execution and provides a more structured program.
2. In the initialization subroutine, load SMB195 and SMB37 according to the desired control operation.
   For example: SMB195 = 1 indicates frequency measurement mode
   SMB37 = 16#F8 produces the following results:
   – Enables the counter
3. Execute the HDEF instruction with the HSC input set to 0 and the MODE input set to 3 for no external reset.
4. In order to capture direction changes, program an interrupt by attaching the direction changed interrupt event (event 27) to an interrupt routine.
5. Execute the global interrupt enable instruction (ENI) to enable interrupts.
6. Execute the HSC instruction to cause the CPU to program HSC0.
7. Exit the subroutine.

Initialization of modes 6

The following steps describe how to initialize HSC0 for two-phase up/down counter with up/down clocks (modes 6):

1. Use the first scan memory bit to call a subroutine in which the initialization operations are performed. Since you use a subroutine call, subsequent scans do not make the call to the subroutine, which reduces scan time execution and provides a more structured program.
2. In the initialization subroutine, load SMB195 and SMB37 according to the desired control operation.
   For example: SMB195 = 1 indicates frequency measurement mode
   SMB37 = 16#F8 produces the following results:
   – Enables the counter
3. Execute the HDEF instruction with the HSC input set to 0 and the MODE set to 6 for no external reset
4. In order to capture direction changes, program an interrupt by attaching the direction changed interrupt event (event 27) to an interrupt routine.
5. Execute the global interrupt enable instruction (ENI) to enable the interrupts.
6. Execute the HSC instruction to cause the CPU to program HSC0.
7. Exit the subroutine.

Initialization of modes 9

The following steps describe how to initialize HSC0 as an AB quadrature phase counter (modes 9):

1. Use the first scan memory bit to call a subroutine in which the initialization operations are performed. Since you use a subroutine call, subsequent scans do not make the call to the subroutine, which reduces scan time execution and provides a more structured program.
2. In the initialization subroutine, load SMB195 and SMB37 according to the desired control operation.
   For example: SMB195 = 1 indicates frequency measurement mode
   SMB37 = 16#FC produces the following results:
   - Enables the counter

   Example (4x counting mode): SMB37 = 16#F8 produces the following results:
   - Enables the counter
3. Execute the HDEF instruction with the HSC input set to 0 and the MODE input set to 9 for no external reset.
4. In order to capture direction changes, program an interrupt by attaching the direction changed interrupt event (event 27) to an interrupt routine.
5. Execute the global interrupt enable instruction (ENI) to enable interrupts.
6. Execute the HSC instruction to cause the CPU to program HSC0.
7. Exit the subroutine.

Change direction in modes 0

The following steps describe how to configure HSC0 for change direction for single-phase counter with internal direction (modes 0):

1. Load SMB37 to write the desired direction:
   SMB37 = 16#90
   - Enables the counter
   - Sets the direction of the HSC to count down

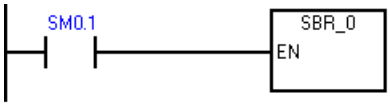
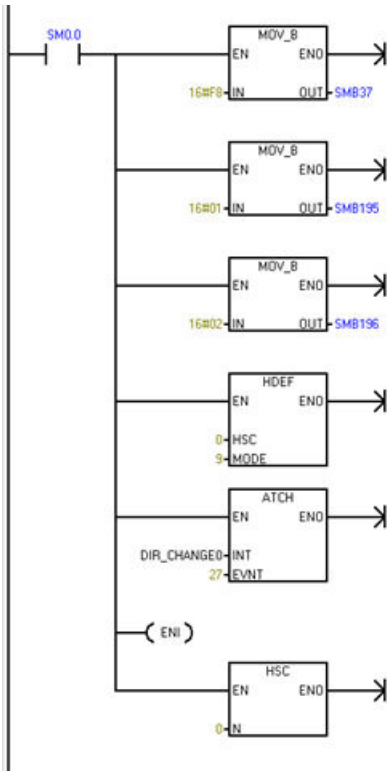
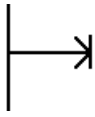
   SMB37 = 16#98
   - Enables the counter
   - Sets the direction of the HSC to count up
2. Execute the HSC instruction to cause the CPU to program HSC0.

Disabling a high-speed counter (any mode)

The following steps describe how to disable the HSC0 high-speed counter (any mode):

1. Load SMB37 to disable the counter:
   SMB37 = 16#00
   - Disables the counter
2. Execute the HSC instruction to disable the counter.

Example: high-speed counter instruction

LAD			STL
MAIN		On the first scan, call SBR_0.	**Network 1** **LD SM0.1** **CALL SBR_0**
SBR0		On the first scan, configure HSC0: 1. Enable the counter – Select 4x mode. 2. Set the mode type as "Frequency". 3. Set the sampling period as 10 ms. 4. Configure HSC0 for AB quadrature phase with reset input. 5. Attach event 27 to interrupt routine DIR_CHANGE0. The interrupt is executed when direction changes. 6. Interrupt enable 7. Configure HSC0.	**Network 1** **LD SM0.0** **MOVB 16#F8, HSC0_Ctrl:SMB37** **MOVB 16#01, SMB195** **MOVB 16#02, SMB196** **HDEF 0, 9** **ATCH DIR_CHANGE0, 27** **ENI** **HSC 0**
INT0		User defined	

Reference information

Refer to the following sections for further information:

- High-speed counter instructions (Page 325)
- High-speed counter summary (Page 327)
- High-speed counter programming (Page 330)
- Interrupt instructions (Page 393)

10.7 Pulse output

10.7.1 Pulse output instruction (PLS)

The Pulse output (PLS) instruction controls the Pulse train output (PTO) and Pulse width modulation (PWM) functions available on the high-speed outputs (Q0.0, Q0.1,Q0.3, Q0.2, and Q0.4).

When using PWM, you can use an optional wizard to create the PWM instructions.

LAD / FBD	STL	Description
PLS EN ENO N	**PLS N**	You can use the PLS instruction to create up to five PTO or PWM operations. PTO allows the user to control the frequency and number of pulses for a square wave (50% duty cycle) output. PWM allows the user control of a fixed cycle time output with a variable duty cycle.

Error conditions with ENO = 0	SM bits affected
• 0005H: Simultaneous HSC/PLS • 000DH: Attempt to redefine pulse output while it is active • 000EH: Number of PTO profile segments was set to 0 • 0017H: Attempt to assign resource for PTO/PWM that is already assigned to motion control • 001BH: Attempt to change time base on enabled PWM • 0090H: N is not 0, 1, 2, 3 or 4. • 0091H: Range error	None

Input / output	Data type	Operand
N (channel)	WORD	Constant: 0 (= Q0.0), 1 (= Q0.1), 2 (= Q0.3), 3(=Q0.2), or 4(=Q0.4)

The CPU has five PTO/PWM generators (PLS0, PLS1, PLS2, PLS3 and PLS4) that create either a high-speed pulse train or a pulse width modulated waveform. PLS0 is assigned to digital output point Q0.0, PLS1 is assigned to digital output point Q0.1, PLS2 is assigned to digital output point Q0.3, PLS3 is assigned to digital output point Q0.2 and PLS4 is assigned to digital output point Q0.4. A designated special memory (SM) location stores the following data for each generator: a PTO status byte (8-bit value), a control byte (8-bit value), a cycle time or frequency (unsigned 16-bit value), a pulse width value (unsigned 16-bit value), and a pulse count value (unsigned 32-bit value).

The PTO/PWM generators and the process image register share the use of Q0.0, Q0.1, Q0.3, Q0.2 and Q0.4. When a PTO or PWM function is active on Q0.0, Q0.1, Q0.3, Q0.2 and Q0.4, the PTO/PWM generator has control of the output, and normal use of the output point is inhibited. The output waveform is not affected by the state of the process image register, the forced value of the point, or the execution of immediate output instructions. When the

PTO/PWM generator is inactive, control of the output reverts to the process image register. The process image register determines the initial and final state of the output waveform, causing the waveform to start and end at a high or low level.

Note

PTO/PWM through the PLS instruction is not possible if the selected output point is already configured for use with motion control through use of the Motion wizard.

The PTO/PWM outputs must have a minimum load of at least 10% of rated load to provide crisp transitions from off to on, and from on to off.

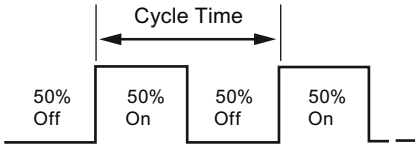
Before enabling PTO/PWM operation, set the value of the process image register for Q0.0, Q0.1, Q0.3, Q0.2 and Q0.4 to 0.

Default values for all control bits, cycle time/frequency, pulse width, and pulse count values are 0.

A running PLS instruction can be overrided by another PLS instruction. When the override is triggered, the CPU will change the speed or pulse count directly to the target speed or pulse count to reach to the new position.

10.7.2 Pulse train output (PTO)

PTO provides a square wave with a 50% duty cycle output for a specified number of pulses at a specified frequency. Refer to the figure below. PTO can produce either a single train of pulses or multiple trains of pulses using a pulse profile. You specify the number of pulses and the frequency:



- Number of pulses: 1 to 2,147,483,647
- Frequency:
  - 1 to 200,000 Hz (multiple-segment)
  - 1 to 65,535 Hz (single-segment)

Use the following formula to convert from cycle time to frequency:

F = 1 / CT

where:

F	Frequency (Hz)
CT	Cycle time (seconds)

See the following table for pulse count and frequency limitations:

Table 10-15 Pulse count and frequency in the PTO function

Pulse count / frequency	Reaction
Frequency < 1 Hz	Frequency defaults to 1 Hz
Frequency > 200,000 Hz	Frequency defaults to 200,000 Hz

Pulse count / frequency	Reaction
Pulse count = 0	Pulse count defaults to 1 pulse
Pulse count > 2,147,483,647	Pulse count defaults to 2,147,483,647 pulses

Note

When using a PTO with very short cycle times (high frequencies), you should take into account the switching delay specifications for the output points and how the switching delay can affect the duty cycle. See Appendix A for the digital output switching delay for your CPU.

The PTO function allows the "chaining" or "pipelining" of pulse trains. When the active pulse train is complete, the output of a new pulse train begins immediately. This allows continuity between subsequent output pulse trains.

The PTO function allows the “overriding” of single-segment. The configured new pulse train will override the current output pulse train.

Single-Segment pipelining of PTO pulses

In single-segment pipelining, you are responsible for updating the SM locations for the next pulse train. After the initial PTO segment has started, you must immediately modify the SM locations with the parameters of the second waveform. After you update the SM values, execute the PLS instruction again. The PTO function holds the attributes of the second pulse train in a pipeline until it completes the first pulse train. The PTO function can store only one entry at a time in the pipeline. When the first pulse train completes, the output of the second waveform begins, and you can store a new pulse train specification in the pipeline. You can then repeat this process to set up the characteristics of the next pulse train. Attempting to load the pipeline while it is still full results in the PTO Overflow bit (SM66.6, SM76.6, SM566.6 or SM580.6) being set and the instruction being ignored.

Smooth transitions between pulse trains occur unless the active pulse train completes before a new pulse train setup is captured by the execution of the PLS instruction.

Note

In single-segment pipelining, the frequency has an upper limit of 65,535 Hz. If a higher frequency is needed (up to 200,000 Hz), multiple-segment pipelining must be used.

Multiple-Segment pipelining of PTO pulses

In multiple-segment pipelining, the S7-200 SMART V3.0 automatically reads the characteristics of each pulse train segment from a profile table located in V memory. The SM locations used in this mode are the control byte, the status byte, and the starting V memory offset of the profile table (SMW168, SMW178, SMW578, SMW592 or SMW2012). Execution of the PLS instruction starts the multiple segment operation.

Each segment entry is 12 bytes in length and is composed of a 32 bit starting frequency, a 32 bit ending frequency, and a 32-bit pulse count value. The table below shows the format of the profile table configured in V memory.

The PTO generator automatically increases or decreases the frequency linearly from the starting frequency to the ending frequency. The frequency is increased or decreased by a constant value at a constant rate. Once the number of pulses reaches the specified pulse count, the next PTO segment is loaded. This sequence repeats until it reaches the end of the profile. A segment's time duration should be greater than 500 microseconds. If the time duration is too small, the CPU may not have enough time to calculate the next PTO segment values. If the next segment cannot be calculated in time, then the PTO pipeline underflow bit (SM66.6, SM76.6, SM566.6, SM580.6 and SM2000.6) is set to "1" and the PTO operation terminated.

While the PTO profile is operating, the number of the currently active segment is available in SMB166, SMB176, SMB576, SMB590 or SMB2010.

Table 10-16 Profile table format for multiple-segment PTO operation[1]

Byte offset	Segment	Description of table entries
0		Number of segments: 1 to 255[2]
1	#1	Starting Frequency (1 to 200,000 Hz)
5		Ending Frequency (1 to 200,000 Hz)
9		Pulse count (1 to 2,147,483,647)
13	#2	Starting Frequency (1 to 200,000 Hz)
17		Ending Frequency (1 to 200,000 Hz)
21		Pulse count (1 to 2,147,483,647)
(Continues)	#3	(Continues)

[1] Entering a profile offset and number of segments that places any part of the profile table outside of V memory generates a non-fatal error. The PTO function does not generate a PTO output.

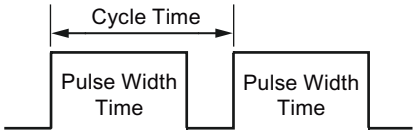
[2] Entering a value of 0 for the number of segments generates a non-fatal error. No PTO output is generated.

Single-Segment overriding of PTO pulses

In single-segment overriding, you are responsible for updating the SM locations for the pulse train that will execute override function. After the initial PTO segment has started, you must immediately modify the SM locations with the parameters of the second waveform. After you update the SM values, execute the PLS instruction again when needed to trigger the override.

10.7.3 Pulse width modulation (PWM)

PWM provides four channels that allow a fixed cycle time output with a variable duty cycle. Refer to the figure below. You can specify the cycle time and the pulse width in either microsecond or millisecond increments:



- Cycle time: 5 µs to 65,535 µs or 2 ms to 65,535 ms
- Pulse width time: 0 µs to 65,535 µs or 0 ms to 65,535 ms

As shown in the following table, setting the pulse width equal to the cycle time (which makes the duty cycle 100 percent) turns the output on continuously. Setting the pulse width to 0 (which makes the duty cycle 0 percent) turns the output off.

Note

When using a PWM with very short cycle times, you should take into account the switching delay specifications for the output points and how the switching delay can affect the pulse width time. See Appendix A for the digital output switching delay for your CPU.

Pulse width time and cycle time and reactions in the PWM function

Pulse width time / cycle time	Reaction
Pulse width time >= Cycle time value	The duty cycle is 100%: the output is turned on continuously.
Pulse width time = 0	The duty cycle is 0%: the output is turned off continuously.
Cycle time < 2 time units	The cycle time defaults to two time units.

Changing the characteristics of a PWM waveform

You can only use synchronous updates to change the characteristics of a PWM waveform. With a synchronous update, the change in the waveform characteristics occurs on a cycle boundary, providing a smooth transition.

10.7.4 Using SM locations to configure and control the PTO/PWM operation

The PLS instruction reads the data stored in the specified SM memory locations and programs the PTO/PWM generator accordingly. SMB67 controls PTO0 or PWM0, SMB77 controls PTO1 or PWM1, SMB567 controls PTO2 or PWM2, SMB581 controls PTO3 or PWM3 and SMB2001 controls PTO4 or PWM4. The "SM locations for the PTO/PWM control registers" table (the first table below) describes the registers used to control the PTO/PWM operation. You can use the "PTO/PWM control byte reference" table (the second table below) as a quick reference to determine the value to place in the PTO/PWM control register to invoke the desired operation.

You can change the characteristics of a PTO or PWM waveform by modifying the locations in the SM area (including the control byte) and then executing the PLS instruction. You can disable the generation of a PTO or PWM waveform at any time by writing 0 to the PTO/PWM enable bit of the control byte (SM67.7, SM77.7, SM567.7, SM581.7, SM2001.7) and then executing the PLS instruction. The output point immediately reverts back to process image register control.

If you disable the PTO or PWM operation while the operation is producing a pulse, that pulse internally completes its full cycle time duration. However, the pulse is not present at the output point because, at that time, the process image register regains control of the output. Your program can enable the pulse generator again with no time delay as long as the following is true: the pulse mode (PTO or PWM) being enabled is the same mode that was disabled. An error may occur if your program first disables a PTO and then enables a PWM on the same output channel or if your program first disables a PWM and then enables a PTO.

The PTO Idle bit in the status byte (SM66.7, SM76.7, SM566.7, SM580.7 or SM2006.7) is provided to indicate the completion of the programmed pulse train. In addition, an interrupt routine can be invoked upon the completion of a pulse train. (Refer to the descriptions of the Interrupt instructions (Page 393).) If you are using the single segment operation, the interrupt routine is invoked upon the completion of each PTO. For example, if a second PTO is loaded into the pipeline, the PTO function invokes the interrupt routine upon the completion of the first PTO, and again upon the completion of the second PTO that was loaded into the pipeline. When using the multiple segment operation, the PTO function invokes the interrupt routine upon completion of the profile table.

The following conditions set the bits of the status byte (SMB66, SMB76, SMB566, SMB580, and SMB2006):

- If an "Add Error" occurs in the pulse generator that results in an invalid frequency value, the PTO function terminates and the Delta Calculation Error bit (SM66.4, SM76.4, SM566.4, SM580.4 and SM2006.4) is set to 1. The output reverts to image register control. To correct this issue, try adjusting the PTO profile parameters.
- Manually disabling a PTO profile in progress sets the PTO Profile Disabled bit (SM66.5, SM76.5, SM566.5, SM580.5 or SM2006.5) to 1.
- The PTO/PWM overflow/underflow bit (SM66.6, SM76.6, SM566.6, SM580.6 or SM2006.6) is set to 1 if either of these situations occur:
  - An attempt is made to load the pipeline while it is full; this is an overflow condition.
  - A PTO profile segment is too short to allow the CPU to calculate the next segment, and an empty pipeline is transferred; this is an underflow condition, and the output reverts to image register control.
- You must clear the PTO/PWM overflow/underflow bit manually after it is set to detect subsequent overflows. The transition to RUN mode initializes this bit to 0.

Note

- Ensure that you understand the definition of the PTO/PWM mode select bit (SM67.6, SM77.6, SM567.6, SM581.6 and SMB2000.6). The bit definition may not be the same as some legacy products that support a Pulse instruction. In the S7-200 SMART, the user selects PTO or PWM mode with the following definition: 0 = PWM, 1 = PTO.
- When you load a cycle time/frequency (SMW68, SMW78, SMW568, SMW582 or SMW2002), pulse width (SMW70, SMW80, SMW570, SMW584 or SMW2004), or pulse count (SMD72, SMD82, SMD572, SMD586 or SMD2006), also set the appropriate update bits in the control register before you execute the PLS instruction.
- For a multiple segment pulse train operation, you must also load the starting offset (SMW168, SMW178, SMW578, SMW592 or SMW2012) of the profile table and the profile table values before you execute the PLS instruction.
- If you attempt to change the time base of a PWM output while the PWM is executing, the request is ignored and a non-fatal error (0x001B - ILLEGAL PWM TIMEBASE CHG) is created.

Table 10-17 SM locations for the PTO/PWM control registers

Q0.0	Q0.1	Q0.3	Q0.2	Q0.4	Status bits
SM66.4	SM76.4	SM566.4	SM580.4	SM2000.4	PTO delta calculation error (due to an add error): • 0 = No error • 1 = Aborted due to error
SM66.5	SM76.5	SM566.5	SM580.5	SM2000.5	PTO profile disabled (due to user command): • 0 = Profile not manually disabled • 1 = User disabled profile
SM66.6	SM76.6	SM566.6	SM580.6	SM2000.6	PTO/PWM pipeline overflow/underflow: • 0 = No overflow/underflow • 1 = Overflow/underflow
SM66.7	SM76.7	SM566.7	SM580.7	SM2000.7	PTO idle: • 0 = In progress • 1 = PTO idle
Q0.0	**Q0.1**	**Q0.3**	**Q0.2**	**Q0.4**	**Control bits**
SM67.0	SM77.0	SM567.0	SM581.0	SM2001.0	PTO/PWM update the frequency/cycle time: • 0 = No update • 1 = Update frequency/cycle time
SM67.1	SM77.1	SM567.1	SM581.1	SM2001.1	PWM update the pulse width time: • 0 = No update • 1 = Update pulse width
SM67.2	SM77.2	SM567.2	SM581.2	SM2001.2	PTO update the pulse count value: • 0 = No update • 1 = Update pulse count
SM67.3	SM77.3	SM567.3	SM581.3	SM2001.3	The PWM time base: • 0 = 1 µs/tick • 1 = 1 ms/tick
SM67.4	SM77.4	SM567.4	SM581.4	SM2001.4	PTO override: • 0 = Disable • 1 = Enable
SM67.5	SM77.5	SM567.5	SM581.5	SM2001.5	PTO single/multiple segment operation: • 0 = Single • 1 = Multiple
SM67.6	SM77.6	SM567.6	SM581.6	SM2001.6	PTO/PWM mode select: • 0 = PWM • 1 = PTO
SM67.7	SM77.7	SM567.7	SM581.7	SM2001.7	PWM enable: • 0 = Disable • 1 = Enable
Q0.0	**Q0.1**	**Q0.3**	**Q0.2**	**Q0.4**	**Other registers**
SMW68	SMW78	SMW568	SMW582	SMW2002	PTO frequency or PWM cycle time value: 1 to 65,535 Hz (PTO); 2 to 65,535 (PWM)

SMW70	SMW80	SMW570	SMW584	SMW2004	PWM pulse width value: 0 to 65,535
SMD72	SMD82	SMD572	SMD586	SMD2006	PTO pulse count value: 1 to 2,147,483,647
SMB166	SMB176	SMB576	SMB590	SMB2010	Number of the segment in progress: Multiple-segment PTO operation only
SMW168	SMW178	SMW578	SMW592	SMW2012	Starting location of the profile table (byte offset from V0): Multiple-segment PTO operation only

Table 10-18 PTO/PWM control byte reference

		Result of executing the PLS instruction						
Control register (Hex value)	**Enable**	**Select mode**	**PTO Segment operation**	**Override**	**Time base**	**Pulse count**	**Pulse width**	**Cycle time / frequency**
16#80	Yes	PWM		No	1 µs/cycle			
16#81	Yes	PWM		No	1 µs/cycle			Update cycle time
16#82	Yes	PWM		No	1 µs/cycle		Update	
16#83	Yes	PWM		No	1 µs/cycle		Update	Update cycle time
16#88	Yes	PWM		No	1 ms/cycle			
16#89	Yes	PWM		No	1 ms/cycle			Update cycle time
16#8A	Yes	PWM		No	1 ms/cycle		Update	
16#8B	Yes	PWM		No	1 ms/cycle		Update	Update cycle time
16#C0	Yes	PTO	Single	No				
16#C1	Yes	PTO	Single	No				Update frequency
16#C4	Yes	PTO	Single	No		Update		
16#C5	Yes	PTO	Single	No		Update		Update frequency
16#D0	Yes	PTO	Single	Yes				
16#D1	Yes	PTO	Single	Yes				Update frequency
16#D4	Yes	PTO	Single	Yes		Update		
16#D5	Yes	PTO	Single	Yes		Update		Update frequency
16#E0	Yes	PTO	Multiple	No				

10.7.5 Calculating the profile table values

The multiple-segment pipelining capability of the PTO generators can be useful in many applications, particularly in stepper motor control.

For example, you can use PTO with a pulse profile to control a stepper motor through a simple ramp up (acceleration), run (no acceleration), and ramp down (deceleration) sequence. More complicated sequences can be created by defining a pulse profile that consists of up to 255 segments, with each segment corresponding to a ramp up, run, or ramp down operation.

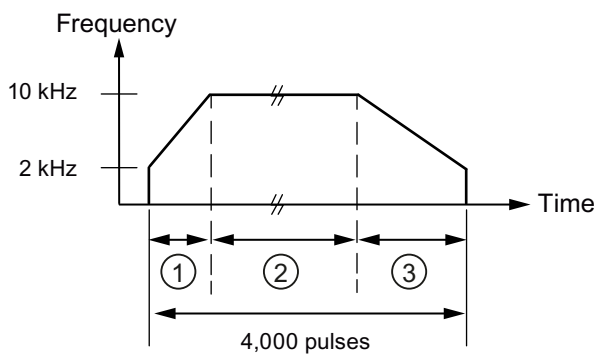
The figure below illustrates sample profile table values required to generate an output waveform:

- Segment 1: Accelerates a stepper motor
- Segment 2: Operates the motor at a constant speed
- Segment 3: Decelerates the motor

To achieve the desired number of motor revolutions for this example, the PTO generator requires the following values:

- Starting and final pulse frequencies of 2 kHz
- A maximum pulse frequency of 10 kHz
- 4000 pulses

During the acceleration portion of the output profile, the output wave form should reach maximum pulse frequency in approximately 200 pulses. The output wave form should complete the deceleration portion of the profile in approximately 400 pulses.



① Segment 1: 200 pulses
② Segment 2: 3400 pulses
③ Segment 3: 400 pulses

The following table lists the values for generating the example waveform. The profile table, for this example, is in V memory and starts at VB500. You can use any block of V memory that is available for a PTO profile table. You can include instructions in your program to load these values into V memory, or you can define the values of the profile in the data block.

Table 10-19 Profile table values

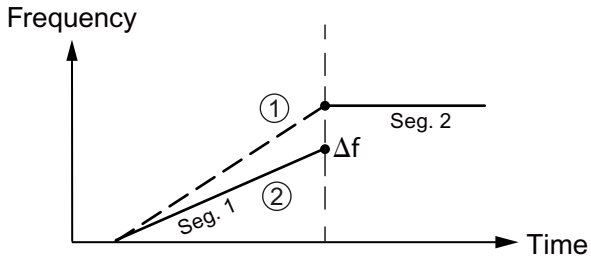
Address	Value	Description	
VB500	3	Total number of segments	
VD501	2,000	Starting frequency (Hz)	Segment 1
VD505	10,000	Ending frequency (Hz)	
VD509	200	Number of pulses	

Address	Value	Description	
VD513	10,000	Starting frequency (Hz)	Segment 2
VD517	10,000	Ending frequency (Hz)	
VD521	3,400	Number of pulses	
VD525	10,000	Starting frequency (Hz)	Segment 3
VD529	2,000	Ending frequency (Hz)	
VD533	400	Number of pulses	

The PTO generator begins by running Segment 1. After the PTO generator reaches the required number of pulses for Segment 1, it automatically loads Segment 2. This continues until the last segment. After the number of pulses for the last segment is reached, the S7-200 SMART CPU disables the PTO generator.

For each segment of the PTO profile, the pulse train begins at the starting frequency assigned in the table. The PTO generator increases or decreases the frequency at a constant rate to achieve the ending frequency with the correct number of pulses. However, the PTO generator limits the frequency to the starting and ending frequencies specified in the table.

The PTO generator performs repeated additions to the working frequency to create a linear change in frequency over time. The constant value added to the frequency has a limited resolution. This limited resolution can introduce some truncation error into the resulting frequency. Thus, the PTO generator does not guarantee that the pulse train frequency can reach the ending frequency that was specified for that segment. In the figure below, you can see that the truncation error affects the accelerating PTO frequency. The output should be measured to verify that the frequency is within an acceptable frequency range.



① Desired frequency plot
② Actual frequency plot

If the frequency difference (Δf) between the end of a segment and the beginning of the next is not acceptable, try adjusting the ending frequency to compensate for the difference. This adjustment might be an iterative process to get the output within an acceptable frequency range.

Note that changes in segment parameters affect the time it takes the PTO to complete. You can use the equation for the time duration of the segment, found later in this section of the manual, to see what effect this has on the timing. An accurate segment duration time can require some flexibility in the value of the ending frequency or the number of pulses for a given segment.

While the simplified example above is useful as an introduction, real applications can require more complicated waveform profiles. Remember that you can only assign frequencies as an integer number of Hz and perform the frequency modification at a constant rate. The S7-200 SMART CPU selects that constant rate and that rate can be different for each segment.

For legacy projects that were developed in terms of cycle time, instead of frequency, you can use the following formulas to convert to frequency:

$$CT_{Final} = CT_{Initial} + (\Delta CT * PC)$$

$$F_{Initial} = 1 / CT_{Initial}$$

$$F_{Final} = 1 / CT_{Final}$$

where:

$CT_{Initial}$	Starting cycle time (s) for this segment
ΔCT	Delta cycle time (s) of this segment
PC	Quantity of pulses in this segment
CT_{Final}	Ending cycle time (s) for this segment
$F_{Initial}$	Starting frequency (Hz) for this segment
F_{Final}	Ending frequency (Hz) for this segment

The acceleration (or deceleration) and time duration of a given PTO profile segment can be useful in the process of determining correct profile table values. Use the following formulas to calculate the length of time, as well as the acceleration for a given profile:

$$\Delta F = F_{Final} - F_{Initial}$$

$$T_s = PC / (F_{min} + (| \Delta F | / 2))$$

$$A_s = \Delta F / T_s$$

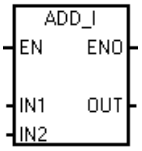
where:

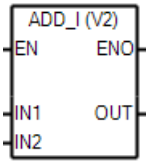
T_s	Time duration (s) of this segment
A_s	Frequency acceleration (Hz/s) of this segment
PC	Quantity of pulses in this segment
F_{min}	Minimum frequency (Hz) for this segment
ΔF	Delta (total change in) frequency (Hz) for this segment

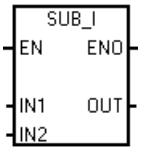
10.8 Math

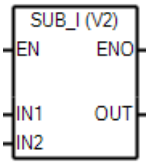
10.8.1 Add, subtract, multiply, and divide

LAD / FBD	STL	Description

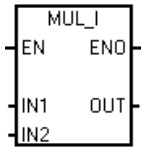

ADD_DI
ADD_R
	`+I IN1, IN2 OUT`
	`+D IN1, IN2 OUT`
	`+R IN1, IN2 OUT`
		The Add Integer instruction adds two 16-bit integers to produce a 16-bit result. The Add Double Integer instruction adds two 32-bit integers to produce a 32-bit result. The Add Real (+R) instruction adds two 32-bit real numbers to produce a 32-bit real number result.
		• LAD, FBD and STL: IN1 + IN2 = OUT
		• STL legacy: IN1 + OUT = OUT

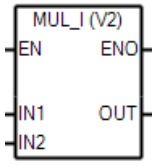

ADD_DI
ADD_R
	Legacy:
	`+I IN1, OUT`
	`+D IN1, OUT`
	`+R IN1, OUT`	

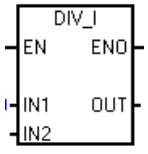

SUB_DI
SUB_R
	`-I IN1, IN2, OUT`
	`-D IN1, IN2, OUT`
	`-R IN1, IN2, OUT`
		The Subtract Integer instruction subtracts two 16-bit integers to produce a 16-bit result. The Subtract Double Integer (-D) instruction subtracts two 32-bit integers to produce a 32-bit result. The Subtract Real (-R) instruction subtracts two 32-bit real numbers to produce a 32-bit real number result.
		• LAD, FBD and STL: IN1 - IN2 = OUT
		• STL legacy: OUT - IN1 = OUT

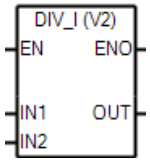
SUB_DI
 
SUB_R
	Legacy:
	`-I IN1, OUT`
	`-D IN1, OUT`
	`-R IN1, OUT`	

LAD / FBD	STL	Description


MUL_DI
MUL_R
	`*I IN1, IN2, OUT`
	`*D IN1, IN2, OUT`
	`*R IN1, IN2, OUT`
		The Multiply Integer instruction multiplies two 16-bit integers to produce a 16-bit result. The Multiply Double Integer instruction multiplies two 32-bit integers to produce a 32-bit result. The Multiply Real instruction multiplies two 32-bit real numbers to produce a 32-bit real number result.
		• LAD, FBD and STL: IN1 * IN2 = OUT
		• STL legacy:IN1 * OUT = OUT


MUL_DI
MUL_R
	Legacy:
	`*I IN1, OUT`
	`*D IN1, OUT`
	`*R IN1, OUT`	


DIV_DI
DIV_R
	`/I IN1, IN2, OUT`
	`/D IN1, IN2, OUT`
	`/R IN1, IN2, OUT`
		The Divide Integer instruction divides two 16-bit integers to produce a 16-bit result. (No remainder is kept.) Divide Double Integer instruction divides two 32-bit integers to produce a 32-bit result. (No remainder is kept.) The Divide Real (/R) instruction divides two 32-bit real numbers to produce a 32-bit real number result.
		• LAD, FBD and STL: IN1 / IN2 = OUT
		• STL legacy: OUT / IN1 = OUT


DIV_DI
DIV_R
	Legacy:
	`/I IN1, OUT`
	`/D IN1, OUT`
	`/R IN1, OUT`	

Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address • SM1.1 Overflow • SM1.3 Divide by zero	• SM1.0 Result of operation = zero • SM1.1 Overflow, illegal value generated during the operation, or illegal input • SM1.2 Negative result • SM1.3 Divide by zero

SM1.1 indicates overflow errors and illegal values. If SM1.1 is set, then the status of SM1.0 and SM1.2 is not valid and the original input operands are not altered. If SM1.1 and SM1.3 are not set, then the math operation has completed with a valid result and SM1.0 and SM1.2

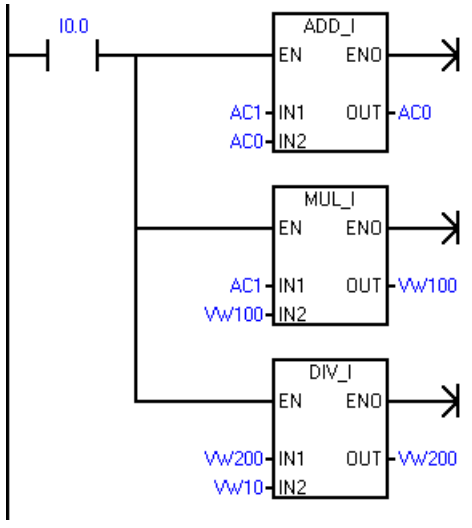
contain valid status. If SM1.3 is set during a divide operation, then the other math status bits are left unchanged.

Input / output	Data Type	Operand
IN1, IN2	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
	REAL[1]	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
OUT	INT	IW, QW, VW, MW, SMW, SW, LW, T, C, AC, *VD, *AC, *LD
	DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

[1] Real (or floating-point) numbers are represented in the format described in the ANSI/IEEE 754-1985 standard (single-precision). Refer to that standard for more information.

Example: Integer math instructions

LAD



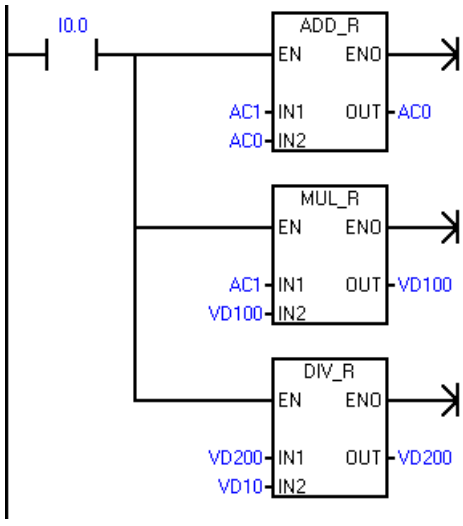
STL

```
Network
LD I0.0
+I AC1, AC0, AC0
*I AC1, VW100, VW100
/I VW200, VW10, VW200
```

Integer operations from the LAD example

	IN1		IN2		OUT
Add data	40	+	60	=	100
Data address	AC1		AC0		AC0
Multiply data	40	*	20	=	800
Data address	AC1		VW100		VW100
Divide data	4000	/	40	=	100
Data address	W200		VW10		VW200

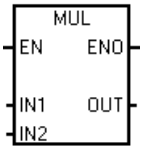
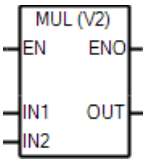
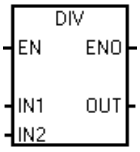
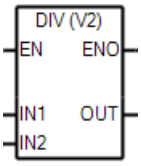
Example: Real math instructions

LAD	STL
	**Network 1** **LD I0.0** **+R AC1, AC0, AC0** ***R AC1, VD100, VD100** **/R VD200, VD10, VD200**

Real number operations from the LAD example

	IN1		IN2		OUT
Add data	4000.0	+	6000.0	=	10000.0
Data address	AC1		AC0		AC0
Multiply data	400.0	*	200.0	=	80000.0
Data address	AC1		VD100		VD100
Divide data	4000.0	/	41.0	=	97.5609
Data address	VD200		VD10		VD200

10.8.2 Multiply integers to double integer and divide integer with remainder

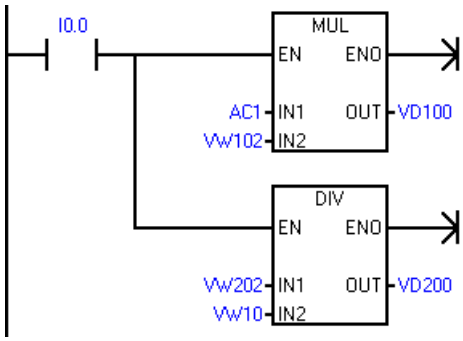
LAD / FBD	STL	Description
	**MUL IN1, IN2, OUT**	The multiply integer to double Integer instruction multiplies two 16-bit integers and produces a 32-bit product. In STL, the least-significant word (16 bits) of the 32-bit OUT is used as one of the factors. • LAD, FBD and STL: IN1 * IN2 = OUT • STL legacy: IN1 * OUT = OUT
	**Legacy**: **MUL IN1, OUT**	
	**DIV IN1, IN2, OUT**	The divide integer with remainder instruction divides two 16-bit integers and produces a 32-bit result consisting of a 16-bit remainder (the most-significant word) and a 16-bit quotient (the least-significant word). In STL, the least-significant word (16 bits) of the 32-bit OUT is used as the dividend. • LAD, FBD and STL: IN1 / IN2 = OUT • STL legacy: OUT / IN1 = OUT **Note**: OUT of this instruction doesn't have DB2 area support.
	**Legacy**: **DIV IN1, OUT**	

Non-fatal errors with ENO=0	SM bits affected [1]
• 0006H Indirect address • SM1.1 Overflow • SM1.3 Divide by zero	• SM1.0 Result of operation = zero • SM1.1 Overflow, illegal value generated during the operation, or illegal input • SM1.2 Negative result • SM1.3 Divide by zero

[1] For both of these instructions, SM bits indicate errors and illegal values. If SM1.3 (divide by zero) is set during a divide operation, then the other math status bits are left unchanged. Otherwise, all supported math status bits contain valid status upon completion of the math operation.

Input / output	Data type	Operand
IN1, IN2	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
OUT	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Example: MUL and DIV instructions

LAD	STL
	**Network 1** **LD I0.0** **MUL AC1, VW102, VD100** **DIV VW202, VW10, VD200**

[1] VD100 contains: VW100 and VW102, and VD200 contains: VW200 and VW202.

Real number operations from the LAD example

	IN1		IN2		OUT	
MUL data	400	*	200	=	80000	
Data address	AC1		VW102		VD100	
					quotient	remainder
DIV data	4000	/	41	=	97	23
Data address	VW202		VW10		VW200 VD200	VW202

10.8.3 Trigonometry, natural logarithm/exponential, and square root

Trigonometric and inverse trigonometric instructions

LAD / FBD	STL	Description
SIN EN ENO IN OUT	**SIN IN, OUT**	The sine (SIN), cosine (COS), and tangent (TAN) instructions evaluate the trigonometric function of the angle value IN and place the result in OUT. The arcsine (ARC_SIN), arccosine (ARC_COS), and arctangent (ARC_TAN) instructions evaluate the inverse trigonometric function of the angle value IN and place the result in OUT. The value range for IN of ARCSIN and ARCCOS is [-1,1]. The input angle value is measured in radians. • SIN (IN) = OUT • COS (IN) = OUT • TAN (IN) = OUT • ARCSIN (IN) = OUT • ARCCOS (IN) = OUT • ARCTAN (IN) = OUT To convert an angle from degrees to radians: Use the MUL_R (*R) instruction to multiply the angle in degrees by 1.745329E-2 (approximately by π/180). For the numeric functions instructions, SM1.1 is used to indicate overflow errors and illegal values. If SM1.1 is set, then the status of SM1.0 and SM1.2 is not valid and the original input operands are not altered. If SM1.1 is not set, then the math operation has completed with a valid result and SM1.0 and SM1.2 contain valid status.
COS EN ENO IN OUT	**COS IN, OUT**	
TAN EN ENO IN OUT	**TAN IN, OUT**	
ARC_SIN EN ENO IN OUT	**ARCSIN IN, OUT**	
ARC_COS EN ENO IN OUT	**ARCCOS IN, OUT**	
ARC_TAN EN ENO IN OUT	**ARCTAN IN, OUT**	

Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address • SM1.1 Overflow	• SM1.0 Result of operation = zero • SM1.1 Overflow, illegal value generated during the operation, or illegal input • SM1.2 Negative result

Input / outputs	Data type	Operand
IN	REAL[1]	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
OUT	REAL[1]	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

[1] Real (or floating-point) numbers are represented in the format described in the ANSI/IEEE 754-1985 standard (single-precision). Refer to that standard for more information.

Natural logarithm (LN) and natural exponential (EXP) instructions

LAD / FBD	STL	Description
LN EN ENO IN OUT	**LN IN, OUT**	The Natural Logarithm instruction (LN) performs the natural logarithm of the value in IN and places the result in OUT. The Natural Exponential instruction (EXP) performs the exponential operation of e raised to the power of the value in IN and places the result in OUT. • LN (IN) = OUT • EXP (IN)= OUT To obtain the base 10 logarithm from the natural logarithm: Divide the natural logarithm by 2.302585 (approximately the natural logarithm of 10). To raise any real number to the power of another real number, including fractional exponents: Combine the Natural Exponential instruction with the Natural Logarithm instruction. For example, to raise X to the Y power, use EXP (Y * LN (X)).
EXP EN ENO IN OUT	**EXP IN, OUT**	

Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address • SM1.1 Overflow	• SM1.0 Result of operation = zero • SM1.1 Overflow, illegal value generated during the operation, or illegal input • SM1.2 Negative result

Input / outputs	Data type	Operand
IN	REAL[1]	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
OUT	REAL[1]	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

[1] Real (or floating-point) numbers are represented in the format described in the ANSI/IEEE 754-1985 standard (single-precision). Refer to that standard for more information.

Square root (SQRT) instruction

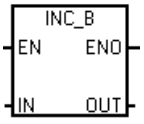
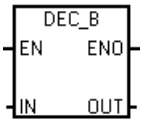
LAD / FBD	STL	Description
SQRT EN ENO IN OUT	**SQRT IN, OUT**	The Square Root instruction (SQRT) takes the square root of a real number (IN) and produces a real number result OUT. • SQRT (IN)= OUT To obtain other roots: • 5 cubed = 5^3 = EXP(3*LN(5)) = 125 • The cube root of 125 = 125^(1/3) = EXP((1/3)*LN(125))= 5 • The square root of 5 cubed = 5^(3/2) = EXP(3/2*LN(5)) = 11.18034

Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address • SM1.1 Overflow	• SM1.0 Result of operation = zero • SM1.1 Overflow, illegal value generated during the operation, or illegal input • SM1.2 Negative result

Input / outputs	Data type	Operand
IN	REAL[1]	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
OUT	REAL[1]	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

[1] Real (or floating-point) numbers are represented in the format described in the ANSI/IEEE 754-1985 standard (single-precision). Refer to that standard for more information.

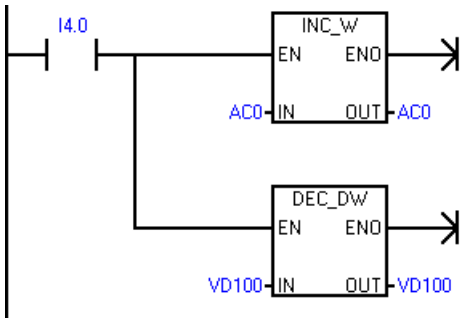
10.8.4 Increment and decrement

LAD / FBD	STL	Description
 INC_W INC_DW	**INCB OUT** **INCW OUT** **INCD OUT**	The increment instructions add 1 to the input value IN and place the result into the location at OUT. • LAD and FBD: IN + 1 = OUT • STL: OUT + 1 = OUT Increment byte (INC_B) operations are unsigned. Increment word (INC_W) operations are signed. Increment double word (INC_DW) operations are signed.
 DEC_W DEC_DW	**DECB OUT** **DECW OUT** **DECD OUT**	The decrement instructions subtract 1 from the input value IN and place the result into the location at OUT. • LAD and FBD: IN - 1 = OUT • STL: OUT - 1= OUT Decrement byte (DEC_B) operations are unsigned. Decrement Word (DEC_W) operations are signed. Decrement double Word (DEC_D) operations are signed.

Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address • SM1.1 Overflow	• SM1.0 Result of operation = zero • SM1.1 Overflow, illegal value generated during the operation, or illegal input • SM1.2 Negative result

Input / output	Data type	Operand
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC,*VD, *LD, *AC
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

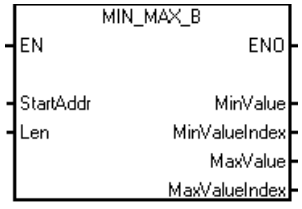
Example: Increment and decrement

LAD	STL
	**Network 1** **LD I4.0** **INCW AC0** **DECD VD100**

Increment/decrement operations from the LAD example

	IN				OUT
Increment word	125	+	1	=	126
Data address	AC0				AC0
Decrement double word	128000	-	1	=	127999
Data address	VD100				VD100

10.8.5 Get the maximum or minimum value

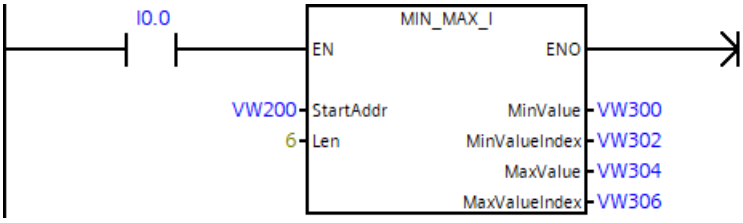
LAD / FBD	STL	Description
 MIN_MAX_I MIN_MAX_DI MIN_MAX_R	**MINMAXB StarAddr, MinValue, MinValueIndex, MaxValue, MaxValueIndex, Len** **MINMAXI StarAddr, MinValue, MinValueIndex, MaxValue, MaxValueIndex, Len** **MINMAXDI StarAddr, MinValue, MinValueIndex, MaxValue, MaxValueIndex, Len** **MINMAXR StarAddr, MinValue, MinValueIndex, MaxValue, MaxValueIndex, Len**	The "Get the maximum or minimum value" instruction compares the values at the available inputs, then writes the value and index of the minimum and maximum value to the output of MinValue, MinValueIndex, MaxValue and MaxValueIndex. If there is more than one minimum or maximum value in the array, the index number of the first minimum or maximum value is output. The index of array starts from 0.

Non-fatal errors with ENO = 0

- 0X0006: Indirect address, Len exceeds the length of the V area.
- 0X0090: invalid parameter, when Len=0.

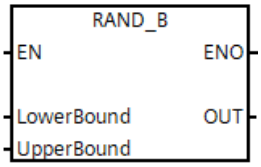
Input / output	Data type	Operand
StartAddr	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC,
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AIW, *VD, *LD, *AC
	DINT	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
	REAL	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
Len	WORD	IW, QW, VW, MW, SMW, SW, LW, AC, Constant, *VD, *LD, *AC
MinValue, MaxValue	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC,*VD, *LD, *AC, AQW
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC
MinValueIndex, MaxValueIndex	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, *VD, *LD, *AC

Example: Get the minimum value of a array

LAD	STL
	**Network 1** **LD I0.0** **MINMAXI VW200, VW300, VW302, VW304, VW306, 6**

Get MinValue, MinValueIndex, MaxValue and MaxValueIndex in source 6 word address sequence (VW200 to VW210) to destination word address sequence (VW300 to VW306).						
Source data values	+98	-522	+100	+0	+6	+10000
Source data addresses	VW200	VW202	VW204	VW206	VW208	VW210
If I0.0 = 1, then execute MIN_MAX_I to get MinValue, MinValueIndex, MaxValue and MaxValueIndex to destination addresses						
Destination data values	-522	1		+10000	5	
Destination data addresses	VW300	VW302		VW304	VW306	

10.8.6 Get a random value

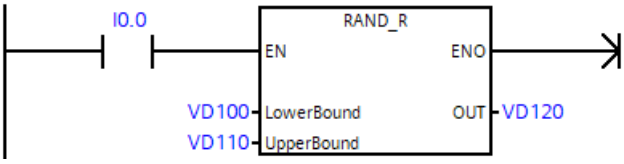
LAD / FBD	STL	Description
 RAND_I RAND_DI RAND_R	**RANDB** **LowerBound** **UpperBound OUT** **RANDI** **LowerBound** **UpperBound OUT** **RANDDI** **LowerBound** **UpperBound OUT** **RANDR** **LowerBound** **UpperBound OUT**	The Random instructions generate a random value in the set range to the OUT output.

Non-fatal errors with ENO = 0

- 0X0090: invalid parameter, when LowerBound > UpperBound.

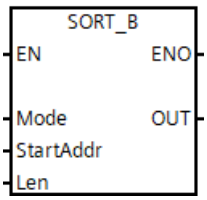
Input / output	Data type	Operand
LowerBound	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD, Constant
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
UpperBound	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD, Constant
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *AC, *LD
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Example: Get a random value

LAD		STL
		**Network 1** **LD I0.0** **RANDR VD100, VD110, VD120**
Get a random number between two closed interval between lowerBound VD100 and Upper Bound VD110.		
Source data values	+10.0	+10000.0

LAD		STL
Source data addresses	VD100	VD110
If I0.0 = 1, then execute RAND_R to get random value from source data values to destination addresses		
Destination data values	+1869.342	
Destination data addresses	VD120	

10.8.7 Sort the array

LAD / FBD	STL	Description
 SORT_I SORT_DI SORT_R	**SORTB Mode, StartAddr, OUT, Len** **SORTI Mode, StartAddr, OUT, Len** **SORTDI Mode, StartAddr, OUT, Len** **SORTR Mode, StartAddr, OUT, Len**	The Sort instruction sorts the given array to the OUT output.

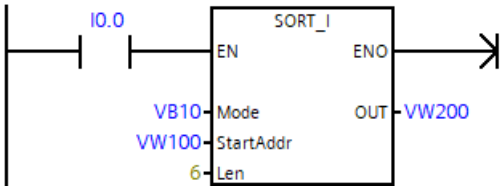
Non-fatal errors with ENO = 0

- 0X0006: Indirect address, Len exceeds the length of the V area.
- 0X0090: invalid parameter, when Len=0 or mode is not 0 or 1.

Input / output	Data type	Operand
Mode [1]	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, Constant
StartAddr	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC,
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AIW, *VD, *LD, *AC
	DINT	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
	REAL	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
Len	WORD	IW, QW, VW, MW, SMW, SW, LW, AC, Constant, *VD, *LD, *AC
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *AC, *LD
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, *VD, *LD, *AC
	DINT	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
	REAL	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC

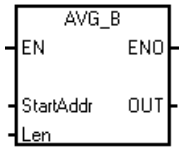
[1] 0: Ascending order; 1: Descending order

Example: Sort the array

LAD	STL
	**Network 1** **LD I0.0** **SORTI VB10, VW100, VW200, 6**

Sort the data in source 6 word address sequence (VW100 to VW110) in ascending order(VB10=0) to destination start as word address (VW200).						
Source data values	+23	+123	+98	+200	+300	-2
Source data addresses	VW100	VW102	VW104	VW106	VW108	VW110
If I0.0 = 1, then execute SORT_INT to sort the source data and write them to destination addresses						
Destination data values	-2	+23	+98	+123	+200	+300
Destination data addresses	VW200	VW202	VW204	VW206	VW208	VW210

10.8.8 Get the average of a array

LAD / FBD	STL	Description
 AVG_I AVG_DI AVG_R	**AVGB StartAddr OUT Len** **AVGI StartAddr OUT Len** **AVGDI StartAddr OUT Len** **AVGR StartAddr OUT Len**	The Average instruction calculates the average value of the given array to the OUT output.

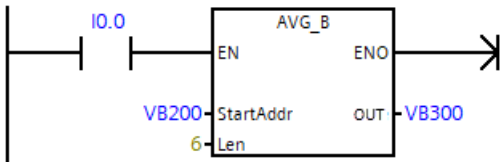
Non-fatal errors with ENO = 0

- 0X0006: Indirect address, Len exceeds the length of the V area.
- 0X0090: invalid parameter, when Len=0.

Input / output	Data type	Operand
StartAddr	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC,
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AIW, *VD, *LD, *AC
	DINT	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
	REAL	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
Len	WORD	IW, QW, VW, MW, SMW, SW, LW, AC, Constant, *VD, *LD, *AC

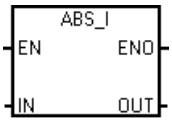
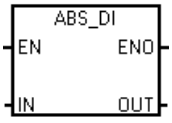
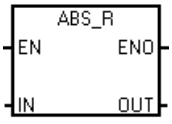
Input / output	Data type	Operand
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Example: Get the average of a array

LAD	STL
	**Network 1** **LD I0.0** **AVGB VB200, VB300, 6**

Get a average value of the set range in source 6 byte address sequence (VB200 to VB205) to destination word address (VB300).						
Source data values	10	20	30	50	60	70
Source data addresses	VB200	VB201	VB202	VB203	VB204	VB205
If I0.0 = 1, execute AVG_B to calculate the average value of the source data to destination addresses						
Destination data values	40					
Destination data addresses	VB300					

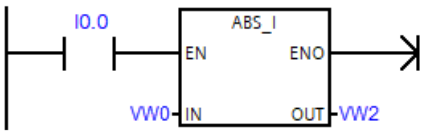
10.8.9 Get absolute values

LAD / FBD	STL	Description
  	**ABSI IN OUT** **ABSDI IN OUT** **ABSR IN OUT**	This instruction get the absolute value of the value specified at input IN and load the result into the memory location OUT. **Note**: This instruction doesn't effect boundary values. For example, for type INT, when the input is -32768, then the output keep -32768.

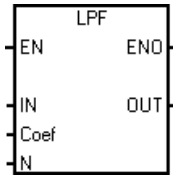
Non-fatal errors with ENO = 0	SM bits affected
• 0X0090: invalid parameter	• SM1.0 Result of operation = zero

Input / output	Data type	Operand
IN	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
OUT	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *AC, *LD
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Example: Absolute instruction

LAD			STL
	Take the absolute value in VW0. Result is put in VW2.		**Network 1** **LD I0.0** **ABSI VW0 VW2**
	IN	-101	
	OUT	101	

10.8.10 Low-pass filter

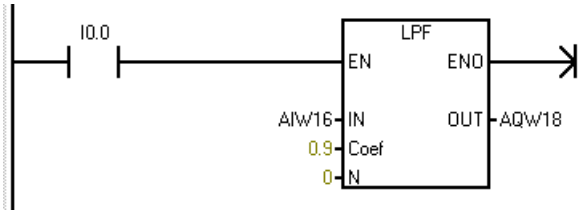
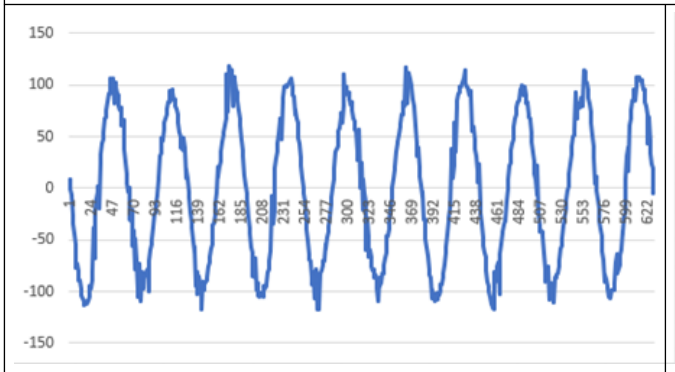
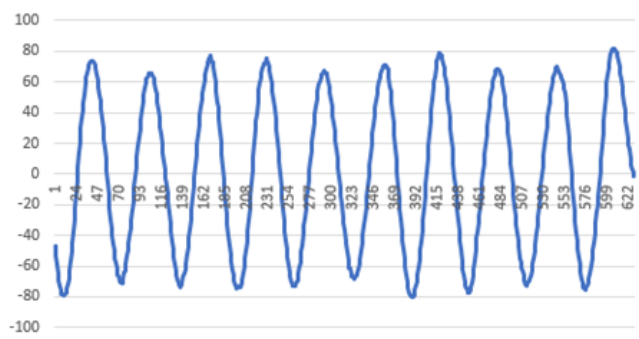
LAD / FBD	STL	Description
	**LPF IN,Coef,N,OUT**	You can use the "Low-pass filter" instruction to attenuate high frequency components, such as noise in a signal. The result of the instruction is calculated using the following equation: y(n) = Coef*y(n-1) + （1- Coef）* x(n); y(n) is the current filter output, Coef is the filter coefficient, y(n-1) is the previous filter output, and x(n) is the current sample value. You can specify the following filter parameters: • Coef: the filter coefficient If the filter coefficient, which is the ratio parameter in the command, is set to 0, no filtering is performed. • N: currently S7-200 SMART supports 64 first-order low-pass filters, so you must specify which filter uses when programming. The same numbered low-pass filter will use the historical filter value to calculate and the current sample value together to get the current filter value.

Non-fatal errors with ENO = 0

- 0X0090: invalid parameter, when the value of N and Coef is out of range.

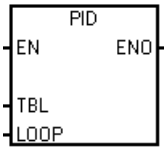
Input / output	Data type	Operand	Remark
IN	WORD	AIW/VW	
Coef	REAL	ID,QD,VD,MD,SMD,SD,LD,AC,*VD,*LD,*AC, Constant	Coef must be in range [0, 1)
N	BYTE	VB, Constant	N must be in range [0, 63]
OUT	WORD	AQW/VW	

Example: Low pass filter

LAD	STL
Select filter 0 to process the input of AIW16 with a filter coefficient of 0.9. The result is put in AQW18.	
	LD I0.0 LPF AIW16, 0.9, 0, AQW18
 Before_LPF	 After_LPF

10.9 PID

10.9.1 PID loop instruction

LAD / FBD	STL	Description
	**PID TBL, LOOP**	The PID loop instruction (PID) executes a PID loop calculation on the referenced LOOP based upon the input and configuration information in Table (TBL).

Non-fatal errors with ENO = 0	SM bits affected
• 0013H Illegal PID loop table	• SM1.1 Overflow

Input / output	Data type	Operand
TBL	BYTE	VB
LOOP	BYTE	Constant (0 to 15)

The PID loop instruction (Proportional, Integral, Derivative Loop) is provided to perform the PID calculation. The top of the logic stack (TOS) must be ON (power flow) to enable the PID calculation. The instruction has two operands: a TABLE address which is the starting address of the loop table and a LOOP number which is a constant from 0 to 15.

Sixteen PID instructions can be used in a program. If two or more PID instructions are used with the same loop number (even if they have different table addresses), the PID calculations will interfere with one another and the output will be unpredictable.

The loop table stores nine parameters used for controlling and monitoring the loop operation and includes the current and previous value of the process variable, the setpoint, output, gain, sample time, integral time (reset), derivative time (rate), and the integral sum (bias).

To perform the PID calculation at the desired sample rate, the PID instruction must be executed either from within a timed interrupt routine or from within the main program at a rate controlled by a timer. The sample time must be supplied as an input to the PID instruction via the loop table.

If there is an error, it will be indicated in SMB220-SMB235: PID control Error code (Page 1093).

Auto-Tune capability has been incorporated into the PID instruction. Refer to "PID loops and tuning" (Page 751) for a detailed description of auto-tuning. The "PID Tune control panel" (Page 765) only works with PID loops created by the PID wizard.

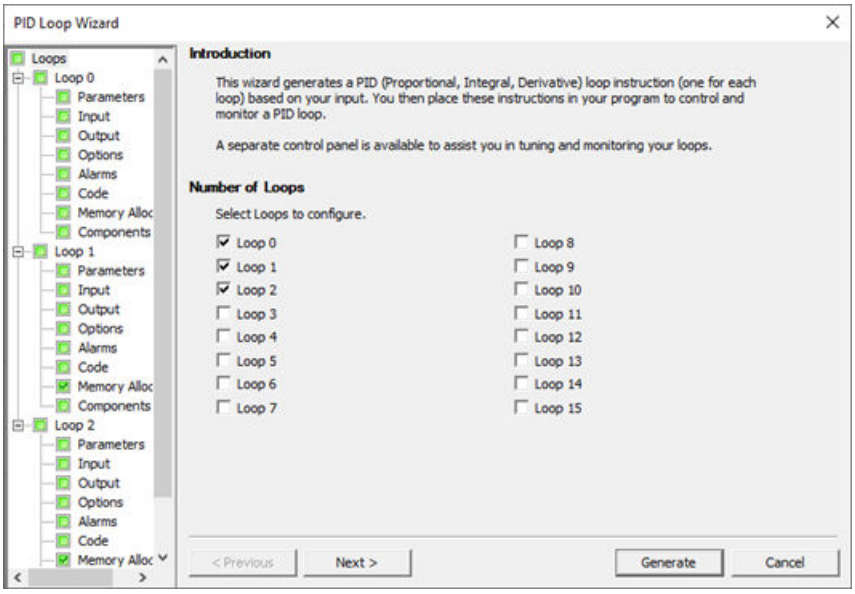
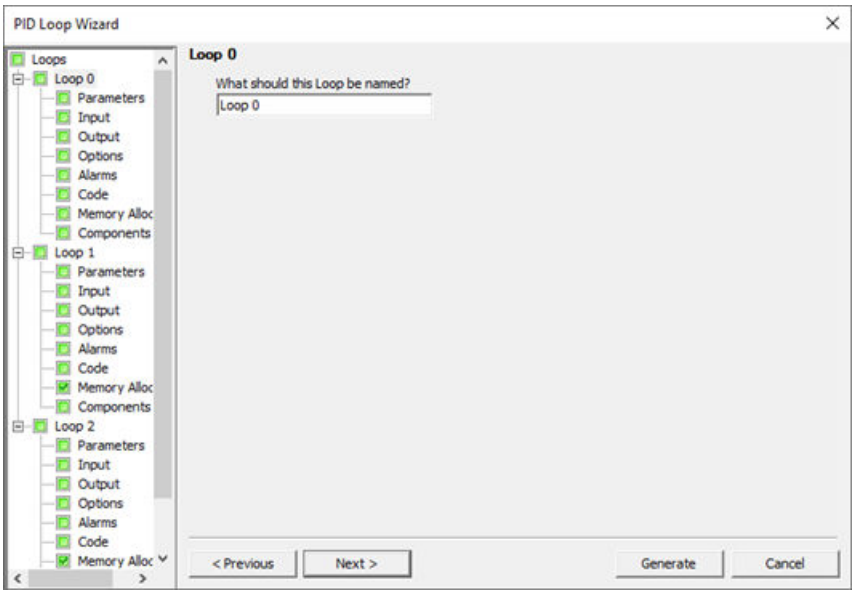
STEP 7-Micro/WIN SMART offers the PID wizard to guide you in defining a PID algorithm for a closed-loop control process. Select the "Instruction wizard" command from the "Tools" menu and then select "PID" from the "Instruction wizard" window.

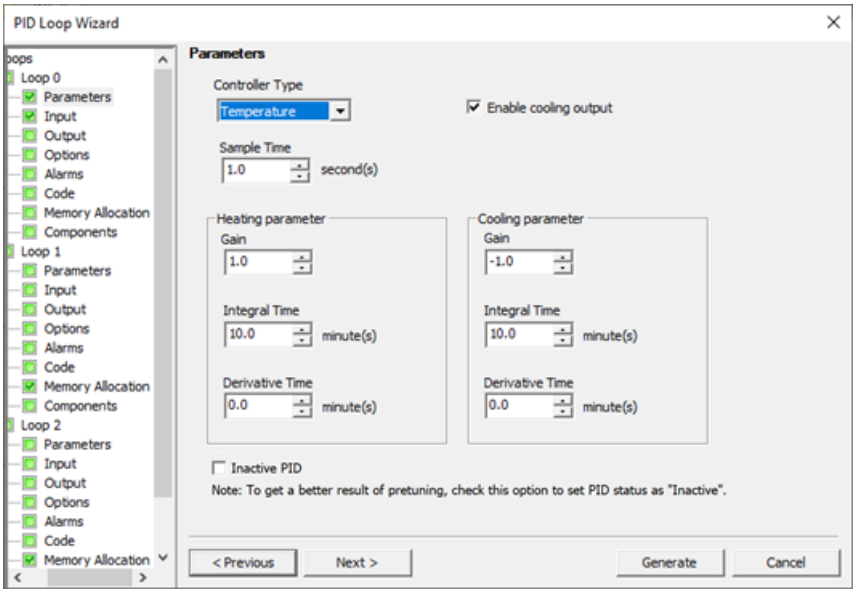
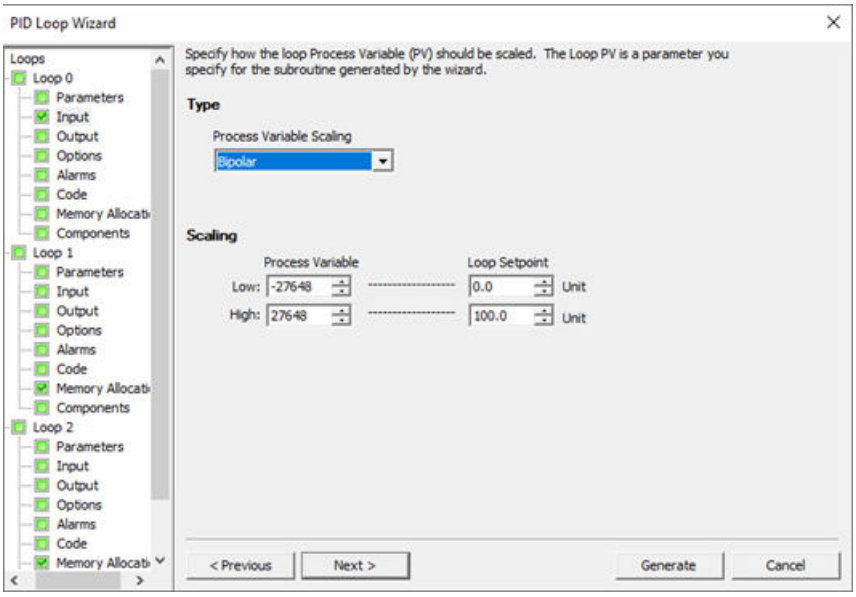
Note

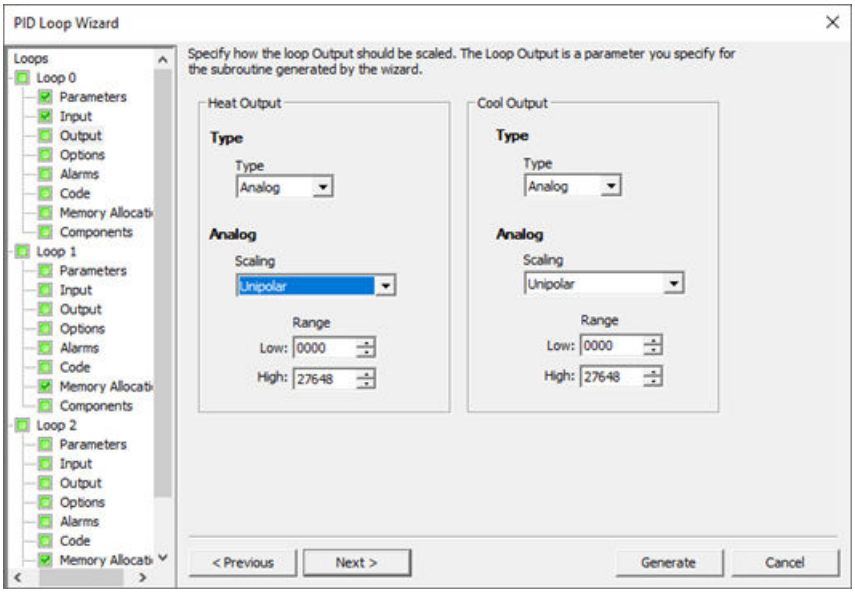
The setpoint of the low range and the setpoint of the high range should correspond to the process variable low range and high range.

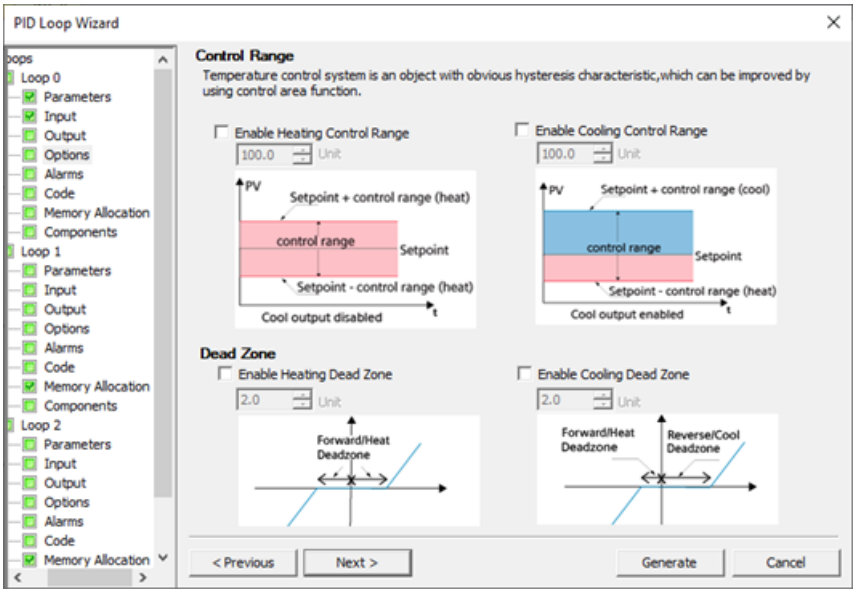
10.9.2 Using the PID wizard

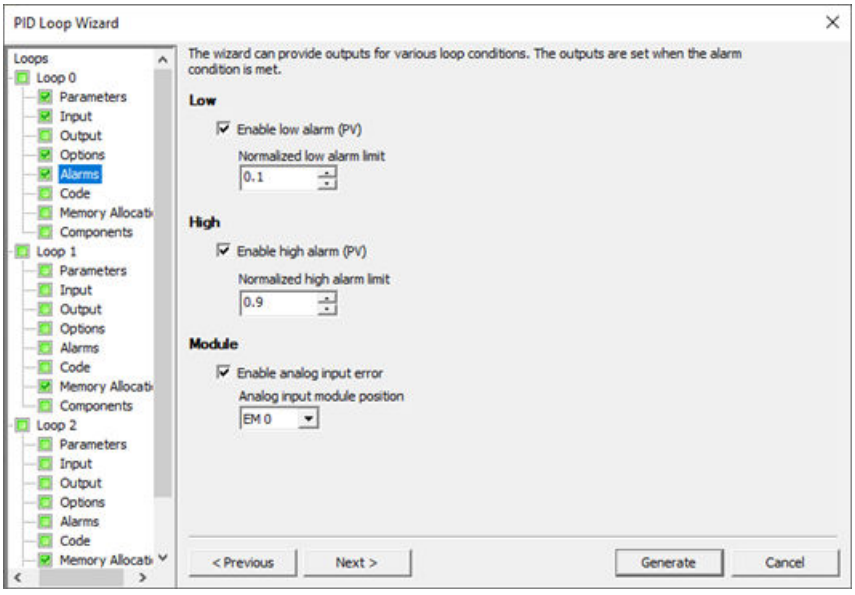
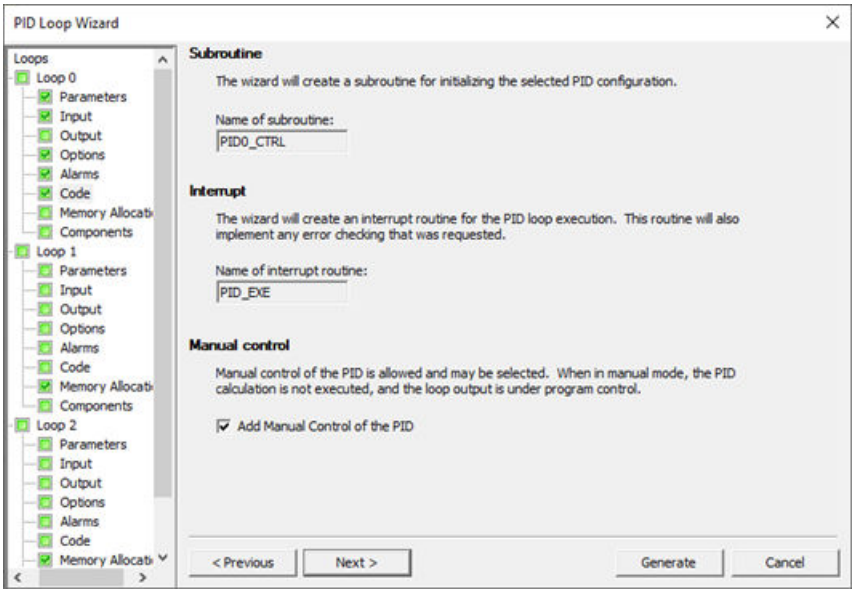
Use the PID wizard to configure your PID loop

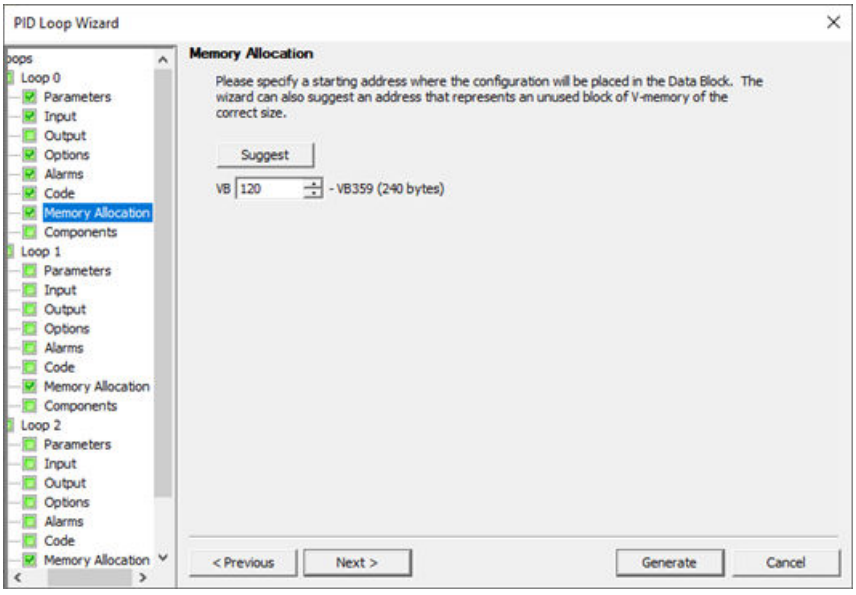
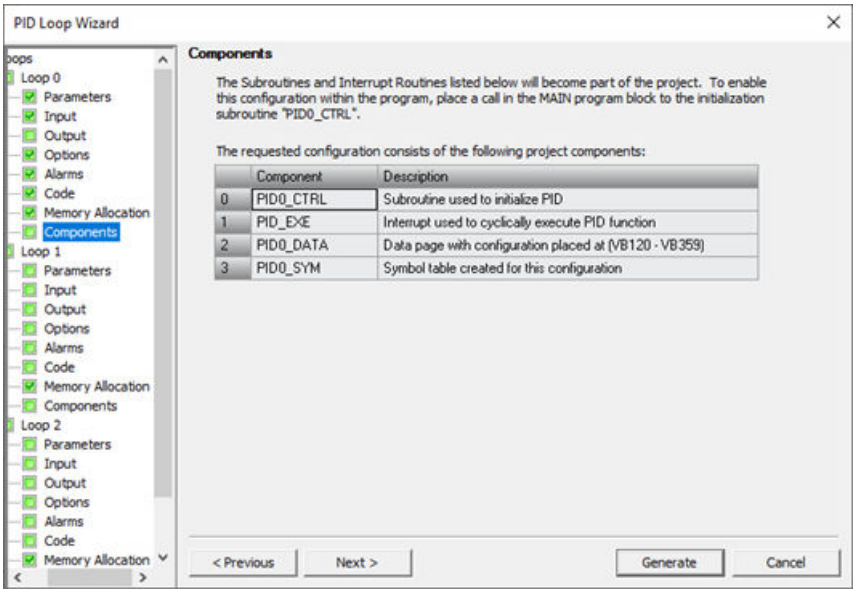
Screen	Description
	In this dialog, you select which loops to configure. You can configure a maximum of 16 loops. When you select a loop on this dialog, the tree view on the left side of the PID wizard updates with all nodes necessary for configuring that loop.
	You can configure a custom name for your loop. The default name of this screen is "Loop x", where "x" is equal to the loop number.

Screen	Description
	Set the following loop parameters: **Bidirectional Output disabled** • Gain (Default value = 1.00) • Sample Time (Default value = 1.00) • Integral Time (Default value = 10.00) • Derivative Time (Default value = 0.00) **Bidirectional Output enabled** Forward/Heating parameter • Gain (default value = 1.00) • Sample Time (default value = 1.00) • Integral Time (default value = 10.00) • Derivative Time (default value = 0.00) Reverse/Cooling parameter • Gain (default value = -1.00) • Sample Time (default value = 1.00) • Integral Time (default value = 10.00) • Derivative Time (default value = 0.00)
	You assign how the loop Process Variable (PV) is to be scaled. You can choose from the following options: • Unipolar (default: 0 to 27648) • Bipolar (default: -27648 to 27648) • Unipolar 20% offset (range: 5530 to 27648) • Temperature x 10 °C • Temperature x 10 °F In the Scaling parameter, you assign how the loop setpoint (SP) is to be scaled. Default is a real number between 0.0 and 100.0.

Screen	Description
	You enter the loop output options: • How the loop output is to be scaled: – Analog – Digital • Analog scaling parameter: – Unipolar (default: 0 to 27648) – Bipolar (default: -27648 to 27648) – Unipolar 20% offset (range: 5530 to 27648) • Analog range parameter: Assign the loop output range. The possible range is -27648 to +27648, depending on your scaling selection.

Screen	Description
	You can set the control range and dead zone for temperature control system. **Control range** When bidirection is disabled: • If Input > Setpoint + control range width, – PID Output is 0 when gain > 0 – PID Output is 1 when gain < 0 • If Input < Setpoint - control range width, – PID Output is 1 when gain > 0 – PID Output is 0 when gain < 0 • If Setpoint - control range width ≤ Input ≤ Setpoint + control range width, then PID Output is calculated according to the PID algorithm. When bidirection is enabled: • If Input > Setpoint + control range width (cooling), then PID Output is Cooling on maximum (-1.0) • If Input < Setpoint - control range width (heating), then PID Output is Heating on maximum (1.0) • If Setpoint - control range width (heating) ≤ Input ≤ Setpoint + control range width (cooling), then PID Output (heating or cooling) is calculated according to the PID algorithm. **Dead zone** • When Bidirection is disabled, the dead zone is located between "Setpoint - dead zone width " and "Setpoint + dead zone width". • When Bidirection is enabled, the dead zone is located between "Setpoint - dead zone width (forward/heating)" and "Setpoint + dead zone width (reverse/cooling)".

Screen	Description
	You can assign what conditions to recognize with alarm inputs. Use the checkboxes to enable the alarms as required: • Low Alarm (PV): Set normalized low alarm limit from 0.0 to high alarm limit; default is 0.10. • High Alarm (PV): Set normalized high alarm limit from low alarm limit to 1.00; default is 0.90. • Analog Input Error: Assign where the input module is attached to the PLC.
	You can make the following code selections: • Subroutine: The PID wizard creates a subroutine for initializing the selected PID configuration. • Interrupt: The PID wizard creates an interrupt routine for the PID loop execution. Note: The wizard assigns a default name for the subroutine and the interrupt routine; you can edit the default names. • Manual control: Use the "Add Manual Control of the PID" checkbox to allow manual control of your PID loops. **Note**: When the PID is in manual mode, the output should be controlled by writing a normalized value (see below) to the Manual Output parameter, instead of changing the output directly. This automatically provides a bumpless transfer when the PID is returned to automatic mode. • Unidirection: 0.00 to 1.00 • Bidirection: -1.00 to 1.00

Screen	Description
	You can assign the starting address of the V memory byte where the configuration is placed in the data block. The wizard can suggest an address that represents an unused block of V memory of the correct size. • Unidirection: 120 bytes • Bidirection: 240 bytes
	This screen shows a list of the subroutines and interrupt routines generated by the PID wizard and gives a brief description of how these should be integrated into your program.

STEP 7-Micro/WIN SMART includes a PID tune control panel (Page 765) that allows you to graphically monitor the behavior of your PID loops. In addition, the control panel allows you to initiate the auto-tune sequence, abort the sequence, and apply the suggested tuning values or your own tuning values.

10.9.3 PID algorithm

In steady state operation, a PID controller regulates the value of the output so as to drive the error (e) to zero. A measure of the error is given by the difference between the setpoint (SP) (the desired operating point) and the process variable (PV) (the actual operating point). The principle of PID control is based upon the following equation that expresses the output, M(t), as a function of a proportional term, an integral term, and a differential term:

Output = Proportional term + Integral term + Differential term

$$\mathbf{M(t) = K_C * e + K_C \int_0^t e\,dt + M_{initial} + K_C * de/dt}$$

where:

M(t)	Loop output as a function of time
$\mathbf{K_C}$	Loop gain
e	Loop error (the difference between setpoint and process variable)
$\mathbf{M_{initial}}$	Initial value of the loop output

In order to implement this control function in a digital computer, the continuous function must be quantized into periodic samples of the error value with subsequent calculation of the output. The corresponding equation that is the basis for the digital computer solution is:

Output = Proportional term + Integral term + Differential term

$$\mathbf{M_n = K_c * e_n + K_I * \sum_1^n e_x + M_{initial} + K_D * (e_n - e_{n-1})}$$

where:

$\mathbf{M_n}$	Calculated value of the loop output at sample time n
$\mathbf{K_c}$	Loop gain
$\mathbf{e_n}$	Value of the loop error at sample time n
$\mathbf{e_{n-1}}$	Previous value of the loop error (at sample time n - 1)
$\mathbf{K_I}$	Proportional constant of the integral term
$\mathbf{M_{initial}}$	Initial value of the loop output
$\mathbf{K_D}$	Proportional constant of the differential term

From this equation, the integral term is shown to be a function of all the error terms from the first sample to the current sample. The differential term is a function of the current sample and the previous sample, while the proportional term is only a function of the current sample. In a digital computer, it is not practical to store all samples of the error term, nor is it necessary.

Since the digital computer must calculate the output value each time the error is sampled beginning with the first sample, it is only necessary to store the previous value of the error and the previous value of the integral term. As a result of the repetitive nature of the digital computer solution, a simplification in the equation that must be solved at any sample time can be made. The simplified equation is:

Output = Proportional term + Integral term + Differential term

$$\mathbf{M_n = K_C * e_n + K_I * e_n + MX + K_D * (e_n - e_{n-1})}$$

where:

M_n	Calculated value of the loop output at sample time n
K_c	Loop gain
e_n	Value of the loop error at sample time n
e_{n-1}	Previous value of the loop error (at sample time n - 1)
K_I	Proportional constant of the integral term
MX	Previous value of the integral term (at sample time n - 1)
K_D	Proportional constant of the differential term

The CPU uses a modified form of the above simplified equation when calculating the loop output value. This modified equation is:

Output = Proportional term + Integral term + Differential term

$$M_n = MP_n + MI_n + MD_n$$

where:

M_n	Calculated value of the loop output at sample time n
MP_n	Value of the proportional term of the loop output at sample time n
MI_n	Value of the integral term of the loop output at sample time n
MD_n	Value of the differential term of the loop output at sample time n

Understanding the elements of the PID equation

Proportional term of the PID equation: The proportional term MP is the product of the gain (K_C), which controls the sensitivity of the output calculation, and the error (e), which is the difference between the setpoint (SP) and the process variable (PV) at a given sample time. The equation for the proportional term as solved by the CPU is:

$$MP_n = K_C * (SP_n - PV_n)$$

where:

MP_n	Value of the proportional term of the loop output at sample time n
K_C	Loop gain
SP_n	Value of the setpoint at sample time n
PV_n	Value of the process variable at sample time n

Integral term of the PID equation: The integral term MI is proportional to the sum of the error (e) over time. The equation for the integral term as solved by the CPU is:

$$MI_n = K_1 e_n + MX = K_C * (T_S / T_I) * (SP_n - PV_n) + MX$$

where:

MI_n	Value of the integral term of the loop output at sample time n
K_C	Loop gain
T_S	Loop sample time
T_I	Integral time (also called the integral time or reset)
SP_n	Value of the setpoint at sample time n

PV_n	Value of the process variable at sample time n
MX	Value of the integral term at sample time n-1 (also called the integral sum or the bias)

The integral sum or bias (MX) is the running sum of all previous values of the integral term. After each calculation of MI_n, the bias is updated with the value of MI_n which might be adjusted or clamped (see the section "Variables and Ranges" for details). The initial value of the bias is typically set to the output value ($M_{initial}$) just prior to the first loop output calculation. Several constants are also part of the integral term, the gain (K_C), the sample time (T_S), which is the cycle time at which the PID loop recalculates the output value, and the integral time or reset (T_I), which is a time used to control the influence of the integral term in the output calculation.

Differential term of the PID equation: The differential term MD is proportional to the change in the error. The CPU uses the following equation for the differential term:

$$MD_n = K_C * (T_D / T_S) * ((SP_n - PV_n) - (SP_{n-1} - PV_{n-1}))$$

To avoid step changes or bumps in the output due to derivative action on setpoint changes, this equation is modified to assume that the setpoint is a constant ($SP_n = SP_{n-1}$). This results in the calculation of the change in the process variable instead of the change in the error as shown:

$$MD_n = K_C * (T_D / T_S) * ((SP_n - PV_n) - (SP_{n-1} - PV_{n-1}))$$

or just:

$$MD_n = K_C * (T_D / T_S) * (PV_{n-1} - PV_n)$$

where:

MD_n	Value of the differential term of the loop output at sample time n
K_C	Loop gain
T_S	Loop sample time
T_D	Differentiation period of the loop (also called the derivative time or rate)
SP_n	Value of the setpoint at sample time n
SP_{n-1}	Value of the setpoint at sample time n - 1
PV_n	Value of the process variable at sample time n - 1
PV_{n-1}	Value of the process variable at sample time n - 1

The process variable rather than the error must be saved for use in the next calculation of the differential term. At the time of the first sample, the value of PV_{n-1} is initialized to be equal to PV_n.

Selecting the type of loop control

In many control systems, it might be necessary to employ only one or two methods of loop control. For example, only proportional control or proportional and integral control might be required. The selection of the type of loop control desired is made by setting the value of the constant parameters.

If you do not want integral action (no "I" in the PID calculation), then a value of infinity "INF", should be specified for the integral time (reset). Even with no integral action, the value of the integral term might not be zero, due to the initial value of the integral sum MX.

If you do not want derivative action (no "D" in the PID calculation), then a value of 0.0 should be assigned for the derivative time (rate).

If you do not want proportional action (no "P" in the PID calculation) and you want I or ID control, then a value of 0.0 should be specified for the gain. Since the loop gain is a factor in the equations for calculating the integral and differential terms, setting a value of 0.0 for the loop gain will result in a value of 1.0 being used for the loop gain in the calculation of the integral and differential terms.

10.9.4 Converting and normalizing the loop inputs

A loop has two input variables, the setpoint and the process variable. The setpoint is generally a fixed value such as the speed setting on the cruise control in your automobile. The process variable is a value that is related to loop output and therefore measures the effect that the loop output has on the controlled system. In the example of the cruise control, the process variable would be a tachometer input that measures the rotational speed of the tires.

Both the setpoint and the process variable are real world values whose magnitude, range, and engineering units could be different. Before these real world values can be operated upon by the PID instruction, the values must be converted to normalized, floating-point representations.

The first step is to convert the real world value from a 16-bit integer value to a floating-point or real number value. The following instruction sequence is provided to show how to convert from an integer value to a real number.

```
ITD    AIW0, AC0    //Convert an input value to a double word
DTR    AC0, AC0     //Convert the 32-bit integer to a real number
```

The next step is to convert the real number value representation of the real world value to a normalized value between 0.0 and 1.0. The following equation is used to normalize either the setpoint or process variable value:

$$\mathbf{R_{Norm} = ((R_{Raw} / Span) + Offset)}$$

where:

$\mathbf{R_{Norm}}$	is the normalized, real number value representation of the real world value
$\mathbf{R_{Raw}}$	is the un-normalized or raw, real number value representation of the real world value
Offset	is 0.0 for unipolar values is 0.5 for bipolar values
Span	is the maximum possible value minus the minimum possible value: = 27,648 for unipolar values (typical) = 55,296 for bipolar values (typical)

The following instruction sequence shows how to normalize the bipolar value in AC0 (whose span is 55,296) as a continuation of the previous instruction sequence:

```
/R      55296.0, AC0  //Normalize the value in the accumulator
+R      0.5, AC0      //Offset the value to the range from 0.0 to 1.0
MOVR    AC0, VD100    //Store the normalized value in the loop TABLE
```

10.9.5 Converting the loop output to a scaled integer value

The loop output is the control variable, such as the throttle setting of the cruise control on an automobile. The loop output is a normalized, real number value between 0.0 and 1.0. Before the loop output can be used to drive an analog output, the loop output must be converted to a 16-bit, scaled integer value. This process is the reverse of converting the PV and SP to a normalized value. The first step is to convert the loop output to a scaled, real number value using the formula given below:

$$R_{Scal} = (M_n - Offset) * Span$$

R_{Scal}	is the scaled, real number value of the loop output
M_n	is the normalized, real number value of the loop output
Offset	is 0.0 for unipolar values is 0.5 for bipolar values
Span	is the maximum possible value minus the minimum possible value = 27,648 for unipolar values (typical) = 55,296 for bipolar values (typical)

The following instruction sequence shows how to scale the loop output:

```
MOVR    VD108, AC0    //Moves the loop output to the accumulator
-R      0.5, AC0      //Include this statement only if the value is bipolar
*R      55296.0, AC0  //Scales the value in the accumulator
```

Next, the scaled, real number value representing the loop output must be converted to a 16-bit integer. The following instruction sequence shows how to do this conversion:

```
ROUND   AC0, AC0      //Converts the real number to a 32-bit integer
DTI     AC0, LW0      //Converts the value to a 16-bit integer
MOVW    LW0, AQW0     //Writes the value to the analog output
```

10.9.6 Forward- or reverse-acting loops

- **Unidirection**
  The loop is forward-acting if the gain is positive and reverse-acting if the gain is negative. (For I or ID control, where the gain value is 0.0, specifying positive values for integral and derivative time will result in a forward-acting loop, and specifying negative values will result in a reverse-acting loop.)
- **Bidirection**
  If bidirectional output is enabled, forward parameters allow only positive gain, and reverse parameters allow only negative gain.

Variables and ranges

The process variable and setpoint are inputs to the PID calculation. Therefore the loop table fields for these variables are read but not altered by the PID instruction.

The output value is generated by the PID calculation, so the output value field in the loop table is updated at the completion of each PID calculation. The output value is clamped between 0.0 and 1.0. The output value field can be used as an input by the user to specify an initial output value when making the transition from manual control to PID instruction (auto) control of the output. (See the discussion in the "Modes" section below.)

If integral control is being used, then the bias value is updated by the PID calculation and the updated value is used as an input in the next PID calculation. When the calculated output value goes out of range (output would be less than 0.0 or greater than 1.0), the bias is adjusted according to the following formulas:

- when the calculated output $M_n > 1.0$

$$MX = 1.0 - (MP_n + MD_n)$$

- when the calculated output $M_n < 0$

$$MX = -(MP_n + MD_n)$$

MX	is the value of the adjusted bias
MP_n	is the value of the proportional term of the loop output at sample time n
MD_n	is the value of the differential term of the loop output at sample time n
M_n	is the value of the loop output at sample time n

By adjusting the bias as described, an improvement in system responsiveness is achieved once the calculated output comes back into the proper range. The calculated bias is also clamped between 0.0 and 1.0 and then is written to the bias field of the loop table at the completion of each PID calculation. The value stored in the loop table is used in the next PID calculation.

The bias value in the loop table can be modified by the user prior to execution of the PID instruction in order to address bias value problems in certain application situations. Care must be taken when manually adjusting the bias, and any bias value written into the loop table must be a real number between 0.0 and 1.0.

A comparison value of the process variable is maintained in the loop table for use in the derivative action part of the PID calculation. You should not modify this value.

Modes

There is no built-in mode control for PID loops. The PID calculation is performed only when power flows to the PID box. Therefore, "automatic" or "auto" mode exists when the PID calculation is performed cyclically. "Manual" mode exists when the PID calculation is not performed.

The PID instruction has a power-flow history bit, similar to a counter instruction. The instruction uses this history bit to detect a 0-to-1 power-flow transition. When the power-flow transition is detected, it will cause the instruction to perform a series of actions to provide a bumpless change from manual control to auto control. In order for change to auto mode control to be bumpless, the value of the output as set by the manual control must be supplied as an input to the PID instruction (written to the loop table entry for M_n) before switching to auto control. The PID instruction performs the following actions to values in the loop table to ensure a bumpless change from manual to auto control when a 0-to-1 power-flow transition is detected:

- Sets setpoint (SP_n) = process variable (PV_n)
- Sets old process variable (PV_{n-1}) = process variable (PV_n)
- Sets bias (MX) = output value (M_n)

The default state of the PID history bits is "set" and that state is established at startup and on every STOP-to-RUN mode transition of the controller. If power flows to the PID box the first time that it is executed after entering RUN mode, then no power-flow transition is detected and the bumpless mode change actions are not performed.

Alarm checking and special operations

The PID instruction is a simple but powerful instruction that performs the PID calculation. If other processing is required such as alarm checking or special calculations on loop variables, these must be implemented using the basic instructions supported by the CPU.

Error conditions

When it is time to compile, the CPU will generate a compile error (range error) and the compilation will fail if the loop table start address or PID loop number operands specified in the instruction are out of range.

Certain loop table input values are not range checked by the PID instruction. You must take care to ensure that the process variable and setpoint (as well as the bias and previous process variable if used as inputs) are real numbers between 0.0 and 1.0.

If any error is encountered while performing the mathematical operations of the PID calculation, then SM1.1 (overflow or illegal value) is set and execution of the PID instruction is terminated. (Update of the output values in the loop table could be incomplete, so you should disregard these values and correct the input value causing the mathematical error before the next execution of the loop's PID instruction.)

Loop table

The loop table is 80 bytes long and has the format shown in the following table.

Offset	Field	Format	Type	Description
0	Process variable (PV_n)	REAL	In	Contains the process variable, which must be scaled between 0.0 and 1.0.
4	Setpoint (SP_n)	REAL	In	Contains the setpoint, which must be scaled between 0.0 and 1.0.
8	Output (M_n)	REAL	In/Out	Contains the calculated output, scaled between 0.0 and 1.0.
12	Gain (K_C)	REAL	In	Contains the gain, which is a proportional constant. Can be a positive or negative number.
16	Sample time (T_S)	REAL	In	Contains the sample time, in seconds. Must be a positive number.
20	Integral time or reset (T_I)	REAL	In	Contains the integral time or reset, in minutes. Must be a positive number.
24	Derivative time or rate (T_D)	REAL	In	Contains the derivative time or rate, in minutes. Must be a positive number.
28	Bias (MX)	REAL	In/Out	Contains the bias or integral sum value between 0.0 and 1.0.
32	Previous process variable (PV_{n-1})	REAL	In/Out	Contains the value of the process variable stored from the last execution of the PID instruction.
36 to 79	Reserved for auto-tuning variables. Refer to PID loop definition table (Page 755) for details.			

10.9.7 Integral anti-windup

When the controller cannot eliminate the deviation between the process value and the setpoint value, the integral will increase until it reaches the limit. At this time, if the deviation between the process value and the setpoint value turns nagative, the integral will affect the output of the controller, which will result in the decreased dynamic performance, the control lag and oscillation may occur.
STEP 7-MicroWIN SMART with firmware V2.8 supports configuring an integral upper limit to limit the intergal and achieve the effect of integral aunti-windup. For detailed information, refer to the content of PID loop definition table (Page 755).

10.10 Interrupt

10.10.1 Interrupt instructions

When you make the transition to RUN mode, interrupts are initially disabled. In RUN mode, you can enable interrupt processing by executing the ENI (enable Interrupt) instruction. Executing the DISI (disable interrupt) instruction inhibits the processing of interrupts; however, active interrupt events will continue to be queued.

ENI, DISI, and CRETI

LAD	FBD	STL	Description
ENI	ENI	**ENI**	The enable interrupt instruction globally enables processing of all attached interrupt events.
DISI	DISI	**DISI**	The disable interrupt instruction globally disables processing of all interrupt events.
RETI	RETI	**CRETI**	The conditional return from interrupt instruction can be used to return from an interrupt, based upon the condition of the preceding program logic.

Non-fatal errors with ENO = 0	SM bits affected
• 0004H Attempted ENI or DISI execution disallowed in an interrupt routine	None

ATCH, DTCH, and CEVENT

LAD / FBD	STL	Description
ATCH EN ENO INT EVNT	**ATCH INT, EVNT**	The attach interrupt instruction associates an interrupt event EVNT with an interrupt routine number INT and enables the interrupt event.
DTCH EN ENO EVNT	**DTCH EVNT**	The detach interrupt instruction disassociates an interrupt event EVNT from all interrupt routines and disables the interrupt event.
CLR_EVNT EN ENO EVNT	**CEVENT EVNT**	The clear interrupt event instruction removes all interrupt events of type EVNT from the interrupt queue. Use this instruction to clear the interrupt queue of unwanted interrupt events. If this instruction is being used to clear out spurious interrupt events, then you should detach the event before clearing the events from the queue. Otherwise new events will be added to the queue after the clear event instruction has been executed.

Non-fatal errors with ENO = 0	SM bits affected
• 0002H Conflicting assignment of inputs to an HSC • 0016H Attempt to use HSC or edge interrupt on input channel that is already allocated to a motion control function • 0019H Attempt to use a signal board function without a signal board installed and configured • 0090H Invalid operand (illegal event number)	None

Input / output	Data type	Operand
INT	BYTE	Constant: interrupt routine number (0 to 127)
EVNT	BYTE	Constant: interrupt event number

10.10.2 Interrupt routine overview and CPU model event support

Before you can invoke an interrupt routine, you must assign an association between the interrupt event and the program segment that you want to execute when the event occurs. Use the attach interrupt instruction to associate an interrupt event (specified by the interrupt event number) and the program segment (specified by an interrupt routine number). You can attach multiple interrupt events to one interrupt routine, but you cannot attach one event to multiple interrupt routines.

When you attach an event and interrupt routine, new occurrences of this event cause execution of the attached interrupt routine only if the program executed the global ENI (enable interrupts) instruction and interrupt event processing is active. Otherwise, the CPU adds the event to the interrupt event queue. If you disable all interrupts using the global DISI (disable interrupts) instruction, the CPU queues each occurrence of the interrupt event until the interrupts are re-enabled, using the global ENI (enable interrupt) instruction or the interrupt queue overflows.

You can disable individual interrupt events by breaking the association between the interrupt event and the interrupt routine with the detach Interrupt instruction. The detach interrupt instruction returns the interrupt to an inactive or ignored state. The following table lists the different types of interrupt events.

Event	Description	CPU SR40/ST40 CPU SR60/ST60
0	I0.0 Rising edge	Y
1	I0.0 Falling edge	Y
2	I0.1 Rising edge	Y
3	I0.1 Falling edge	Y
4	I0.2 Rising edge	Y

Event	Description	CPU SR40/ST40 CPU SR60/ST60
5	I0.2 Falling edge	Y
6	I0.3 Rising edge	Y
7	I0.3 Falling edge	Y
8	Port 0 Receive character	Y
9	Port 0 Transmit complete	Y
10	Timed interrupt 0 (SMB34 controls the time interval)	Y
11	Timed interrupt 1 (SMB35 controls the time interval)	Y
12	HSC0 CV=PV (current value = preset value)	Y
13	HSC1 CV=PV (current value = preset value)	Y
14	HSC1 Direction changed	Y
15	HSC1 External reset	Y
16	HSC2 CV=PV (current value = preset value)	Y
17	HSC2 Direction changed	Y
18	HSC2 External reset	Y
19	PTO0 Pulse count complete	Y
20	PTO1 Pulse count complete	Y
21	Timer T32 CT=PT (current time = preset time)	Y
22	Timer T96 CT=PT (current time = preset time)	Y
23	Port 0 Receive message complete	Y
24	Port 1 Receive message complete	Y
25	Port 1 Receive character	Y
26	Port 1 Transmit complete	Y
27	HSC0 Direction changed	Y
28	HSC0 External reset	Y
29	HSC4 CV=PV	Y
30	HSC4 direction changed	Y
31	HSC4 external reset	Y
32	HSC3 CV=PV (current value = preset value)	Y
33	HSC5 CV=PV	Y
34	PTO2 Pulse count complete	Y
35	I7.0 Rising edge (signal board slot 0, input 0)	Y
36	I7.0 Falling edge (signal board slot 0, input 0)	Y
37	I7.1 Rising edge (signal board slot 0, input 1)	Y
38	I7.1 Falling edge (signal board slot 0, input 1)	Y
43	HSC5 direction changed	Y
44	HSC5 external reset	Y
45	PTO3 Pulse count complete	Y
46	Port 2 Receive message complete	Y
47	Port 2 Receive character	Y
48	Port 2 Transmit complete	Y
49	I72.0 Rising edge (signal board slot 1, input 0)	Y
50	I72.0 Falling edge (signal board slot 1, input 0)	Y

Event	Description	CPU SR40/ST40 CPU SR60/ST60
51	I72.1 Rising edge (signal board slot 1, input 1)	Y
52	I72.1 Falling edge (signal board slot 1, input 1)	Y
53	HSC3 Direction changed	Y
54	HSC3 External reset	Y
55	HSC6 CV=PV	Y
56	HSC6 Direction changed	Y
57	HSC6 External reset	Y
58	HSC7 CV=PV	Y
59	HSC7 Direction changed	Y
60	HSC7 External reset	Y
61	Timer T262 CT=PT (current time = preset time)	Y
62	Timer T263 CT=PT (current time = preset time)	Y
63	Timer T264 CT=PT (current time = preset time)	Y
64	Timer T265 CT=PT (current time = preset time)	Y
65	Timer T266 CT=PT (current time = preset time)	Y
66	Timer T267 CT=PT (current time = preset time)	Y
67	PTO4 Pulse count complete	Y

10.10.3 Interrupt programming guidelines

Interrupt routine execution

An interrupt routine executes in response to an associated internal or external event. Once the last instruction of an interrupt routine has executed, control returns to the point in the scan cycle at the point of the interruption. You can exit the routine by executing a conditional return from interrupt instruction (CRETI).

Interrupt processing provides quick reaction to special internal or external events. Optimize your interrupt routines to perform a specific task, and then return control to the scan cycle.

Note

- You cannot use the disable interrupt (DISI), enable interrupt (ENI), high-speed counter definition (HDEF), and end (END) instructions in an interrupt routine.
- Keep interrupt routine program logic short and to the point, so execution is quick and other processes are not deferred for long periods of time. If this is not done, unexpected conditions can cause abnormal operation of equipment controlled by the main program.

System support for interrupts

Because interrupts can affect contact, coil, and accumulator logic, the system saves and reloads the logic stack, accumulator registers, and the special memory bits (SM) that indicate the status of accumulator and instruction operations. This avoids disruption to the main user program caused by branching to and from an interrupt routine.

Calling subroutines from interrupt routines

You can call four nesting levels of subroutines from an interrupt routine. The accumulators and the logic stack are shared between an interrupt routine and the four nesting levels of subroutines called from the interrupt routine

Sharing data between the main program and the interrupt routines

You can share data between the main program and one or more interrupt routines. Because it is not possible to predict when the CPU might generate an interrupt, it is desirable to limit the number of variables that are used by both the interrupt routine and elsewhere in the program. Problems with the consistency of shared data can result due to the actions of interrupt routines when the execution of instructions in your main program is interrupted by interrupt events. Use the interrupt block "variable table" (block call interface table) to ensure that your interrupt routine uses only the temporary memory and does not overwrite data used somewhere else in your program.

Ensuring access for a single shared variable

- For an STL program that is sharing a single variable: If the shared data is a single byte, word, or double word variable and your program is written in STL, then correct shared access can be ensured by storing the intermediate values from operations on shared data only in non-shared memory locations or accumulators.
- For a LAD program that is sharing a single variable: If the shared data is a single byte, word, or double word variable and your program is written in LAD, then correct shared access can be ensured by establishing the convention that access to shared memory locations be made using only Move instructions (MOVB, MOVW, MOVD, MOVR). While many LAD instructions are composed of interruptible sequences of STL instructions, these Move instructions are composed of a single STL instruction whose execution cannot be affected by interrupt events.

Ensuring access for multiple shared variables

For an STL or LAD program that is sharing multiple variables: If the shared data is composed of a number of related bytes, words, or double words, then the interrupt disable/enable instructions (DISI and ENI) can be used to control interrupt routine execution. At the point in your main program where operations on shared memory locations are to begin, disable the interrupts. Once all actions affecting the shared locations are complete, re-enable the interrupts. During the time that interrupts are disabled, interrupt routines cannot be executed and therefore cannot access shared memory locations; however, this approach can result in delayed response to interrupt events.

10.10.4 Types of interrupt events that the S7-200 SMART CPU supports

Communication port interrupts

The serial communications port of the CPU can be controlled by your program. This mode of operating the communications port is called Freeport mode. In Freeport mode, your program defines the baud rate, bits per character, parity, and protocol. The Receive and Transmit interrupts are available to facilitate your program-controlled communications. Refer to the Transmit and Receive instructions for more information.

I/O interrupts

I/O interrupts include rising/falling edge interrupts, high-speed counter interrupts, and pulse train output interrupts. The CPU can generate an interrupt on rising and/or falling edges of an input for input channels I0.0, I0.1, I0.2, and I0.3 (and for I7.0, I7.1, I72.0 and I72.1 for standard CPUs with an optional digital input signal board). The rising edge and the falling edge events can be captured for each of these input points. These rising/falling edge events can be used to signify a condition that must receive immediate attention when the event happens.

The high-speed counter interrupts allow you to respond to conditions such as the current value reaching the preset value, a change in counting direction that might correspond to a reversal in the direction in which a shaft is turning, or an external reset of the counter. Each of these high-speed counter events allows action to be taken in real time in response to high-speed events that cannot be controlled at programmable logic controller scan speeds.

The pulse train output interrupts provide immediate notification of completion of the output of the prescribed number of pulses. A typical use of pulse train outputs is stepper motor control.

You enable each of the above interrupts by attaching an interrupt routine to the related I/O event.

Time-based interrupts

Time-based interrupts include timed interrupts, the timer T32/T96 interrupts and the timer T262 - T267 interrupts. You can specify actions to be taken on a cyclic basis using a timed interrupt. The cycle time is set in 1-ms increments from 1 ms to 255 ms. You must write the cycle time in SMB34 for timed interrupt 0, and in SMB35 for timed interrupt 1.

The timed interrupt event transfers control to the appropriate interrupt routine each time the timer expires. Typically, you use timed interrupts to control the sampling of analog inputs or to execute a PID loop at regular intervals.

A timed interrupt is enabled and timing begins when you attach an interrupt routine to a timed interrupt event. During the attachment, the system captures the cycle time value, so subsequent changes to SMB34 and SMB35 do not affect the cycle time. To change the cycle time, you must modify the cycle time value, and then re-attach the interrupt routine to the timed interrupt event. When the re-attachment occurs, the timed interrupt function clears any accumulated time from the previous attachment and begins timing with the new value.

After being enabled, the timed interrupt runs continuously and executes the attached interrupt routine, at the end of each successive time interval. If you exit RUN mode or detach the timed interrupt, the timed interrupt is disabled. If the global DISI (disable interrupt)

instruction is executed, timed interrupts continue to occur, but the attached interrupt routine is not processed yet. Each occurrence of the timed interrupt is queued (until either interrupts are enabled or the queue is full).

The timer T32/T96 interrupts and the timer T262 - T267 interrupts allow timely response to the completion of a specified time interval. These interrupts are only supported for the 1-ms resolution on-delay (TON) and off-delay (TOF) timers T32, T96 and T262 - T267. The T32, T96 and T262 - T267 timers and otherwise behave normally. Once the interrupt is enabled, the attached interrupt routine is executed when the active timer's current value becomes equal to the preset time value during the normal 1-ms timer update performed in the CPU. You enable these interrupts by attaching an interrupt routine to the T32 (event 21), T96 (event 22), T262 - T267 (event 61 - 66) interrupt events.

10.10.5 Interrupt priority, queuing, and example program

Interrupt service

Interrupts are serviced by the CPU on a first-come-first-served basis within their respective priority group. Only one user-interrupt routine is ever being executed at any point in time. Once the execution of an interrupt routine begins, the routine is executed to completion. It cannot be pre-empted by another interrupt routine, even by a higher priority routine. Interrupts that occur while another interrupt is being processed are queued for later processing. The following table shows the three interrupt queues and the maximum number of interrupts they can store.

It is possible that more interrupts can occur than a queue can hold. Therefore, queue overflow memory bits (identifying the type of interrupt events that have been lost) are maintained by the system. The following table shows the interrupt queue overflow bits. You should use these bits only in an interrupt routine because they are reset when the queue is emptied, and control is returned to the scan cycle.

If multiple interrupt events occur at the same time, the priority (group and within a group) determines which interrupt event is processed first. Once the highest priority has been handled, the queue is examined to find the current highest priority event that remains in the queue and the interrupt routine attached to that event is executed. This continues until the queue is empty and control is returned to the scan cycle.

Maximum number of entries per interrupt queue

The following table shows all interrupt events, with their priority and assigned event number.

Queue	Queue depth for all S7-200 SMART CPU models
Communications queue	4
I/O interrupt queue	16
Timed interrupt queue	8

Interrupt queue overflow bits

Description (0 = No Overflow, 1 = Overflow)	SM Bit
Communications queue	SM4.0
I/O Interrupt queue	SM4.1
Timed Interrupt queue	SM4.2

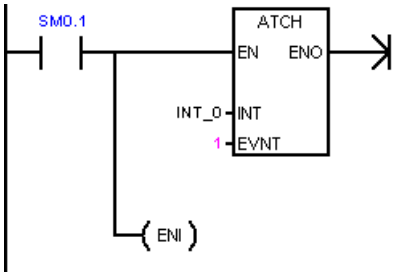
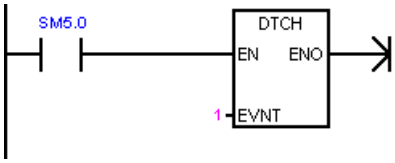
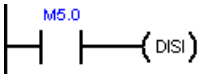
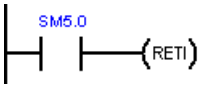
Priority order for interrupt events

Priority group	Event	Description
Communications Highest Priority	8	Port 0 Receive character
	9	Port 0 Transmit complete
	23	Port 0 Receive message complete
	24	Port 1 Receive message complete
	25	Port 1 Receive character
	26	Port 1 Transmit complete
	46	Port 2 Receive message complete
	47	Port 2 Receive character
	48	Port 2 Transmit complete

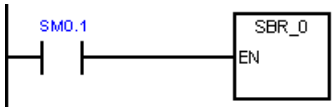
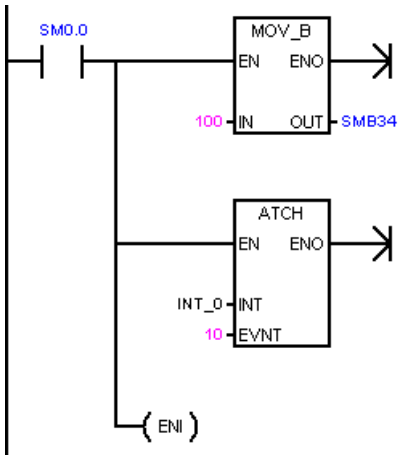
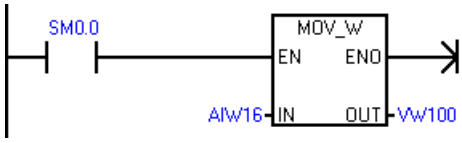
Priority group	Event	Description
Discrete Medium Priority	0	I0.0 Rising edge
	1	I0.0 Falling edge
	2	I0.1 Rising edge
	3	I0.1 Falling edge
	4	I0.2 Rising edge
	5	I0.2 Falling edge
	6	I0.3 Rising edge
	7	I0.3 Falling edge
	12	HSC0 CV=PV (current value = preset value)
	13	HSC1 CV=PV (current value = preset value)
	14	HSC1 direction changed
	15	HSC1 external reset/Z phase
	16	HSC2 CV=PV (current value = preset value)
	17	HSC2 Direction changed
	18	HSC2 External reset
	27	HSC0 Direction changed
	28	HSC0 External reset
	29	HSC4 CV=PV
	30	HSC4 direction changed
	31	HSC4 external reset
	32	HSC3 CV=PV (current value = preset value)
	33	HSC5 CV=PV
	35	I7.0 Rising edge (signal board slot 0, input 0)
	36	I7.0 Falling edge (signal board slot 0, input 0)
	37	I7.1 Rising edge (signal board slot 0, input 1)
	38	I7.1 Falling edge (signal board slot 0, input 1)
	43	HSC5 direction changed
	44	HSC5 external reset
	19	PTO0 Pulse count complete
	20	PTO1 Pulse count complete
	34	PTO2 Pulse count complete
	45	PTO3 Pulse count complete
	67	PTO4 Pulse count complete
	46	Port 2 Receive message complete
	47	Port 2 Receive character
	48	Port 2 Transmit complete
	49	I72.0 Rising edge (signal board slot 1, input 0)
	50	I72.1 Rising edge (signal board slot 1, input 1)
	51	I72.0 Falling edge (signal board slot 1, input 0)
	52	I72.1 Falling edge (signal board slot 1, input 1)
	53	HSC3 Direction changed
	54	HSC3 external reset/Z phase
	55	HSC6 CV=PV

Priority group	Event	Description
	56	HSC6 direction changed
	57	HSC6 external reset
	58	HSC7 CV=PV
	59	HSC7 direction changed
	60	HSC7 external reset
Timed Lowest Priority	10	Timed interrupt 0 SMB34
	11	Timed interrupt 1 SMB35
	21	Timer T32 CT=PT interrupt
	22	Timer T96 CT=PT interrupt
	61	Timer T262 CT=PT interrupt
	62	Timer T263 CT=PT interrupt
	63	Timer T264 CT=PT interrupt
	64	Timer T265 CT=PT interrupt
	65	Timer T266 CT=PT interrupt
	66	Timer T267 CT=PT interrupt

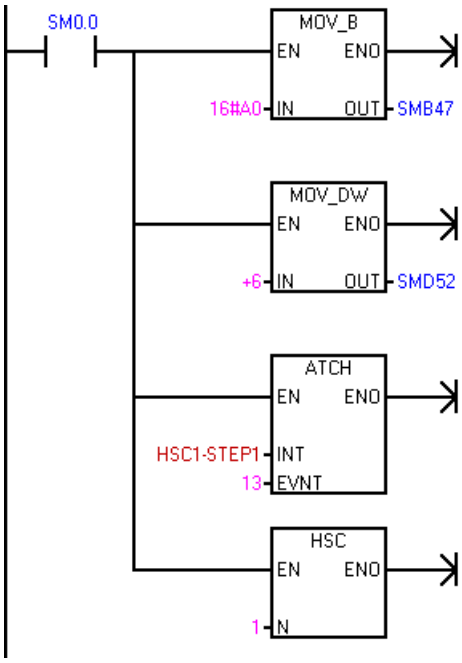
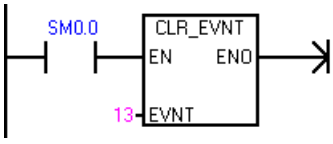
Example 1: Input signal edge detector interrupt

LAD			STL
MAIN Network 1		On the first scan: 1. Define interrupt routine INT_0 to be a falling-edge interrupt for I0.0. 2. Globally enable interrupts.	**Network 1** **LD SM0.1** **ATCH INT_0, 1** **ENI**
Network 2		If an I/O error is detected, disable the falling-edge interrupt for I0.0. (This network is optional.)	**Network 2** **LD SM5.0** **DTCH 1**
Network 3		When M5.0 is on, disable all interrupts. When disabled, attached interrupt events will be queued, but the corresponding interrupt routines will not be executed until interrupts are re-enabled with the ENI instruction.	**Network 3** **LD M5.0** **DISI**
INT 0 Network 1		I0.0 falling-edge interrupt routine: Conditional return based on an I/O error.	**Network 1** **LD SM5.0** **CRETI**

Example 2: Timed interrupt for reading the value of an analog input

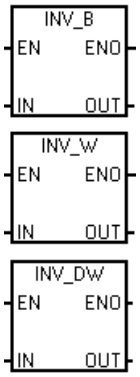
LAD			STL
MAIN Network 1		On the first scan, call subroutine 0.	**Network 1** **LD SM0.1** **CALL SBR_0**
SBR 0 Network 1		Set the interval for the timed interrupt 0 to 100 ms. Attach timed interrupt 0 (Event 10) to INT_0. Global interrupt enable.	**Network 1** **LD SM0.0** **MOVB 100, SMB34** **ATCH INT_0, 10** **ENI**
INT 0 Network 1		Read the value of AIW16 every 100 ms.	**Network 1** **LD SM0.0** **MOVW AIW16, VW100**

Example 3: Clear interrupt event instruction

LAD			STL
SBR 1 Network 1		HSC instruction wizard: Set control bits, write preset. PV = 6 Attach interrupt HSC1_STEP1: CV = PV for HC1 Configure HSC 1.	**Network 1** **LD SM0.0** **MOVB 16#A0, SMB47** **MOVD +6, SMD52** **ATCH HSC1_STEP1, 13**
SBR 1 Network 2		Clear unwanted interrupts caused by machine vibration.	**Network 2** **LD SM0.0** **CEVNT 13**

10.11 Logical operations

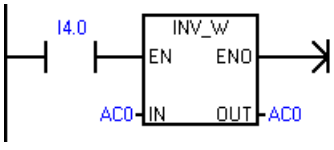
10.11.1 Invert

LAD / FBD	STL	Description
	**INVB OUT** **INVW OUT** **INVD OUT**	The Invert Byte, Invert Word, and Invert Double Word instructions form the one's complement of the input IN and load the result into the memory location OUT

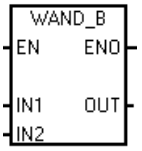
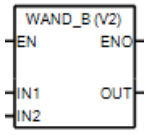
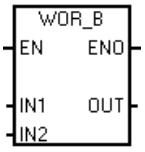
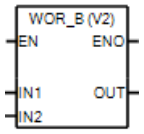
Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address	• SM1.0 Result of operation = zero

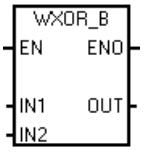
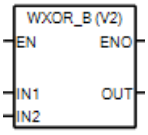
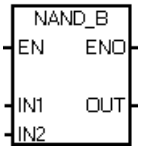
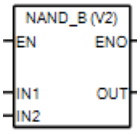
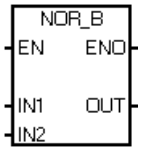
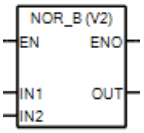
Input / output	Data type	Operand
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC,*VD, *LD, *AC
	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

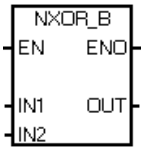
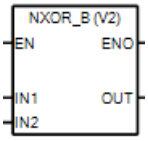
Example: Invert instruction

LAD		STL
	Invert word value in AC0. Result is put in AC0. Invert Word input AC0 1101 0111 1001 0101 Execute one's complement inversion Invert Word output AC0 0010 1000 0110 1010	**Network 1** **LD I4.0** **INVW AC0**

10.11.2 AND, OR, XOR, NAND, NOR, and NXOR

LAD / FBD	STL	Description
 WAND_W WAND_DW	**ANDB** **IN1,IN2,OUT** **ANDW** **IN1,IN2,OUT** **ANDD IN1,** **IN2,OUT**	The AND Byte, AND Word, and AND Double Word instructions logically AND the corresponding bits of two input values IN1 and IN2 and load the result into a memory location assigned to OUT. • LAD and FBD: IN1 AND IN2 = OUT • STL: IN1 AND OUT = OUT
 WAND_W WAND_DW	Legacy: **MOVB IN1,IN2** **ANDB IN2,OUT** **MOVW IN1,IN2** **ANDW IN2,OUT** **MOVD IN1,IN2** **ANDD IN2,OUT**	
 WOR_W WOR_DW	**ORB** **IN1,IN2,OUT** **ORW** **IN1,IN2,OUT** **ORD** **IN1,IN2,OUT**	The OR Byte, OR Word, and OR Double Word instructions logically OR the corresponding bits of two input values IN1 and IN2 and load the result into a memory location assigned to OUT. • LAD and FBD: IN1 OR IN2 = OUT • STL: IN1 OR OUT = OUT
 WOR_W WOR_DW	Legacy: **MOVB IN1,IN2** **ORB IN2,OUT** **MOVW IN1,IN2** **ORW IN2,OUT** **MOVD IN1,IN2** **ORD IN2,OUT**	

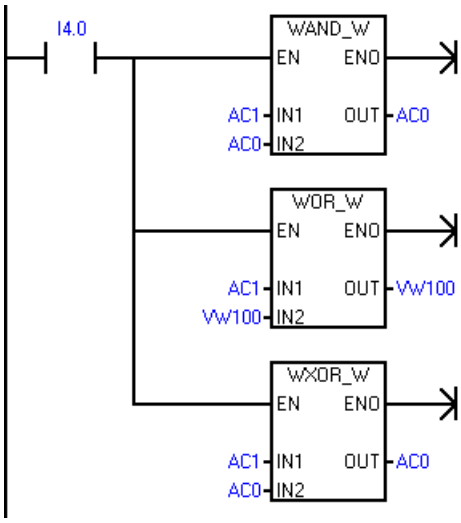
LAD / FBD	STL	Description
 WXOR_W WXOR_DW	**XORB** **IN1,IN2,OUT** **XORW** **IN1,IN2,OUT** **XORD** **IN1,IN2,OUT**	The Exclusive OR Byte, Exclusive OR Word, and Exclusive OR Double Word instructions logically XOR the corresponding bits of two input values IN1 and IN2 and load the result into a memory location OUT. • LAD and FBD: IN1 XOR IN2 = OUT • STL: IN1 XOR OUT = OUT
 WXOR_W WXOR_DW	Legacy: **MOVB IN1,IN2** **XORB IN2,OUT** **MOVW IN1,IN2** **XORW IN2,OUT** **MOVD IN1,IN2** **XORD IN2,OUT**	
 NAND_W NAND_DW	NANDB IN1,IN2,OUT **NANDW IN1,IN2,OUT** **NANDD IN1,IN2,OUT**	The NAND Byte, NAND Word, and NAND Double Word instructions logically NAND the corresponding bits of two input values IN1 and IN2 and load the result into a memory location assigned to OUT. • LAD and FBD: IN1 NAND IN2 = OUT • STL: IN1 NAND OUT = OUT
 NAND_W NAND_DW	Legacy: **MOVB IN1,IN2** **NANDB IN2,OUT** **MOVW IN1,IN2** **NANDW IN2,OUT** **MOVD IN1,IN2** **NANDD IN2,OUT**	
 NOR_W NOR_DW	**NORB** **IN1,IN2,OUT** **NORW** **IN1,IN2,OUT** **NORD** **IN1,IN2,OUT**	The NOR Byte, NOR Word, and NOR Double Word instructions logically NOR the corresponding bits of two input values IN1 and IN2 and load the result into a memory location assigned to OUT. • LAD and FBD: IN1 NOR IN2 = OUT • STL: IN1 NOR OUT = OUT
 NOR_W NOR_DW	Leacy: **MOVB IN1,IN2** **NORB IN2,OUT** **MOVW IN1,IN2** **NORW IN2,OUT** **MOVD IN1,IN2** **NORD IN2,OUT**	

LAD / FBD	STL	Description
 NXOR_W NXOR_DW	**NXORB IN1,IN2,OUT** **NXORW IN1,IN2,OUT** **NXORD IN1,IN2,OUT**	The Exclusive NOR Byte, Exclusive NOR Word, and Exclusive NOR Double Word instructions logically NXOR the corresponding bits of two input values IN1 and IN2 and load the result into a memory location OUT. • LAD and FBD: IN1 NXOR IN2 = OUT • STL: IN1 NXOR OUT = OUT
 NXOR_W NXOR_DW	Legacy: **MOVB IN1,IN2** **NXORB IN2,OUT** **MOVW IN1,IN2** **NXORW IN2,OUT** **MOVD IN1,IN2** **NXORD IN2,OUT**	

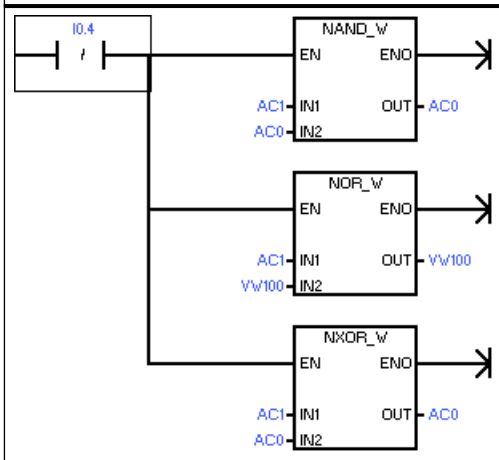
Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address	• SM1.0 Result of operation = zero

Input / output	Data type	Operand
IN1, IN2	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD
	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *AC, *LD
	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Example: AND, OR, and Exclusive OR instructions

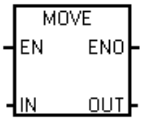
LAD		STL
	AC1 0001 1111 0110 1101 AND AC0 1101 0011 1110 0110 equals AC0 0001 0011 0110 0100 AC1 0001 1111 0110 1101 OR VW100 1101 0011 1010 0000 equals VW100 1101 1111 1110 1101 AC1 0001 1111 0110 1101 XOR AC0 0001 0011 0110 0100 equals AC0 0000 1100 0000 1001	**Network 1** **LD I4.0** **ANDW AC1, AC0, ACO** **ORW AC1, VW100, VW100** **XORW AC1, AC0, AC0**

Example: NAND, NOR, and NXOR instructions

LAD		STL
	AC1 0001 1111 0110 1101 NAND AC0 1101 0011 1110 0110 equals AC0 1110 1100 1001 1011 AC1 0001 1111 0110 1101 NOR VW100 1101 0011 1010 0000 equals VW100 0010 0000 0001 0010 AC1 0001 1111 0110 1101 NXOR AC0 1110 1100 1001 1011 equals AC0 0000 1100 0000 1001	**Network 1** **LDN I0.4** **NANDW AC1, AC0, AC0** **NORW AC1, VW100, VW100** **NXORW AC1, AC0, AC0**

10.12 Move

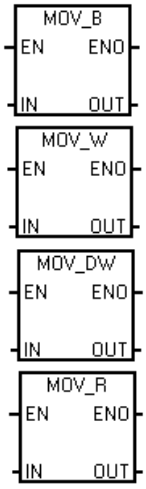
10.12.1 Move

LAD / FBD	STL	Description
	**MOVE IN, OUT**	The Move instructions move a data value between the same data types from a source (constant or memory location) IN to a new memory location OUT, without changing the value stored in a source memory location.

Non-fatal error with ENO = 0	SM bits affected
• 0006H Indirect address	None

Input / output	Data type	Operand
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, Constant
	WORD, INT	IW, QW, VW, MW, SMW, SW, Constant
	DWORD, DINT	ID, QD, VD, MD, SMD, SD, LD, Constant
	REAL	ID, QD, VD, MD, SMD, SD, LD, Constant
	UDT	
	ARRAY	
	STRING	
	FB Instance	
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB,
	WORD, INT	IW, QW, VW, MW, SMW, SW
	DWORD, DINT, REAL	ID, QD, VD, MD, SMD, SD, LD
	UDT	
	ARRAY	
	STRING	
	FB Instance	

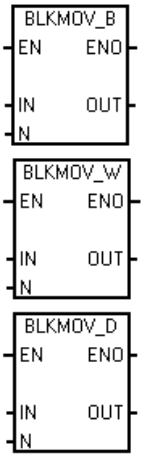
10.12.2 Move byte, word, double word, or real

LAD / FBD	STL	Description
	**MOVB IN, OUT** **MOVW IN, OUT** **MOVD IN, OUT** **MOVR IN, OUT**	The Move Byte, Move Word, Move Double Word, and Move Real instructions move a data value from a source (constant or memory location) IN to a new memory location OUT, without changing the value stored in a source memory location. Use the Move Double Word instruction to create a pointer. For more information, refer to the section on pointers and indirect addressing (Page 121).

Non-fatal error with ENO = 0	SM bits affected
• 0006H Indirect address	None

Input / output	Data type	Operand
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
	WORD, INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	DWORD, DINT	ID, QD, VD, MD, SMD, SD, LD, HC, &VB, &IB, &QB, &MB, &SB, &T, &C, &SMB, &AIW, &AQW, AC, *VD, *LD, *AC, Constant
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC
	WORD, INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AQW, *VD, *LD, *AC
	DWORD, DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

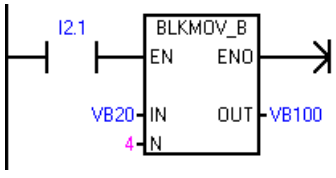
10.12.3 Block move

LAD / FBD	STL	Description
	**BMB IN, OUT, N** **BMW IN, OUT, N** **BMD IN, OUT, N**	The Block Move Byte, Block Move Word, and Block Move Double Word instructions move an assigned block of data values from a source memory location (starting address IN and successive addresses) to a new memory location (starting address OUT and successive addresses). Parameter N assigns the number of bytes, words, or double words to move. The block of data values stored in the source location are not changed. N has a range of 1 to 255.

Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address • 0091H Operand out of range	None

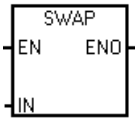
Input / output	Data type	Operand
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC
	WORD, INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AIW, *VD, *LD, *AC
	DWORD, DINT	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC
	WORD, INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AQW, *VD, *LD, *AC
	DWORD, DINT	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, Constant, *VD, *LD, *AC

Example: Block Move instruction

LAD			STL	
	Move (copy) data in source four byte address sequence (VB20 to VB23) to destination four byte address sequence (VB100 to VB103).		**Network 1** **LD I2.1** **BMB VB20, VB100, 4**	
Source data values	30	31	32	33
Source data addresses	VB20	VB21	VB22	VB23

LAD			STL	
If I2.1 = 1, then execute BLKMOV_B to move source data values to destination addresses				
Destination data values	30	31	32	33
Destination data addresses	VB100	VB101	VB102	VB103

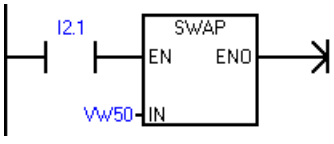
10.12.4 Swap bytes

LAD / FBD	STL	Description
	**SWAP IN**	The Swap Bytes instruction exchanges the most significant byte with the least significant byte of the word IN.

Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address	None

Input / output	Data type	Operand
IN	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC

Example: Swap instructions

LAD	STL
	**Network 1** **LD I2.1** **SWAP VW50**

Hexadecimal data values	D6	C3
Data addresses	VB50	VB51
If I2.1 = 1, then execute SWAP to exchange the byte data in a data word		
Hexadecimal data values	C3	D6
Data addresses	VB50	VB51

10.12.5 Move byte immediate (read and write)

LAD / FBD	STL	Description
MOV_BIR EN ENO IN OUT	**BIR IN, OUT**	The Move Byte Immediate Read instruction reads the state of physical input IN and writes the result to the memory address OUT, but the process image register is not updated.
MOV_BIW EN ENO IN OUT	**BIW IN, OUT**	The Move Byte Immediate Write instruction reads the data from the memory address IN and writes to physical output OUT, and the corresponding process image location.

Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address • Unable to access expansion module	None

Input / output	Data type	Operand
IN (BIR)	BYTE	IB, *VD, *LD, *AC
IN (BIW)	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC
OUT (BIR)	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC
OUT (BIW)	BYTE	QB, *VD, *LD, *AC

10.13 Program control

10.13.1 FOR-NEXT loop

LAD / FBD	STL	Description
FOR EN ENO INDX INIT FINAL	**FOR INDX, INIT, FINAL**	The FOR instruction executes the instructions between the FOR and the NEXT instructions. You assign the index value or current loop count INDX, the starting loop count INIT, and the ending loop count FINAL.
LAD (NEXT) FBD NEXT	**NEXT**	The NEXT instruction marks the end of the FOR loop program segment.

Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address	None

Input / output	Data type	Operand
INDX	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
INIT, FINAL	INT	VW, IW, QW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant

Use the FOR and NEXT instructions to execute a program segment in a loop that is repeated for the assigned count. Each FOR instruction requires one NEXT instruction. You place a FOR-NEXT loop within a FOR-NEXT loop to a maximum nesting depth of eight.

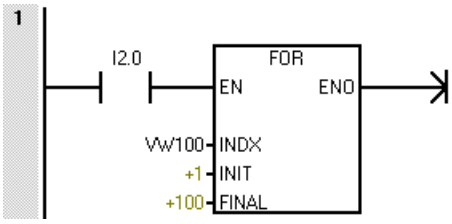
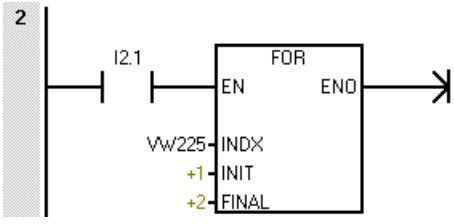
If you enable a FOR-NEXT loop, the execution loop continues until it finishes the iterations, unless you change the FINAL value from within the loop itself. You can change the values while the FOR-NEXT loop is in the looping process. When the loop is enabled again, it copies the INIT value to the INDX value (current loop number).

For example, given an INIT value of 1 and a FINAL value of 10, the instructions between the FOR instruction and the NEXT instruction are executed 10 times with the INDX value being incremented: 1, 2, 3, ... 10.

If the INIT value is greater than the FINAL value, the loop is not executed. After each execution of the instructions between the FOR instruction and the NEXT instruction, the INDX value is incremented and the result is compared to the final value. If the INDX is greater than the final value, the execution loop is terminated.

For STL, if the top of the logic stack value is 1 when your program enters the FOR-NEXT loop, then the top of the logic stack will be 1 when your program exits the FOR-NEXT loop.

Example: FOR-NEXT loop

LAD		STL
1 	When I2.0 is ON, the outside loop (Network 1 - 4) is executed 100 times.	**Network 1** **LD I2.0** **FOR VW100, +1, +100**
2 	The inside loop (Network 2 - 3) is executed twice for each execution of the outside loop when I2.1 is on.	**Network 2** **LD I2.1** **FOR VW225, +1, +2**

LAD		STL
3 (NEXT)	End of inside loop	`Network 3` `NEXT`
4 (NEXT)	End of outside loop	`Network 4` `NEXT`

10.13.2 JMP (jump to label)

You can use the JMP (Jump) instruction in the main program, in subroutines, in function blocks or in interrupt routines. The JMP and its corresponding LBL (Label) instruction must be located within the same program segment either in the main program, a subroutine, or an interrupt routine.

Note

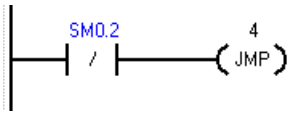
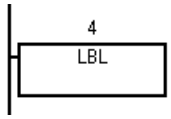
You cannot jump from the main program to a label in either a subroutine, a function block or an interrupt routine. Likewise, you cannot jump from a subroutine or interrupt routine to a label outside that subroutine or interrupt routine.

You can use a Jump instruction within an SCR program segment, but the corresponding Label instruction must be located within the same SCR program segment.

LAD / FBD	STL	Description
LAD n (JMP) FBD n JMP	`JMP N`	The JMP (Jump) instruction performs a branch to the label N within the program.
LAD n LBL FBD n LBL	`LBL N`	The LBL (Label) instruction marks the location of the jump destination n.

Input / output	Data type	Operand
n	WORD	Constant (0 to 255)

Example: Jump to Label

LAD		STL
	If the retentive data has not been lost, jump to label 4.	**Network 1** **LDN SM0.2** **JMP 4**
	Label 4	**Network 2** **LBL 4**

10.13.3 SCR (sequence control relay)

SCR (Sequence Control Relay) instructions provide a simple yet powerful state control programming technique for a LAD, FBD, or STL program. Whenever an application consists of a sequence of operations that must be performed repetitively, you can use SCRs to structure the program so that it corresponds directly to your application. As a result, you can program and debug the application quickly and easily.

WARNING

S bit usage in POUs

Do not use the same S bit in more than one POU. For example, if you use S0.1 in the main program, do not use it in a subroutine.

Multiple POUs accessing the same S bit could result in unexpected process operation, possibly resulting in death or severe personal injury.

Check your program to ensure that multiple POUs do not access the same S bit.

Note

SCR programming restrictions

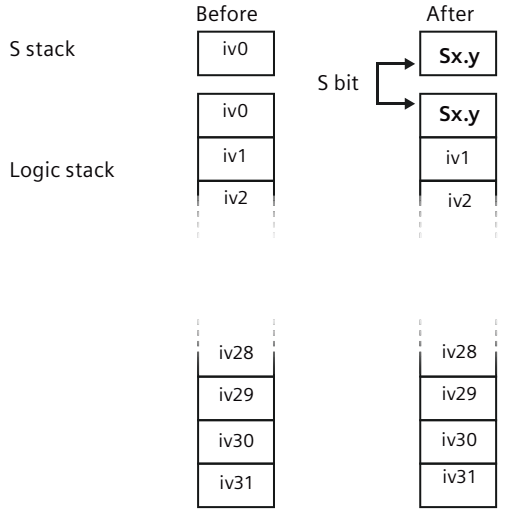
- You cannot jump into or out of an SCR segment; however, you can use Jump and Label instructions to jump around SCR segments or to jump within an SCR segment.
- You cannot use the END instruction in an SCR segment.

LAD	FBD	STL	Description
S_bit SCR	S_bit SCR	`LSCR S_bit`	The **SCR** instruction loads the SCR and logic stacks with the value of the S bit referenced by the instruction. The resulting value of the SCR stack either energizes or de-energizes the SCR stack. The value of the SCR stack is copied to the top of the logic stack so that LAD boxes or output coils can be tied directly to the left power rail and no preceding contact instruction is required.
S_bit SCRT	S_bit SCRT	`SCRT S_bit`	
	CSCRE	`CSCRE`	The **SCRT** instruction identifies the SCR bit to be enabled (the next S_bit to be set). When power flows to the coil or FBD box, the CPU turns on the referenced S_bit and turns off the S_bit of the LSCR instruction (that enabled this SCR segment).
SCRE	SCRE	`SCRE`	The **CSCRE** (conditional SCR end) instruction, for STL and FBD, terminates execution of the SCR segment when enabled. For LAD, a conditional contact placed before a **SCRE** coil performs the conditional SCR end function. The **SCRE** (unconditional SCR end) instruction, for STL and FBD, terminates execution of the SCR segment. For LAD, an SCRE coil connected directly to the power rail performs the unconditional SCR end function.

Input / output	Data type	Operand
S_bit	BOOL	S

S stack and logic stack interaction

Load the value of Sx.y onto the SCR and logic stacks.



The figure shows the S stack and the logic stack and the effect of executing the Load SCR instruction.

Controlling program flow with SCR segments

The main program consists of instructions that execute sequentially once per scan of the PLC. For many applications, it may be appropriate to logically divide the main program into a series of operational steps that mirror steps within a controlled process (for example, a series of machine operations).

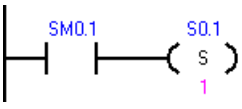
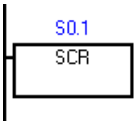
One way to logically divide a program into multiple steps is to use SCR segments. SCR segments can divide your program into a single stream of sequential steps, or into multiple streams that can be active simultaneously. It is possible to have a single stream conditionally diverge into multiple streams, and to have multiple streams conditionally re-converge into a single stream.

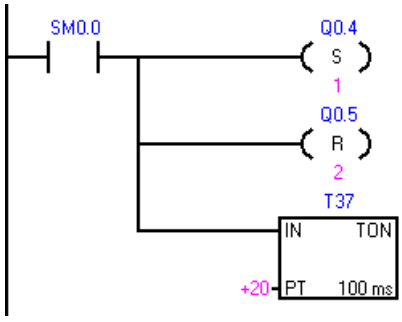
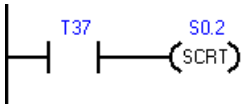
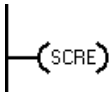
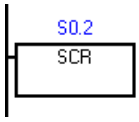
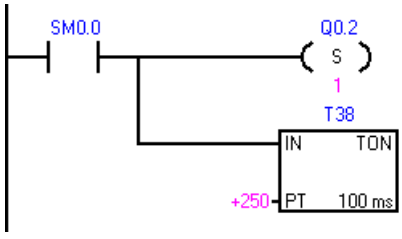
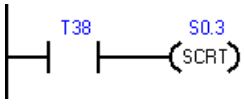
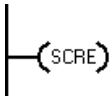
SCR operations

- SCR (Load SCR) marks the beginning of an SCR segment, and the SCRE (End SCR) marks the end of an SCR program segment. All logic between the SCR and the SCRE instructions is dependent upon the value of the S stack for its execution. Logic between SCRE and the next SCR instruction is not dependent on the value of the S stack.
- SCRT (SCR Transition) transfers control from an active SCR segment to another SCR segment. Execution of the SCR transition instruction, when it has power flow, will reset the S bit of the currently active SCR segment and will set the S bit of the referenced segment. Resetting the S bit of the active segment does not affect the S stack at the time the SCR Transition instruction executes. Consequently, the SCR segment remains energized until it is exited.
- The STL only instruction CSRE (Conditional SCR End) exits an active SCR segment without executing the instructions between the CSRE and the SCRE (SCR End) instructions. The Conditional SCR End instruction does not affect any S bit nor does it affect the S stack.

Example: SCR sequential control flow

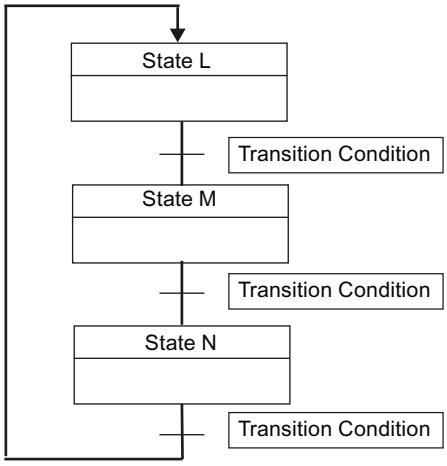
In the following sample program, the first scan bit SM0.1, is used to set S0.1, which will be the active State 1 on the first scan. After a 2-second delay, T37 causes a transition to State 2. This transition deactivates the State 1 SCR (S0.1) segment and activates the State 2 SCR (S0.2) segment.

LAD		STL
	On the first scan enable state 1.	**Network 1** **LD SM0.1** **S S0.1, 1**
	Beginning of state 1 control region.	**Network 2** **LSCR S0.1**

LAD		STL
	Control the signals for street 1: 1. Set: Turn on the red light. 2. Reset: Turn off the yellow and green lights. 3. Start a 2-second timer.	**Network 3** **LD SM0.0** **S Q0.4, 1** **R Q0.5, 2** **TON T37, +20**
	After a 2 second delay, transition to state 2.	**Network 4** **LD T37** **SCRT S0.2**
	End of SCR region for state 1.	**Network 5** **SCRE**
	Beginning of state 2 control region.	**Network 6** **LSCR S0.2**
	Control the signals for street 2: 1. Set: Turn on the green light. 2. Start a 25-second timer.	**Network 7** **LD SM0.0** **S Q0.2, 1** **TON T38, +250**
	After a 25 second delay, transition to state 3.	**Network 8** **LD T38** **SCRT S0.3**
	End of SCR region for state 2.	**Network 9** **SCRE**

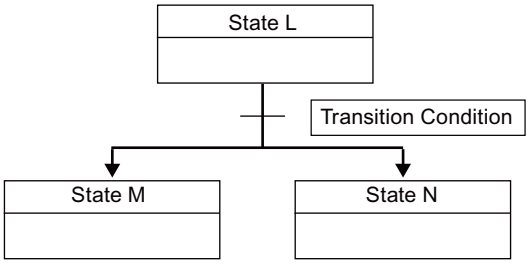
Sequential control flow

A process with a well-defined sequence of steps is easy to model with SCR segments. For example, consider a cyclical process, with 3 steps, that should return to the first step when the third has completed.



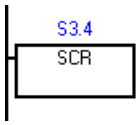
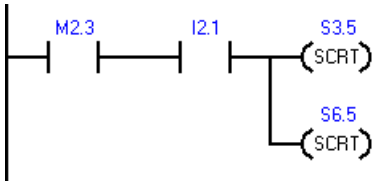
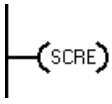
Divergent control flow

In many applications, a single stream of sequential states must be split into two or more different streams. When a control stream diverges into multiple streams, all outgoing streams must be activated simultaneously.



The divergence of control streams can be implemented in an SCR program by using multiple SCRT instructions enabled by the same transition condition, as shown in the following example.

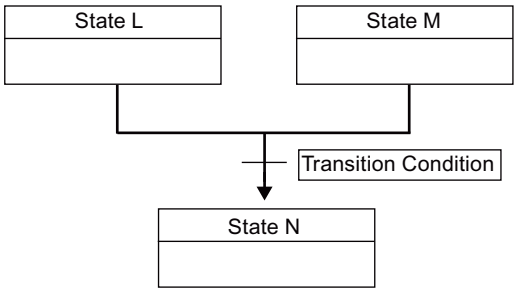
Example: SCR divergent flow control

LAD		STL
	Beginning of state L control region	**Network 1** **LSCR S3.4**
	S3.5: Transition to state M S6.5: Transition to state N	**Network 2** **LD M2.3** **A I2.1** **SCRT S3.5** **SCRT S6.5**
	End of the state region for state L	**Network 3** **SCRE**

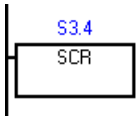
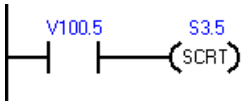
Convergent flow control

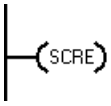
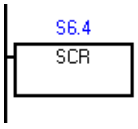
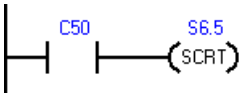
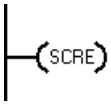
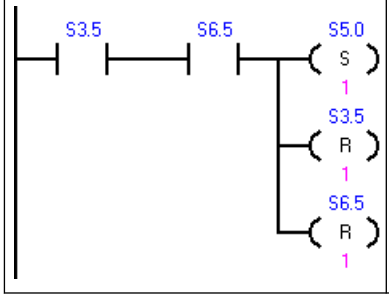
When streams converge, all incoming streams must be complete before the next state is executed.

The convergence of control streams can be implemented in an SCR program by making the transition from state L to state L' and by making the transition from state M to state M'. When both SCR bits representing L' and M' are true, state N is enabled as shown in the following example.



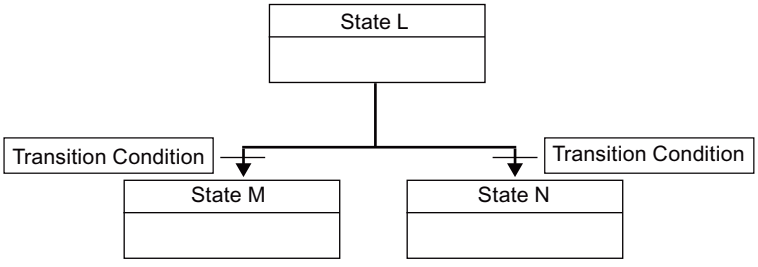
Example: SCR convergent flow control

LAD		STL
	Beginning of state L control region	**Network 1** **LSCR S3.4**
	Transition to State L'	**Network 2** **LD V100.5** **SCRT S3.5**

LAD		STL
	End of SCR region for state L	**Network 3** **SCRE**
	Beginning of state M control region	**Network 4** **LSCR S6.4**
	Transition to state M'	**Network 5** **LD C50** **SCRT S6.5**
	End of SCR region for state M	**Network 6** **SCRE**
	When both State L' and State M' are activated: • Enable State N (S5.0) • Reset State L' (S3.5) • Reset State M' (S6.5)	**Network 7** **LD S3.5** **A S6.5** **S S5.0, 1** **R S3.5, 1** **R S6.5, 1**

Divergence of a control stream, depending on transition conditions

In other situations, a control stream might be directed into one of several possible control streams, depending upon which transition condition becomes true first.



Example: SCR divergent flow control, depending of transition conditions

LAD		STL
S3.4 SCR	Beginning of state L control region	**Network 1** **LSCR S3.4**
M2.3 S3.5 (SCRT)	Transition to state M	**Network 2** **LD M2.3** **SCRT S3.5**
I3.3 S6.5 (SCRT)	Transition to state N	**Network 3** **LD I3.3** **SCRT S6.5**
(SCRE)	End of SCR region for state L	**Network 4** **SCRE**

10.13.4 END, STOP, and WDR (watchdog timer reset)

LAD	FBD	STL	Description
(END)	END	**END**	The conditional END instruction terminates the current scan based upon the condition of the preceding logic. You can use the conditional END instruction in the main program, but you cannot use it in either subroutines, function blocks or interrupt routines.
(STOP)	STOP	**STOP**	The conditional STOP instruction terminates the execution of your program by causing a transition of the CPU from RUN to STOP mode. If the STOP instruction is executed in an interrupt routine, the interrupt routine is terminated immediately and all pending interrupts are ignored. Remaining actions in the current scan cycle are completed, including execution of the main user program. The transition from RUN to STOP mode is made at the end of the current scan.
(WDR)	WDR	**WDR**	The watchdog reset instruction retriggers the system watchdog timer and adds 500 milliseconds to the time allowed for the scan to complete before a watchdog timeout error occurs.

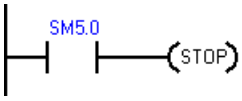
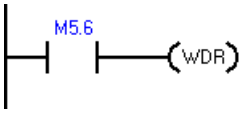
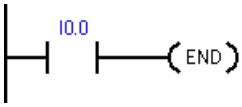
Watchdog timer operation

When the CPU is in RUN mode, the duration of the main scan is limited to 500 milliseconds, by default. If the duration of the main scan exceeds 500 milliseconds, then the CPU automatically transitions to STOP mode and the non-fatal error 001AH (Scan watchdog timeout) is issued.

You can extend the duration of the Main Scan by executing the Watchdog Reset (WDR) instruction in your program. The scan watchdog timeout period is reset to 500 milliseconds each time the WDR instruction is executed.

However, there is an absolute maximum main scan duration of 5 seconds. The CPU will unconditionally transition to STOP mode if the current scan duration reaches 5 seconds.

Example: STOP, END, and WDR (Watchdog reset) instructions

LAD		STL
	When an I/O error is detected, force the transition to STOP mode.	**Network 1** **LD SM5.0** **STOP**
	When M5.6 is ON, execute the watchdog reset instruction to extend the allowed scan time by 500 milliseconds.	**Network 1** **LD SM5.6** **WDR**
	When I0.0 is ON, terminate the current scan.	**Network 1** **LD I.0** **END**

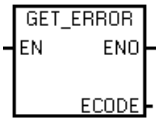
Note

If you expect your scan time to exceed 500 ms, or if you expect a burst of interrupt activity that prevents returning to the main scan for more than 500 ms, you should use the watchdog reset instruction to retrigger the watchdog timer.

Use the watchdog reset instruction carefully. If program execution loops prevent scan completion or excessively delay the completion of the scan, then the following processes are inhibited until the scan cycle is completed.

- Communications (except Freeport mode)
- I/O updating (except Immediate I/O)
- Forced values updating
- SM bit updating (SM0, SM5 to SM29 are not updated)
- RUN-time diagnostics
- STOP instruction, when used in an interrupt routine

10.13.5 GET_ERROR (Get non-fatal error code)

LAD / FBD	STL	Description
	**GERR ECODE**	The get non-fatal error code instruction stores the CPU's current non-fatal error code in the location assigned to ECODE. After the error code is stored, the non-fatal error code is cleared in the CPU.

Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address	None

Input / output	Data type	Operand
ECODE	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC

Non-fatal run-time errors also affect certain special memory error flag addresses that can be evaluated along with the GET_ERROR instruction to determine the cause of a run-time fault. In the event that the generic error flag SM4.3 = 1 (Run-time programming problem) is active, a GET_ERROR execution can be used to identify the specific error.

Non-fatal error code 0000H indicates that no actual error currently exists. In the case of a temporary run-time non-fatal error, a GET_ERROR (ECODE output) produces a non-zero error value and then the next program scan can produce a zero ECODE value.

You should use compare logic to save the ECODE value in another memory location. Your program can then test the saved error code value and begin a programmatic reaction.

Note

The error codes for the ECODE output are listed in the PLC non-fatal error codes table (see reference below). The error code values are in hexadecimal (16#xxxx).

See Also

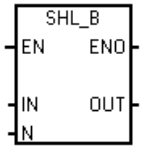
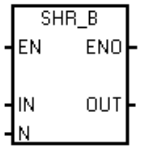
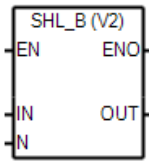
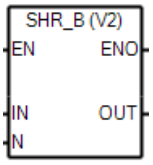
PLC non-fatal error codes (Page 1062)

PLC non-fatal error SM flags (Page 1064)

10.14 Shift and rotate

10.14.1 Shift and rotate

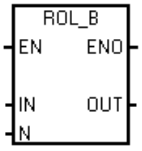
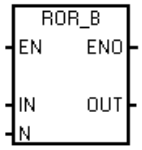
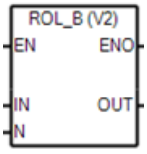
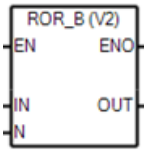
Shift instructions (only the byte size LAD box is illustrated, the others are similar)

LAD / FBD	STL	Shift type	Description
 	**SLB IN, N, OUT** **SRB IN, N, OUT**	Shift left byte Shift right byte	The shift instructions shift the bit values of input value IN right or left by the bit position shift count N and load the result in the memory location assigned to OUT. The shift instructions fill empty bit positions with zero as each bit is shifted out. If the shift count N is greater than or equal to the maximum allowed (8 for byte operations, 16 for word operations, and 32 for double word operations), the value is shifted the maximum number of times for the operation. If the shift count is greater than 0, the overflow memory bit SM1.1 is set to the value of the last bit shifted out. The SM1.0 zero memory bit is set if the result of the shift operation is zero. Byte operations are unsigned. For word and double word operations, the sign bit is shifted when you use signed data values.
 	Legacy **SLB OUT, N** **SRB OUT, N**	Shift left byte Shift right byte	
SHL_W SHR_W	**SLW IN, N, OUT** **SRW IN, N, OUT**	Shift left word Shift right word	
	Legacy **SLW OUT, N** **SRW OUT, N**	Shift left word Shift right word	
SHL_DW SHR_DW	**SLD IN, N, OUT** **SRD IN, N, OUT**	Shift left double word Shift right double word	
	Legacy **SLD OUT, N** **SRD OUT, N**	Shift left double word Shift right double word	

Non-fatal errors with ENO=0	SM bits affected
• 0006H Indirect address	• SM1.0 Result of operation = zero • SM1.1 Overflow (last bit shifted out)

Input / output	Data type	Operand
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
	WORD, INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
	DWORD, DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC
	WORD, INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
	DWORD, DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant

Rotate instructions (only the byte size LAD box is illustrated, the others are similar)

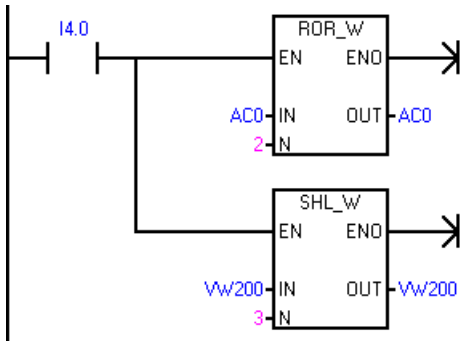
LAD / FBD	STL	Rotate type	Description
 	**RLB IN, N, OUT** **RRB IN, N, OUT**	Rotate left byte Rotate right byte	The rotate instructions rotate the bit values of input value IN right or left by the bit position rotate count N and load the result in the memory location assigned to OUT. The rotate operation is circular. If the rotate count is greater than or equal to the maximum for the operation (8 for a byte operation, 16 for a word operation, or 32 for a double-word operation), the CPU performs a modulo operation on the rotate count to obtain a valid shift count before the rotation is executed. This result is a shift count of 0 to 7 for byte operations, 0 to 15 for word operations, and 0 to 31 for double-word operations. If the rotate count is 0, a rotate operation is not performed. If the rotate operation is performed, the overflow bit SM1.1 is set to the value of the last bit rotated out. If the rotate count is not an integer multiple of 8 (for byte operations), 16 (for word operations), or 32 (for double-word operations), the last bit value rotated out is copied to the overflow memory bit SM1.1. The zero memory bit SM1.0 is set when the value to be rotated is zero. Byte operations are unsigned. For word and double word operations, the sign bit is rotated when you use signed data types.
 	Legacy **RLB OUT, N** **RRB OUT, N**	Rotate left byte Rotate right byte	
ROL_W ROR_W	**RLW IN, N, OUT** **RRW IN, N, OUT**	Rotate left word Rotate right word	
	Legacy **RLW OUT, N** **RRW OUT, N**	Rotate left word Rotate right word	
ROL_DW ROR_DW	**RLD IN, N, OUT** **RRD IN, N, OUT**	Rotate left double word Rotate right double word	
	Legacy **RLD OUT, N** **RRD OUT, N**	Rotate left double word Rotate right double word	

Non-fatal errors with ENO=0	SM bits affected
• 0006H Indirect address	• SM1.0 Result of operation = zero • SM1.1 Overflow (last bit shifted out)

Input / output	Data type	Operand
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
	WORD, INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
	DWORD, DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC
	WORD, INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
	DWORD, DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant

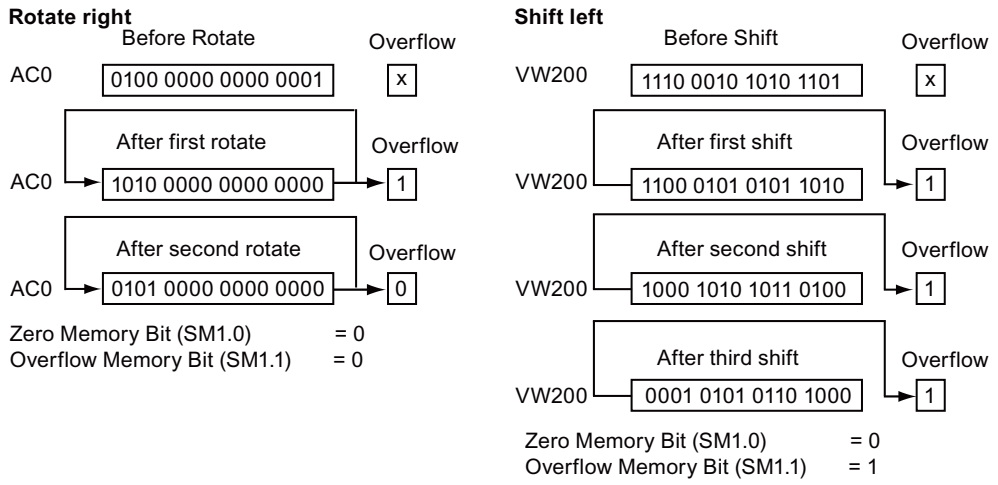
Example: Shift and Rotate instructions

LAD



STL

```
Network 1
LD I4.0
RRW AC0, 2, AC0
SLW VW200, 3, VW200
```



10.14.2 Shift register bit

The Shift Register Bit instruction shifts a bit value into the Shift Register. This instruction provides an easy method for sequencing and controlling product flow or data. Use this instruction to shift the entire register one bit, once per scan.

LAD / FBD	STL	Description
SHRB EN ENO DATA S_BIT N	**SHRB DATA, S_bit, N**	The shift register bit instruction shifts the bit value of DATA into the Shift Register. S_BIT assigns the location of the least significant bit of the shift register. N assigns the length of the Shift Register and the direction of the shift (Shift Plus = N, Shift Minus = -N). Each bit value shifted out by the SHRB instruction is copied to the overflow memory bit SM1.1. The shift register bits are defined by both the least significant bit S_BIT location and the number of bits assigned by the length N.

Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address • 0091H Operand out of range • 0092H Error in count field	• SM1.1 Overflow (last bit shifted out)

Input / output	Data type	Operand
DATA, S_bit	BOOL	I, Q, V, M, SM, S, T, C, L
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant

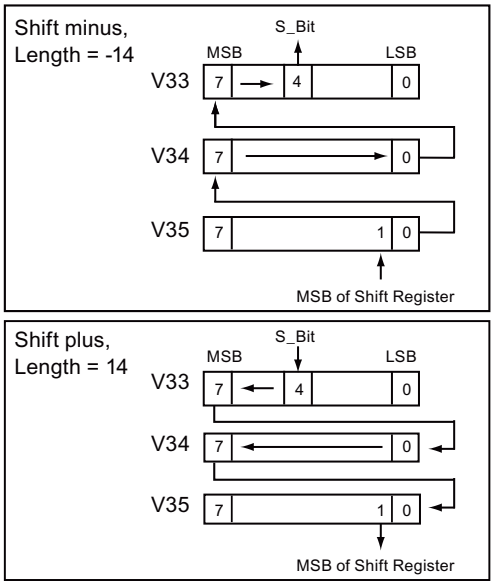
Use the following equation to compute the address of the most significant bit of the Shift Register (MSB.b):

MSB.b = *[(Byte of S_BIT) + ([N] - 1 + (bit of S_BIT)) / 8].[remainder of the division by 8]*

For example: if S_BIT is V33.4 and N is 14, the following calculation shows that the MSB.b is V35.1.

MSB.b = V33 + ([14] - 1 +4)/8
= V33 + 17/8
= V33 + 2 with a remainder of 1
= V35.1

The following figure shows bit shifting for negative and positive values of N.



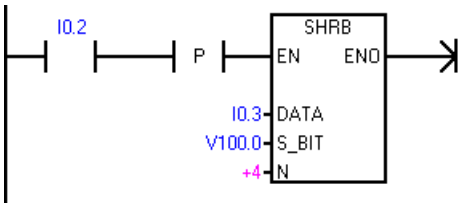
A Shift Minus operation is indicated by a negative value of length N. The input value of DATA shifts into the most significant bit of the shift register, and shifts out of the least significant bit location assigned by S_BIT. The data shifted out is then placed in the overflow memory bit SM1.1.

A Shift Plus operation is indicated by a positive value of length N. The input value of DATA shifts into the least significant bit location assigned by S_BIT and out of the most significant bit of the Shift Register. The bit value shifted out is then placed in the overflow memory bit SM1.1.

The maximum length of the shift register assigned by N is 64 bits (positive or negative).

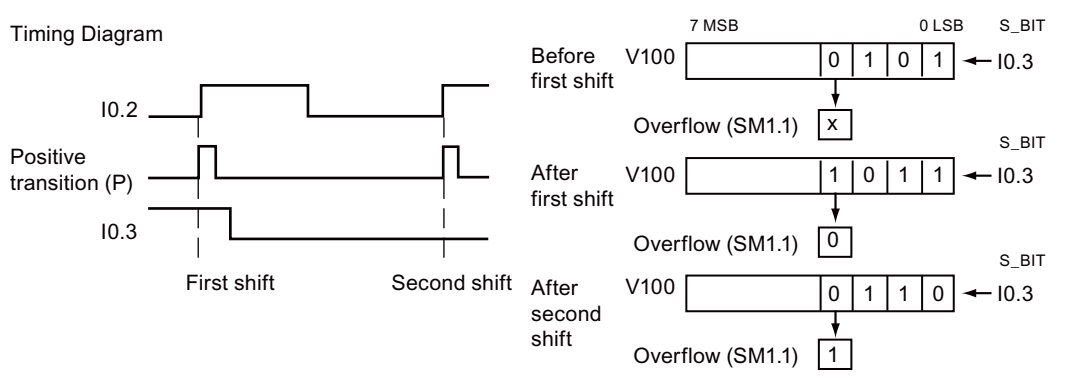
Example: SHRB instruction

LAD



STL

```
Network 1
LD I0.2
EU
SHRB I0.3, V100.0, +4
```



10.15 String

10.15.1 String (Get length, copy, and concatenate)

SLEN (String length)

LAD / FBD	STL	Description
STR_LEN EN ENO IN OUT	**SLEN IN, OUT**	The string length instruction returns the length in bytes of the string specified by IN. Note: Because Chinese characters are not represented by a single byte, the STR_LEN function does not return the number of characters in a string containing Chinese characters.

Error conditions with ENO = 0	SM bits affected
• 0006H Indirect address • 0091H Operand out of range	None

Input / output	Data type	Operand
IN	STRING	VB, LB, *VD, *LD, *AC, Constant string
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC

SCPY and SCAT (String copy and string concatenate)

LAD / FBD	STL	Description
STR_CPY EN ENO IN OUT	**SCPY IN, OUT**	The copy string instruction copies the string assigned by IN to the string assigned by OUT.
STR_CAT EN ENO IN OUT	**SCAT IN, OUT**	The concatenate string instruction appends the string assigned by IN to the end of the string assigned by OUT.

Note: The STR_CPY and STR_CAT instructions operate on bytes and not characters. Because Chinese characters are not represented by a single byte, unexpected results can occur with the STR_CPY and STR_CAT instructions with strings containing Chinese characters. If you know the number of bytes that a character string occupies, you can use the STR_CPY and STR_CAT instructions with the correct number of bytes.

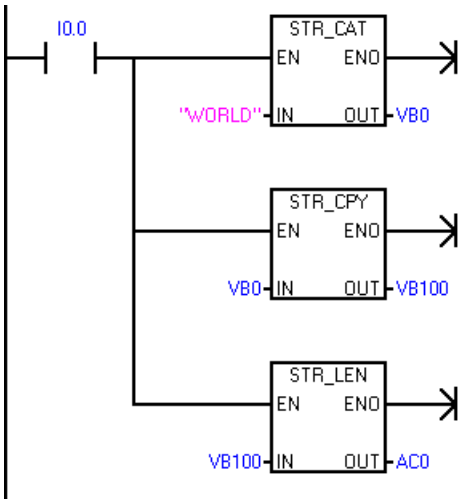
Error conditions with ENO = 0	SM bits affected
• 0006H Indirect address • 0091H Operand out of range	None

Note: If the length of the input string (IN) exceeds the length of the output variable (OUT), the error code 0091H is reported at runtime.

Inputs/Outputs	Data types	Operands
IN	STRING	VB, LB, *VD, *LD, *AC, Constant string
OUT	STRING	VB, LB, *VD, *LD, *AC

Example: Concatenate string, copy string, and string length Instructions

LAD



1. Append the string "WORLD" to the string at VB0.
2. Copy the string at VB0 to a new string at VB100.
3. Get the length of the string that starts at VB100.

STL

```
Network 1
LD I0.0
SCAT "WORLD", VB0
SCPY VB0, VB100
SLEN VB100, AC0
```

Before executing the program

VB0						VB6
6	'H'	'E'	'L'	'L'	'O'	' '

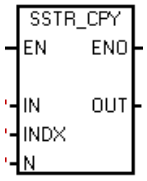
After executing the program

VB0											VB11
11	'H'	'E'	'L'	'L'	'O'	' '	'W'	'O'	'R'	'L'	'D'

VB100											VB111
11	'H'	'E'	'L'	'L'	'O'	' '	'W'	'O'	'R'	'L'	'D'

AC0
11

10.15.2 Copy substring from string

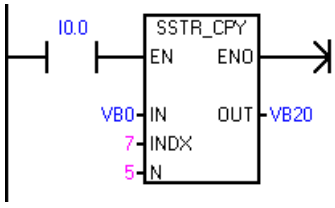
LAD / FBD	STL	Description
	**SSCPY IN, INDX, N, OUT**	The copy substring from string instruction copies the assigned number of characters N from the string specified by IN, starting at the index INDX, to a new string assigned by OUT. Note: The SSTR_CPY instruction operates on bytes and not characters. Because Chinese characters are not represented by a single byte, unexpected results can occur with the SSTR_CPY instruction with strings containing Chinese characters. If you know the number of bytes that a character string occupies, you can use the SSTR_CPY instruction with the correct number of bytes.

Non-fatal errors with ENO=0	SM bits affected
• 0006H Indirect address • 0091H Operand out of range • 009BH Index = 0	None

Note: If the length of the input string (IN) exceeds the length of the output variable (OUT), the error code 0091H is reported at runtime.

Input / output	Data type	Operand
IN	STRING	VB, LB, *VD, *LD, *AC, Constant string
OUT	STRING	VB, LB, *VD, *LD, *AC
INDX, N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant

Example: Copy substring instruction

AD		STL
	Starting at the seventh byte after the byte count in the string at VB0, copy 5 bytes to a new string at VB20.	**Network 1** **LD I0.0** **SSCPY VB0, 7, 5, VB20**

Before executing the program

VB0											VB11
11	'H'	'E'	'L'	'L'	'O'	' '	'W'	'O'	'R'	'L'	'D'

After executing the program

VB20					VB25
5	'W'	'O'	'R'	'L'	'D'

10.15.3 Find string and first character within string

LAD / FBD	STL	Description
STR_FIND EN ENO IN1 OUT IN2	**SFND IN1, IN2, OUT**	STR_FIND searches for the first occurrence of the string IN2 within the string IN1. The search begins at the starting position assigned by the initial value of OUT (which must be in the range of 1 to the IN1 string length before STR_FIND execution). If a sequence of characters is found that matches exactly the string IN2, the position of the first character in the sequence within the IN1 string is written to OUT. If the string IN2 was not found in the string IN1, then OUT is set to 0.
CHR_FIND EN ENO IN1 OUT IN2	**CFND IN1, IN2, OUT**	CHR_FIND searches the string IN1 for the first occurrence of any character from the character set in the string IN2. The search begins at starting position assigned by the initial value of OUT (which must be in the range of 1 to the IN1 string length before CHR_FIND execution). If a matching character is found, then the position of the character is written to OUT. If no matching character is found, OUT is set to 0.

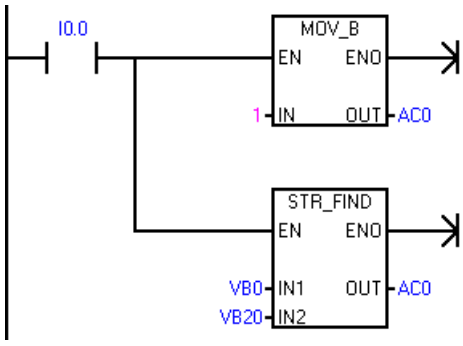
Note: Because Chinese characters are not represented by a single byte, and the string instructions operate on bytes and not characters, unexpected results can occur with the STR_FIND and CHR_FIND instructions with strings containing Chinese characters.

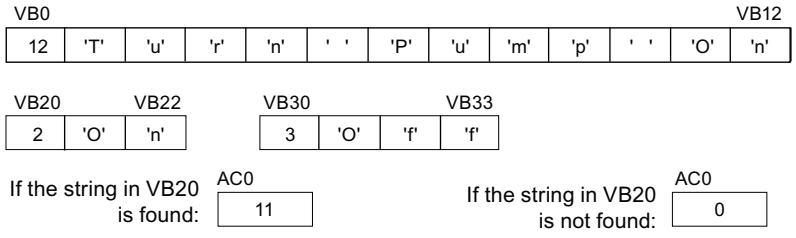
Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address • 0091H Operand out of range • 009BH Index = 0	None

Input / output	Data type	Operand
IN1, IN2	STRING	VB, LB, *VD, *LD, *AC, Constant String
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC

Example: Find string within string instruction

A string stored at VB0 is used as a command for turning a pump on or off. A string "On" is stored at VB20 and a string "Off" is stored at VB30. The result of the find string within string instruction is stored in AC0 (the OUT parameter). If the result is not 0, then the string 'On' was found in the command string (VB12).

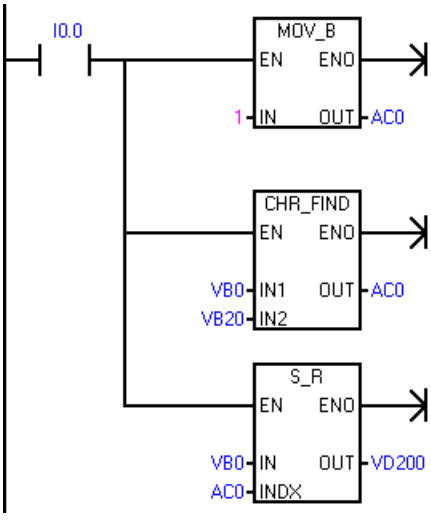
LAD		STL
	1. Set AC0 to 1. (AC0 is used as the OUT parameter.) 2. Search the string at VB0 for the string at VB20 ('On'), starting at the first position (AC0=1).	**Network 1** **LD I0.0** **MOVB 1, AC0** **SFND VB0, VB20, AC0**



Example: Find character within string instruction

A string stored at VB0 contains the temperature. The string constant at IN1 provides all the numeric characters (0-9, +, and -) that can identify a temperature number in a string. CHR_FIND execution finds the starting position of the character "9" in the VB0 string and then S_R execution converts the real number characters into a real number value. VD200 is used to store the real-number value of the temperature.

LAD



1. Set AC0 to 1. (AC0 is used as the OUT parameter and points to the first character position in the string.)
2. Find the first numeric character in the string stored at VB0.
3. Convert the string to a real number value.

STL

```
Network 1
LD I0.0
MOVB 1, AC0
CFND VB0, VB20, AC0
STR VB0, AC0, VD200
```

VB0											VB11
11	'T'	'e'	'm'	'p'	' '	' '	'9'	'8'	'.'	'6'	'F'

VB20												VB32
12	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'	'0'	'+'	'-'

Starting position of the temperature stored in VB0: AC0 = 7

Real-number value of the temperature: VD200 = 98.6

10.16 Table

10.16.1 Add to table

LAD / FBD	STL	Description
AD_T_TBL EN ENO DATA TBL	**ATT DATA, TBL**	The add to table instruction adds word values DATA to a table TBL. The first value in the table is the maximum table length TL. The second value is the entry count EC, which stores the number of entries in the table and is updated automatically. New data are added to the table after the last entry. Each time new data are added to the table, the entry count is incremented. A table can have up to 100 data entries.

Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address • 0091H Operand out of range • SM1.4 Table overfill	• SM1.4 Set to 1 if you try to overfill the table

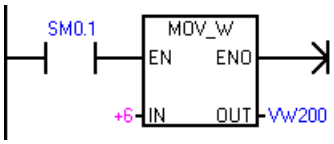
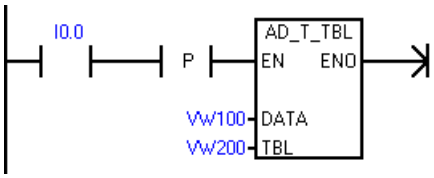
Input / output	Data type	Operand
DATA	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
TBL	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, *VD, *LD, *AC **Note**: TBL of this instruction doesn't have DB2 area support.

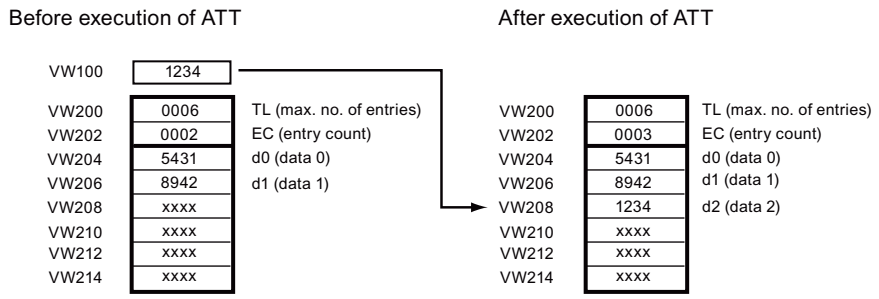
Note

To create a table, first make an entry for the maximum number of table entries. If you do not do this, then you cannot make any entries in the table.

Edge trigger instructions must activate all table read and table write instructions.

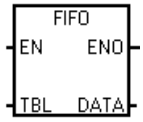
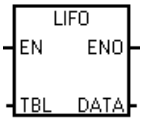
Example: Add to Table instruction

LAD		STL
	On the first scan only, load the maximum table length 6 to VW200.	**Network 1** **LD SM0.1** **MOVW +6, VW200**
	When I0.0 makes a transition to 1, add a third data value (from VW100) to the table at VW200. Two data entries were previously stored in the table which can hold up to six entries.	**Network 2** **LD I0.0** **ATT VW100, VW200**



10.16.2 First-in-first-out and last-in-first-out

Table 10-20 FIFO and LIFO instructions

LAD / FBD	STL	Description
	**FIFO TBL, DATA**	The first-in-first-out instruction moves the oldest (or first) entry in a table to an output memory address, by removing the first entry in the assigned table (TBL) and moving the value to the location assigned by DATA. All other entries of the table are shifted up one location. The entry count in the table is decremented for each FIFO execution.
	**LIFO TBL, DATA**	The last-in-first-out instruction moves the newest (or last) entry in the table to an output memory address, by removing the last entry in the table (TBL) and moving the value to the location assigned by DATA. The entry count in the table is decremented for each LIFO execution.

Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address • 0091H Operand out of range • SM1.5: Attempt to remove entry from empty table	• SM1.5: Attempt to remove entry from empty table

Input / output	Data type	Operand
TBL	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, *VD, *LD, *AC
DATA	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AQW, *VD, *LD, *AC

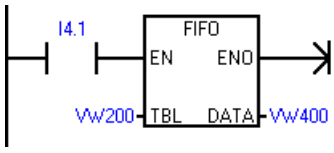
Note

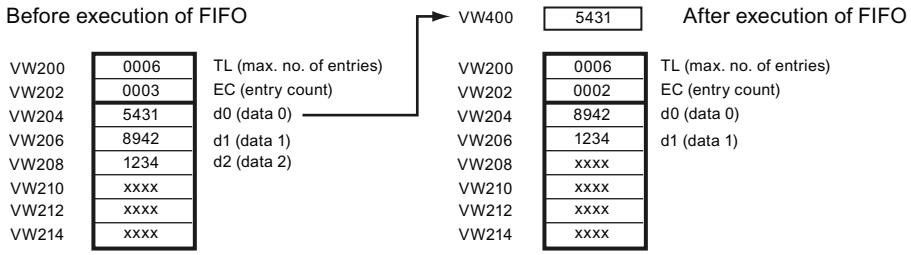
All table read and table write instructions must be activated by edge trigger instructions.

To create a table, you must first make an entry for the maximum number of table entries before any entries can be put in the table.

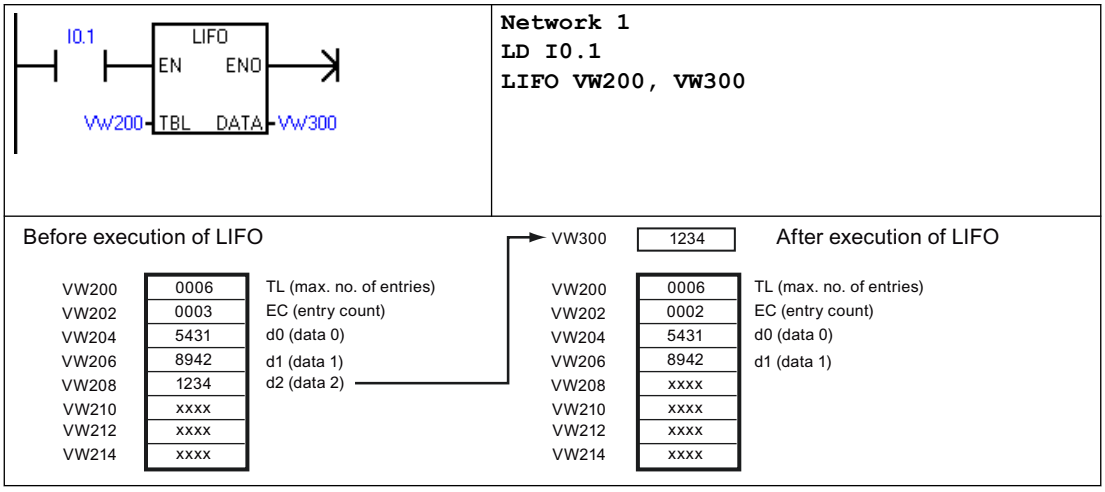
Example: FIFO instruction

LAD	STL

	`Network 1`
	`LD I4.1`
	`FIFO VW200, VW400`



Example: LIFO instruction



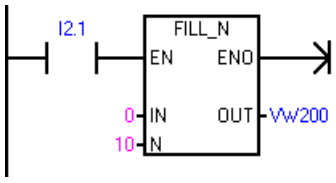
10.16.3 Memory fill

LAD / FBD	STL	Description
FILL_N EN ENO IN OUT N	`FILL IN, OUT, N`	The memory fill instruction writes N consecutive words, beginning at address OUT, with the word value contained in address IN. N has a range of 1 to 255.

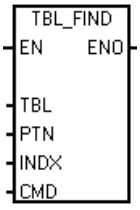
Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address • 0091H Operand out of range	None

Input / output	Data type	Operand
IN	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
OUT	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AQW, *VD, *LD, *AC

Example: Memory fill instruction

LAD	STL
	**Network 1** **LD I2.1** **FILL +0, VW200, 10**

10.16.4 Table find

LAD / FBD	STL	Description
	**FND= TBL, PTN, INDX** **FND<> TBL, PTN, INDX** **FND< TBL, PTN, INDX** **FND> TBL, PTN, INDX**	The Table Find instruction searches a table for data that matches your search criteria. The Table Find instruction searches the table TBL, starting with the table entry INDX, for the data value or pattern PTN that matches the search criteria defined by CMD. The command parameter CMD is given a numeric value of 1 to 4 that corresponds to =, <>, <, and >, respectively. If a match is found, the INDX points to the matching entry in the table. To find the next matching entry, the INDX must be incremented before invoking the table find instruction again. If a match is not found, the INDX has a value equal to the entry count. A table can have up to 100 data entries. The data entries (area to be searched) are numbered from 0 to a maximum value of 99.

Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address • 0091H Operand out of range	None

Input / output	Data type	Operand
TBL	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, *VD, *LD, *AC
PTN	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant

Input / output	Data type	Operand
INDX	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
CMD	BYTE	Constant: • 1 = Equal (=) • 2 = Not Equal (<>) • 3 = Less Than (<) • 4 = Greater Than (>)

Note

When you use the table find instruction with tables generated with the Add-to-table, Last-in-first-out, and First-in-first-out instructions, the entry count and the data entries correspond directly. The maximum-number-of-entries word required for the Add-to-table, Last-in-first-out, or First-in-first-out instructions is not required by the Table find instruction. See the following figure.

Consequently, you should set the TBL operand of a Find instruction to one-word address (two bytes) higher than the TBL operand of a corresponding the Add-to-table, Last-in-first-out, or First-in-first-out instruction.

Differences in table formats for ATT, LIFO, FIFO, and TBL_FIND instructions

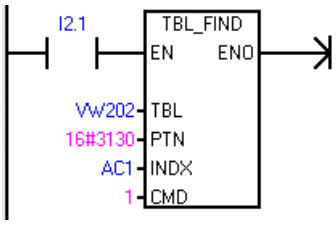
Table format for ATT, LIFO, and FIFO

VW200	0006	TL (max. no. of entries)
VW202	0006	EC (entry count)
VW204	xxxx	d0 (data 0)
VW206	xxxx	d1 (data 1)
VW208	xxxx	d2 (data 2)
VW210	xxxx	d3 (data 3)
VW212	xxxx	d4 (data 4)
VW214	xxxx	d5 (data 5)

Table format for TBL_FIND

VW202	0006	EC (entry count)
VW204	xxxx	d0 (data 0)
VW206	xxxx	d1 (data 1)
VW208	xxxx	d2 (data 2)
VW210	xxxx	d3 (data 3)
VW212	xxxx	d4 (data 4)
VW214	xxxx	d5 (data 5)

Example: Table Find instruction

LAD	STL
	**Network 1** **LD I2.1** **FND= VW202, 16#3130, AC1**

When I2.1 is on, search the table for a value equal to 3130 HEX.

VW202	0006	EC (entry count)
VW204	3133	d0 (data 0)
VW206	4142	d1 (data 1)
VW208	3130	d2 (data 2)
VW210	3030	d3 (data 3)
VW212	3130	d4 (data 4)
VW214	4541	d5 (data 5)

If the table was created using ATT, LIFO, and FIFO instructions, VW200 contains the maximum number of allowed entries and is not required by the Find instructions.

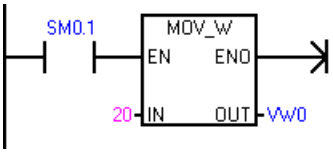
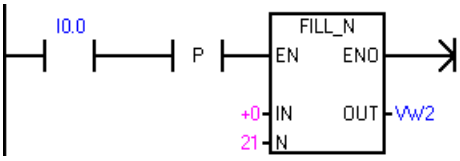
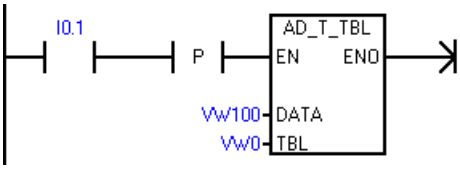
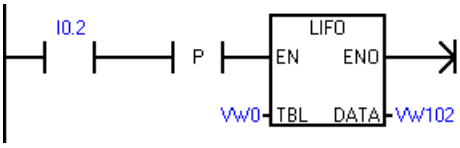
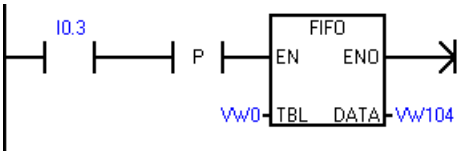
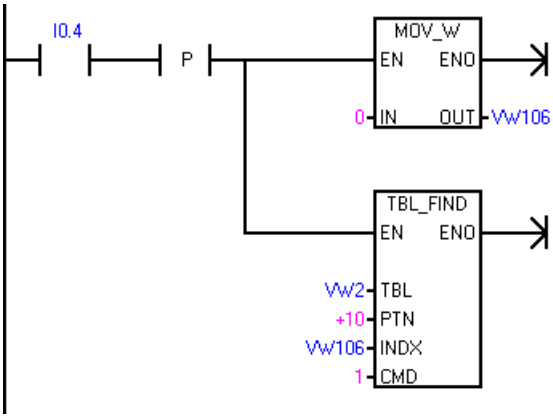
AC1	0	AC1 must be set to 0 to search from the top of table.
Execute a table search		
AC1	2	AC1 contains the data entry number corresponding to the first match in the table (d2).
AC1	3	Increment the INDX by one, before searching the remaining enties of the table.
Execute a table search		
AC1	4	AC1 contains the data entry number corresponding to the second match in the table (d4).
AC1	5	Increment the INDX by one, before searching the remaining enties of the table.
Execute a table search		
AC1	6	AC1 contains a value equal to the entry count. The entire table has been searched without finding another match.
AC1	0	Before the table can be searched again, the INDX value must be reset to 0.

Example: Table

The following program creates a table with 20 entries. The first memory location of the table contains the length of the table (in this case 20 entries). The second memory location shows the current number of table entries. The other locations contain the entries. A table can have up to 100 entries. It does not include the parameters defining the maximum length of the table or the actual number of entries (here VW0 and VW2). The actual number of entries in the table (here VW2) is automatically incremented or decremented by the CPU with every command.

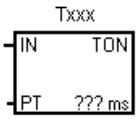
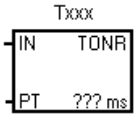
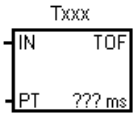
Before you work with a table, assign the maximum number of table entries. Otherwise, you cannot make entries in the table. Also, be sure that all read and write commands are activated with edge instructions.

To search the table, the index (VW106) must set to 0 before doing the find. If a match is found, the index will have the table entry number, but if no match is found, the index will match the current entry count for the table (VW2).

LAD		STL
	Create table with 20 entries starting with memory location 4. • On the first scan, define the maximum length of the table.	**Network 1** **LD SM0.1** **MOVW +20, VW0**
	Reset table with input I0.0. • On the rising edge of I0.0, fill memory locations from VW2 with "+0".	**Network 2** **LD I0.0** **EU** **FILL +0, VW2, 21**
	Write value to table with input I0.1. • On the rising edge of I0.1, copy value of memory location VW100 to table.	**Network 3** **LD I0.1** **EU** **ATT VW100, VW0**
	Read last table value with input I0.2. • Move the last table value to location VW102. This reduces the number of entries. On the rising edge of I0.2, move last table value to VW102.	**Network 4** **LD I0.2** **EU** **LIFO VW0, VW102**
	Read first table value with input I0.3. • Move the first table value to location VW104. This reduces the number of entries. On the rising edge of I0.3, move first table value to VW104.	**Network 5** **LD I0.3** **EU** **FIFO VW0, VW104**
	Search table for the first location that has a value of 10. • On the rising edge of I0.4, reset index pointer. • Find a table entry that equals 10.	**Network 6** **LD I0.4** **EU** **MOVW +0, VW106** **FND= VW2, +10, VW106**

10.17 Timers

10.17.1 Timer instructions

LAD / FBD	STL	Description
	**TON Txxx, PT**	Use **TON** On-delay timers for a timing a single time interval.
	**TONR Txxx, PT**	Use **TONR** On-delay retentive timers for accumulating the time value of many timed intervals.
	**TOF Txxx, PT**	Use the **TOF** Off-delay timer for extending a time interval past an OFF (or FALSE) condition, such as a delay time for cooling a motor.

Input / output	Data type	Operand
Txxx	WORD	Timer number (T0 to T511)
IN	BOOL	I, Q, V, M, SM, S, T, C, L, Power Flow
PT	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant

Timer resolution

TON, TONR, and TOF timers are available in three resolutions. The resolution is determined by the timer number, as shown below. Each unit of the current value is a multiple of the time base. For example, a count of 50 on a 10 ms timer represents 500 ms of elapsed time.

Your Txxx timer number assignment determines the resolution of the timer. When a valid timer number is assigned, the resolution is displayed in LAD or FBD timer boxes.

Timer number and resolution options

Timer type	Resolution	Maximum value	Timer number
TON, TOF	1 ms	32.767 s	T32, T96, T262-T267,
	10 ms	327.67 s	T33-T36, T97-T100, T292-T315
	100 ms	3276.7 s	T37-T63, T101-T255, T414-T511

Timer type	Resolution	Maximum value	Timer number
TONR	1 ms	32.767 s	T0, T64, T256-T261
	10 ms	327.67 s	T1-T4, T65-T68, T268-T291
	100 ms	3276.7 s	T5-T31, T69-T95, T316-T413

Note

Avoid timer number conflicts

You cannot use the same timer number for both a TON and TOF timer. For example, you cannot have both a TON T32 and a TOF T32.

Note

To guarantee a minimum time interval, increase the preset value (PV) by 1.

For example: To ensure a minimum timed interval of at least 2100 ms for a 100-ms timer, set the PV to 22.

TON and TONR timer operation

The TON and TONR instructions begin timing when the enabling input IN is ON. When the current value is equal to or greater than the preset time, the timer bit is set ON.

- The current value of a TON timer is cleared when the enabling input is OFF.
- The current value of the TONR timer is maintained when the enabling input is OFF. You can use the TONR timer to accumulate time when the input IN is ON. Use the Reset instruction (R) to clear the current value of the TONR.
- Both the TON and the TONR timers continue timing after the preset time is reached, and they stop timing at the maximum value of 32,767.

TOF timer operation

The TOF instruction is used to delay turning an output OFF for a fixed period of time after the input turns OFF. When the enabling input turns ON, the timer bit turns ON immediately, and the current value is set to 0. When the input turns OFF, timing begins and continues until the current time equals the preset time.

- When the preset is reached, the timer bit turns OFF and the current value stops incrementing; however, if the enabling input turns ON again before the TOF reaches the preset value, the timer bit remains ON.
- The enabling input must make an ON-OFF transition for a TOF timer to begin timing the OFF-delay time interval.
- If a TOF timer is inside an SCR region and the SCR region is inactive, then the current value is set to 0, the timer bit is turned OFF, and the current value does not increment.

Type	Current >= Preset	State of IN, the enabling input	Power cycle / first scan
TON	Timer bit ON Current value continues timing to 32,767	ON: Current value = timing value OFF: Timer bit OFF, current value = 0	Timer bit = OFF Current value = 0
TONR[1]	Timer bit ON Current value continues timing to 32,767	ON: Current value = timing value OFF: Timer bit and current value maintain last state and value	Timer bit = OFF Current value can be maintained[1]
TOF	Timer bit OFF Current = Preset, stops timing	ON: Timer bit ON, current value = 0 OFF: Timer begins timing after ON-to-OFF transition	Timer bit = OFF Current value = 0
[1] The retentive timer current value can be assigned for retention through a power cycle. See Configuring the retentive ranges for details (Page 190).			

Note

Using the Reset instruction with timer instructions

The TONR timer can only be reset with the Reset (R) instruction.

The TON and TOF timers can be reset by the timer's enable input and also the Reset (R) instruction.

The Reset instruction performs the following actions:

- Timer bit = OFF
- Timer current value = 0
- After a reset, TOF timers require the enable input to make the transition from ON-to-OFF in order restart the OFF-delay timer.

10.17.2 Timer programming tips and examples

Timer types

You can use timers to implement time-based counting functions. The S7-200 SMART instruction set provides three different types of timers.

- On-Delay Timer (TON) for timing a single interval
- Retentive On-Delay Timer (TONR) for accumulating a number of timed intervals
- Off-Delay Timer (TOF) for extending time past an off (or false condition), such as for cooling a motor after it is turned off

Addressing timer values

The meaning of the T number depends on the context in your program.

- "T37" assigned to a timer box identifies which timer is to use.
- "T37" assigned to normally open contacts addresses the Boolean T37 timer bit.
- "T37" assigned to integer operations addresses the T37 current time value, as a data word.

1-millisecond resolution

The 1-ms timers count the number of 1-ms timer intervals that have elapsed since the active 1-ms timer was enabled. The execution of the timer instruction starts the timing; however, the 1-ms timers are updated (timer bit and timer current) every millisecond asynchronous to the scan cycle. In other words, the timer bit and timer current are updated multiple times throughout any scan that is greater than 1 ms.

The timer instruction is used to turn the timer on, reset the timer, or, in the TONR timer, to turn the timer off.

Since the timer can be started anywhere within a millisecond, the preset must be set to one time interval greater than the minimum desired timer interval. For example, to guarantee a timed interval of at least 56 ms using a 1-ms timer, the preset time value should be set to 57.

10-millisecond resolution

The 10-ms timers count the number of 10-ms timer intervals that have elapsed since the active 10-ms timer was enabled. The execution of the timer instruction starts the timing; however the 10-ms timers are updated at the beginning of each scan cycle (in other words, the timer current and timer bit remain constant throughout the scan), by adding the accumulated number of 10-ms intervals (since the beginning of the previous scan) to the current value for the active timer.

Since the timer can be started anywhere within a 10-ms interval, the preset must be set to one time interval greater than the minimum desired timer interval. For example, to guarantee a timed interval of at least 140 ms using a 10-ms timer, the preset time value should be set to 15.

100-millisecond resolution

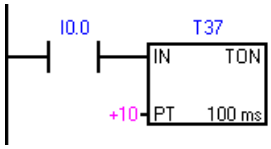
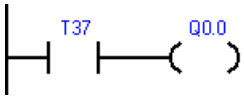
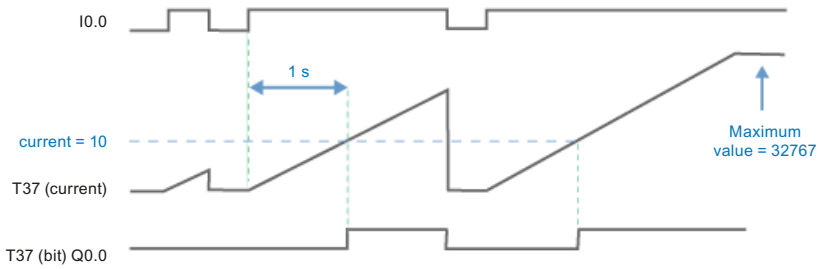
The 100-ms timers count the number of 100-ms timer intervals that have elapsed since the active 100-ms timer was last updated. These timers are updated by adding the accumulated number of 100-ms intervals (since the previous scan cycle) to the timer's current value when the timer instruction is executed.

The current value of a 100-ms timer is updated only if the timer instruction is executed. Consequently, if a 100-ms timer is enabled but the timer instruction is not executed each scan cycle, the current value for that timer is not updated and it loses time. Likewise, if the same 100-ms timer instruction is executed multiple times in a single scan cycle, the number of 100-ms intervals is added to the timer's current value multiple times, and it gains time. 100-ms timers should only be used where the timer instruction is executed exactly once per scan cycle.

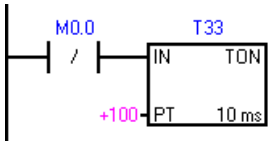
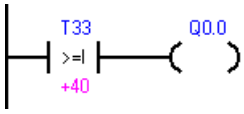
Since the timer can be started anywhere within a 100-ms interval, the preset must be set to one time interval greater than the minimum desired timer interval. For example, to

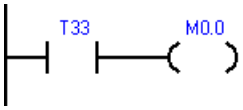
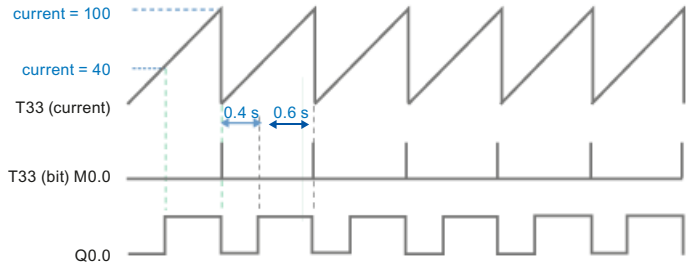
guarantee a timed interval of at least 2100 ms using a 100-ms timer, the preset time value should be set to 22.

TON On-delay timer example

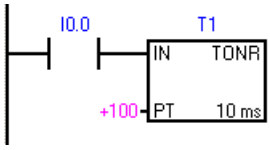
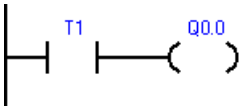
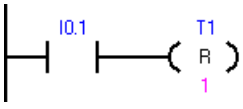
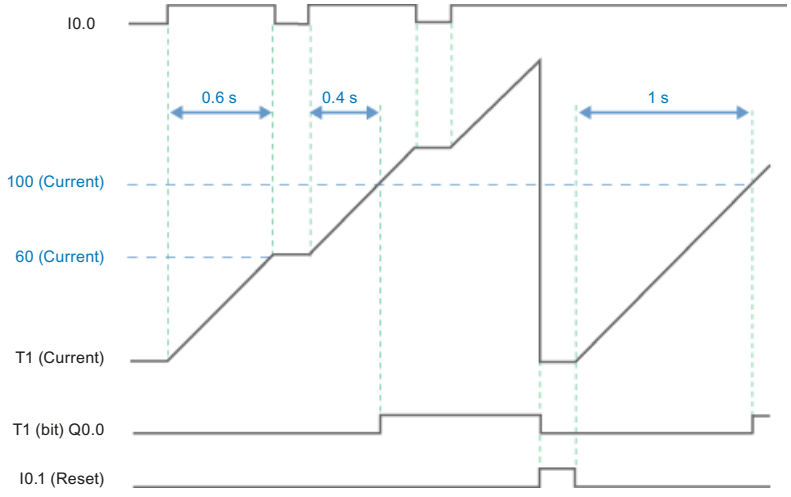
LAD		STL
	100 ms timer T37 times out after 1 s (10 x 100 ms) • I0.0 ON = T37 enabled, • I0.0 OFF = disable and reset T37	**Network 1** **LD I0.0** **TON T37, +10**
	T37 bit is controlled by timer T37	**Network 2** **LD T37** **= Q0.0**
Timing Diagram 		

TON self-resetting On-delay timer example

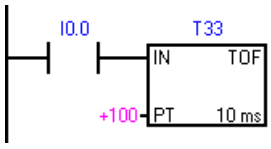
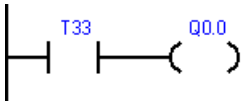
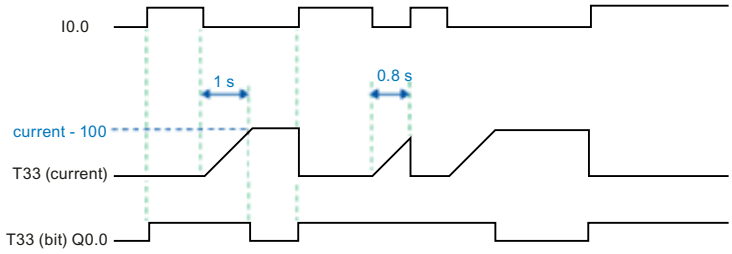
LAD		STL
	10 ms timer T33 times out after 1 s (100 x 10 ms). M0.0 pulse is too fast to monitor with Status view.	**Network 1** **LDN M0.0** **TON T33, +100**
	The Compare contact becomes TRUE at a rate that is visible in Status view. Turn ON Q0.0 after (40 x 10 ms) for 40% OFF / 60% ON.	**Network 2** **LDW>= T33, +40** **= Q0.0**

LAD		STL
	T33 (bit) pulse is too fast to monitor with Status view. Reset the timer with M0.0 after the (100 x 10 ms) period.	**Network 3** **LD T33** **= M0.0**
Timing Diagram 		

TONR retentive On-delay timer example

LAD		STL
	10 ms TONR timer T1 times out at PT = 1 s (100 x 10 ms).	**Network 1** **LD I0.0** **TONR T1, +100**
	T1 bit is controlled by timer T1. Q0.0 is ON after the timer accumulates a total of 1 second.	**Network 2** **LD T1** **= Q0.0**
	TONR timers must be reset by a Reset instruction with a T address. Reset timer T1 (current value and bit) when I0.1 is on.	**Network 3** **LD I0.1** **R T1, 1**
Timing Diagram 		

TOF Off-delay timer example

LAD		STL
	10-ms timer T33 times out after 1 s (100 x 10 ms). • I0.0 ON-to-OFF = T33 enabled • I0.0 OFF-to-ON=disable and reset T33	**Network 1** **LD I0.0** **TOF T33, +100**
	Timer T33 controls Q0.0 through timer contact T33.	**Network 2** **LD T33** **= Q0.0**
Timing Diagram 		

Effect of timer resolution on when timer bits and current time values are updated

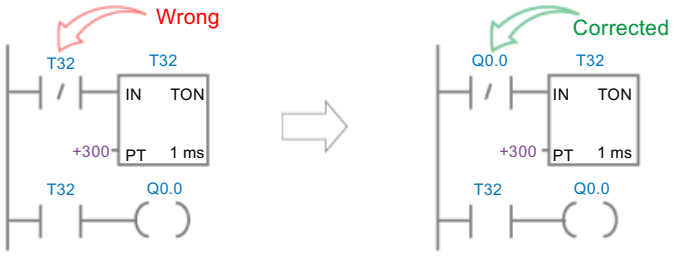
- **1 ms** timer: The timer bit and the current value are updated asynchronous to the scan cycle. For scans greater than 1 ms, the timer bit and the current value are updated multiple times throughout the scan.
- **10 ms** timer: The timer bit and the current value are updated at the beginning of each scan cycle. The timer bit and current value remain constant throughout the scan. Time intervals that accumulate during the scan are added to the current value at the start of each scan.
- **100 ms** timer: The timer bit and current value are updated when the instruction is executed; therefore, ensure that your program executes the instruction for a 100-ms timer only once per scan cycle in order for the timer to maintain the correct timing.

Example: automatically retriggered One-shot timers

The corrected examples use the normally closed contact Q0.0 instead of the timer bit as the timer enabling input. This ensures that output Q0.0 is turned ON for one scan cycle, each time a timer reaches the preset value.

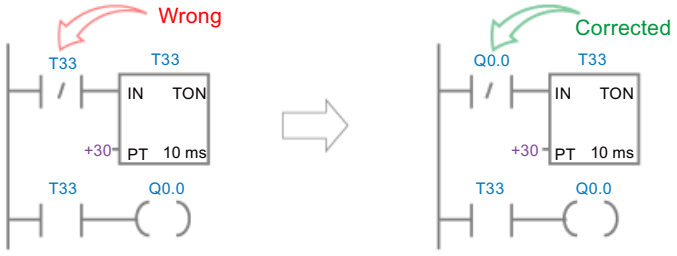
1 ms timer

Q0.0 is turned ON for one scan whenever the timer's current value is updated after the normally closed contact T32 is executed and before the normally open contact T32 is executed.



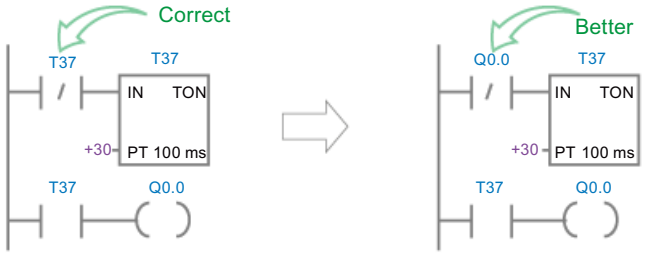
10 ms timer

Q0.0 is never turned ON, because the timer bit T33 is turned ON from the top of the scan to the point where the timer box is executed. Once the timer box has been executed, the timer's current value and its T–bit are set to zero. When the normally open contact T33 is executed, T33 is OFF and Q0.0 is turned OFF.



100 ms timer

Q0.0 is always turned ON for one scan whenever the timer's current value reaches the preset value.



See also

Configuring the retentive ranges (Page 190)

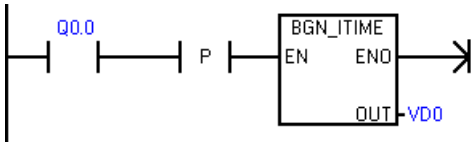
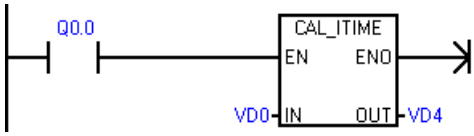
10.17.3 Interval timers

LAD / FBD	STL	Description
	BITIM OUT	The Begin interval time instruction reads the current value of the built-in 1 millisecond counter and stores the value in OUT. The maximum timed interval for a DWORD millisecond value is 2 raised to the 32 power or 49.7 days.
	CITIM IN, OUT	The Calculate interval time instruction calculates the time difference between the current time and the time provided at IN. The difference is stored in OUT. The maximum timed interval for a DWORD millisecond value is 2 raised to the 32 power or 49.7 days. CITIM automatically handles the one millisecond timer rollover that occurs within the maximum interval, depending on when the BITIM instruction was executed.

Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address	None

Input / output	Data type	Operand
IN	DWORD	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC
OUT	DWORD	VD, ID, QD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Example: Begin and Calculate interval timers

LAD		STL
 Ex1_Interval_time_net1	Capture the time that Q0.0 turned ON.	**Network 1** **LD Q0.0** **EU** **BITIM VD0**
	Calculate the time Q0.0 has been ON.	**Network 2** **LD Q0.0** **CITIM VD0, VD4**

10.18 Subroutine

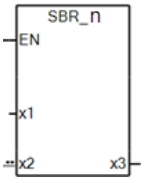
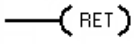
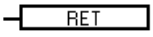
10.18.1 CALL (subroutine) and RET (conditional return)

To add a new subroutine, select the **Edit** ribbon strip then **Insert Object** and **Subroutine** command. STEP 7-Micro/WIN SMART automatically adds an unconditional return from each subroutine. You can also add conditional return CRET instructions within the subroutine.

From the main program, you can nest subroutines and FBs (place a subroutine/FB call within a subrountine/FB), with a maximum nesting depth of eight. In interrupt routines, the nesting depth for subroutines/FBs is four.

Note

Recursion (a subroutine that calls itself) is not prohibited, but you should use caution when using recursion with subroutines.

LAD / FBD	STL	Description
	**CALL SBR_n, x1, x2, x3**	The Call subroutine instruction transfers control to subroutine SBR_n. You can use a Call subroutine instruction with or without parameters. After the subroutine completes its execution, control returns to the instruction that follows the Call subroutine. The call parameters x1 (IN), x2 (IN_OUT), and x3 (OUT) represent three call parameters passed in, in and out, or out of the subroutine. The call parameters are optional. You may use from 0 to 63 call parameters. When a subroutine is called, the entire logic stack is saved, the top of stack is set to one, all other stack locations are set to zero, and control is transferred to the called subroutine. When this subroutine is completed, the stack is restored with the values saved at the point of call, and control is returned to the calling routine. Accumulators are common to subroutines and the calling routine. No save or restore operation is performed on accumulators due to subroutine use. When a subroutine is called more than once in the same cycle, the edge up, edge down, timer and counter instructions should not be used.
LAD:  FBD: 	**CRET**	The Conditional Return from Subroutine instruction (CRET) terminates the subroutine based upon the preceding logic.

Error conditions with ENO = 0	SM bits affected
• 0006H Indirect address • 008H Maximum subroutine nesting exceeded	None

Input / output	Data type	Operand
SBR_n	WORD	Constant: 0-127
IN	BOOL	V, I, Q, M, SM[2], S, T, C, L, Power Flow (LAD), Logic flow (FBD)
	BYTE	VB, IB, QB, MB, SMB[2], SB, LB, AC, *VD, *LD, *AC[1], Constant
	WORD, INT	VW, T, C, IW, QW, MW, SMW[2], SW, LW, AC, AIW, *VD, *LD, *AC[1], Constant
	DWORD, DINT	VD, ID, QD, MD, SMD[2], SD, LD, AC, HC, *VD, *LD, *AC[1], &VB, &IB, &QB, &MB, &T, &C, &SB, &AI, &AQ, &SMB, Constant
	REAL	VD, ID, QD, MD, SMD[2], SD, LD, AC, *VD, *LD, *AC1, &VB, &IB, &QB, &MB, &T, &C, &SB, &AI, &AQ, &SMB, Constant
	STRING	*VD, *LD, *AC[1], Constant
	ARRAY, UDT	
IN_OUT	BOOL	V, I, Q, M, SM[2], S, T, C, L
	BYTE	VB, IB, QB, MB, SMB[2], SB, LB, AC, *VD, *LD, *AC[1]
	WORD, INT	VW, T, C, IW, QW, MW, SMW[2], SW, LW, AC, *VD, *LD, *AC[1]
	DWORD, DINT	VD, ID, QD, MD, SMD[2], SD, LD, AC, *VD, *LD, *AC[1]
	REAL	VD, ID, QD, MD, SMD[2], SD, LD, AC, *VD, *LD, *AC1
	ARRAY, UDT	
OUT	BOOL	V, I, Q, M, SM[2], S, T, C, L
	BYTE	VB, IB, QB, MB, SMB[2], SB, LB, AC, *VD, *LD, *AC[1]
	WORD, INT	VW, T, C, IW, QW, MW, SMW[2], SW, LW, AC, AQW, *VD, *LD, *AC[1]
	DWORD, DINT	VD, ID, QD, MD, SMD[2], SD, LD, AC, *VD, *LD, *AC[1]
	REAL	VD, ID, QD, MD, SMD[2], SD, LD, AC, *VD, *LD, *AC[1]
	ARRAY, UDT	

[1] Only AC1, AC2 or AC3 (AC0 not allowed)

[2] Must be from byte offset 30 to byte offset 999 for read/write access

10.18.2 Subroutine call parameters and return examples

Calling a subroutine with call parameters

Subroutines have the option of using passed parameters. The parameters are defined in the local variable table of the subroutine. Each parameter must be assigned a local variable name (a maximum of 23 characters), a variable type, and a data type. A maximum of sixteen parameters can be passed to or from a subroutine. The VAR_Type type field in the variable table defines whether the variable is passed into the subroutine (IN), passed into and out of the subroutine (IN_OUT), or passed out of the subroutine (OUT).

To add a new parameter row, place the cursor on the Var_Type field of the type (IN, IN_OUT, OUT, or TEMP) that you want to add. Click the right mouse button to get a menu of options. Select the Insert option and then the Row Below option. Another parameter row of the selected type will appear below the current entry.

Temporary (TEMP) parameters can be assigned in the variable table to store data that is valid only within the scope of the subroutine execution. Local TEMP data is not passed as a call parameter. You can also assign TEMP parameters in the main routine and interrupt routines, but only subroutines can use IN, IN_OUT, and OUT call parameters.

Variable table parameter types for a subroutine

Parameter	Description
IN	Parameters are passed into the subroutine. If the parameter is a direct address (such as VB10), the value at the specified location is passed into the subroutine. If the parameter is an indirect address (such as *AC1), the value at the location pointed to is passed into the subroutine. If the parameter is a data constant (16#1234) or an address (&VB100), the constant or address value is passed into the subroutine.
IN_OUT	The value at the specified parameter location is passed into the subroutine, and the result value from the subroutine is returned to the same location. Constants (such as 16#1234) and addresses (such as &VB100) are not allowed for input/output parameters.
OUT	The result value from the subroutine is returned to the specified parameter location. Constants (such as 16#1234) and addresses (such as &VB100) are not allowed as output parameters. Since output parameters do not retain the value assigned by the last execution of the subroutine, you must assign values to outputs each time the subroutine is called.
TEMP	Any local memory that is not used for passed parameters can be used for temporary storage within the subroutine.

Data types allowed for call parameters

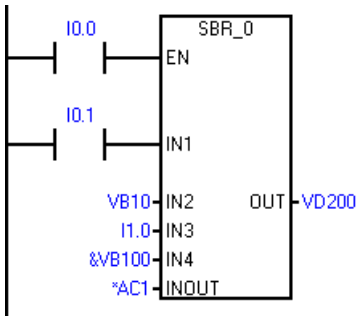
- Power Flow: Boolean power flow is allowed only for bit (Boolean) inputs. This declaration assigns an input parameter to the result of power flow based on a combination of bit logic instructions. Power flow inputs are similar to the EN input in that they connect to bit logic (for ex. LAD contacts) and not to a direct/indirect address assignment. Boolean power flow input(s) must be assigned in the top row(s) of the variable table before any non-BOOL data type assignment. Only input parameters are allowed to be used this way. The enable input (EN) and the IN1 inputs in the following example use power flow logic.
- BOOL: This data type is used for single bit inputs and outputs. IN3 in the following example is a Boolean input assigned to a direct address.
- BYTE, WORD, DWORD: These data types identify an unsigned input or output parameter of 1, 2, or 4 bytes, respectively.
- INT, DINT: These data types identify signed input or output parameters of 2 or 4 bytes, respectively.
- REAL: This data type identifies a single precision (4 byte) IEEE floating-point value.
- STRING: This data type is used as a four-byte pointer to a string.

Example variable table

Local Variable Table

	Name	Address	Var Type	Data Type	Comment
1	EN		IN	BOOL	
2	Lv1	L0.0	IN	BOOL	Tank trans...
3	HSet	L0.1	IN	BOOL	High setpoint
4	Offset	L0.2	IN	BOOL	Offset
5			IN		
6			IN_OUT		
7	Result	L0.3	OUT	BOOL	Result value
8	Valve	L0.4	OUT	BOOL	Control value
9			OUT		
10			TEMP		

Example: Subroutine call with call parameters

LAD	STL
	STL only: **Network 1** **LD I0.0** **CALL SBR_0, I0.1, VB10, I1.0, &VB100, *AC1, VD200** To display correctly in LAD and FBD: **Network 1** **LD I0.0** **= L60.0** **LD I0.1** **= L63.7** **LD L60.0** **CALL SBR_0, L63.7, VB10, I1.0, &VB100, *AC1, VD200**

Note

There are two STL examples provided above. STL programmers can use the first simplified STL instructions, which can only be displayed in the STL editor. This is because the BOOL parameters used as LAD/FBD power flow inputs are not saved to L memory.

The second set of compiler generated STL instructions can be displayed in the LAD, FBD, and STL editors, because L memory is used by the program compiler to save the state of the BOOL input parameters that are assigned as power flow inputs in LAD/FBD.

Address parameters such as IN4 (&VB100) are passed into a subroutine as a DWORD (unsigned double word) value. The type of a constant parameter must be specified for the parameter in the calling routine with a constant descriptor in front of the constant value. For example, to pass an unsigned double word constant with a value of 12,345 as a parameter, the constant parameter must be specified as DW#12345. If the constant describer is omitted from the parameter, the constant can be assumed to be a different type.

There are no automatic data type conversions performed on the input or output parameters. For example, if the variable table specifies that a parameter has the data type REAL, and in

the calling routine a double word (DWORD) is specified for that parameter, the value in the subroutine will be a double word.

When values are passed to a subroutine, they are placed into the local memory of the subroutine. The left-most column of the variable table shows the local memory address for each passed parameter. Input parameter values are copied to the subroutine's local memory when the subroutine is called. Output parameter values are copied from the subroutine's local memory to the specified output parameter addresses when the subroutine execution is complete.

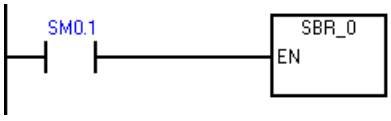
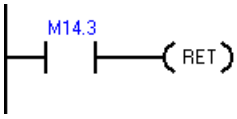
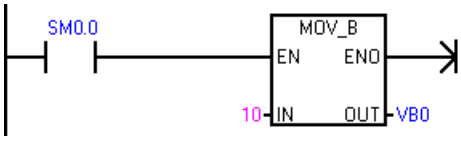
The data element size and type are represented in the coding of the parameters. Assignment of parameter values to local memory in the subroutine is as follows:

- Parameter values are assigned to local memory in the order specified by the call subroutine instruction with parameters starting at L 0.0.
- One to eight consecutive bit parameter values are assigned to a single byte starting with Lx.0 and continuing to Lx.7.
- Byte, word, and double word values are assigned to local memory on byte boundaries (LBx, LWx, or LDx).

In the Call Subroutine instruction with parameters, parameters must be arranged in order with input parameters first, followed by input/output parameters, and then followed by output parameters.

If you are programming in STL, the format of the CALL instruction is:
`CALL subroutine number, parameter 1, parameter 2, ... , parameter 16`

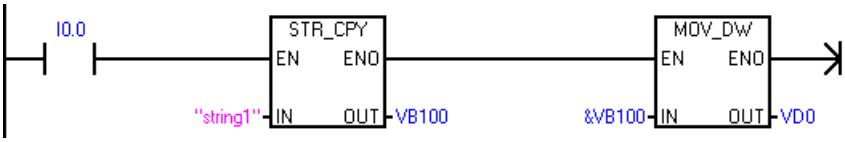
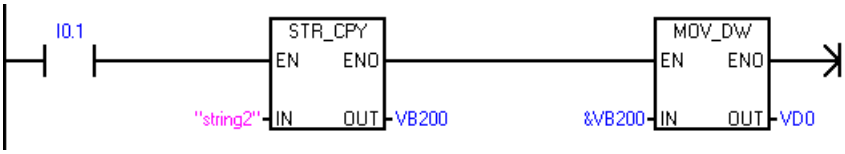
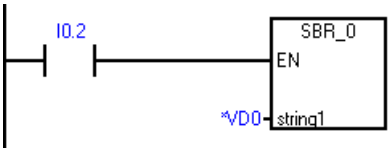
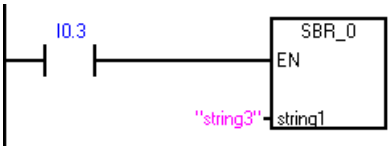
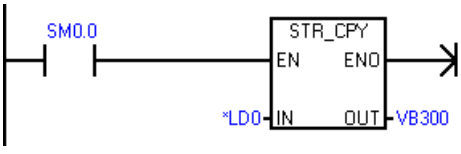
Example: Subroutine and return from subroutine instructions

LAD			STL
MAIN		On the first scan, call subroutine 0 for initialization.	**`Network 1`** **`LD SM0.1`** **`CALL SBR_0`**
SBR0		You can use a conditional return to leave the subroutine before the last network.	**`Network 1`** **`LD M14.3`** **`CRET`**
SBR0		This network will be skipped if M14.3 is ON.	**`Network 2`** **`LD SM0.0`** **`MOVB 10, VB0`**

Example: Subroutine call using string parameter

This example copies a different string literal to a unique address depending upon the given input. The unique address of this string is saved. The string address is then passed to the subroutine by using an indirect address. The data type of the subroutine input parameter is string. The subroutine then moves the string to a different location.

A string literal can also be passed to the subroutine. The string reference inside the subroutine is always the same.

LAD		STL
MAIN		**Network 1** **LD I0.0** **SCPY "string1", VB100** **AENO** **MOVD &VB100, VD0**
MAIN		**Network2** **LD I0.1** **SCPY "string2", VB200** **AENO** **MOVD &VB200, VD0**
MAIN		**Network3** **LD I0.2** **CALL SBR_0, *VD0**
MAIN		**Network4** **LD I0.3** **CALL SBR_0, "string3"**
SBR0		**Network 1** **LD SM0.0** **SSCPY *LD0, VB300**

10.19 Function block

10.19.1 CALL (function block) and RET (conditional return)

To add a new function block, select the **Edit** ribbon strip then **Insert Object** and **function block** command. STEP 7-Micro/WIN SMART automatically adds an unconditional return from each function block. You can also add conditional return CRET instructions within the function block.

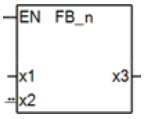
From the main program, you can nest function blocks (place a function block call within a function block) to a depth of eight.

From an interrupt routine, you can nest function blocks to a depth of four.

From the main program, you can nest subroutines and FBs (place a subroutine/FB call within a subrountine/FB), with a maximum nesting depth of eight. In interrupt routines, the nesting depth for subroutines/FBs is four.

Note

Recursion (a function block that calls itself) is prohibited.

LAD / FBD	STL	Description
	**FBCALL FB_n, FB_n_instance, FB_n_instance.x1, FB_n_instance.x2, FB_n_instance.x3**	The Call function block instruction transfers control to function block SBR_n. You can use a Call function block instruction with or without parameters. After the function block completes its execution, control returns to the instruction that follows the Call function block. The call parameters x1 (IN), x2 (IN_OUT), x3 (OUT) represent three call parameters passed in, in and out, or out of the function block. The call parameters are optional. You may use from 0 to 62 call parameters. When a function block is called, the entire logic stack is saved, the top of stack is set to one, all other stack locations are set to zero, and control is transferred to the called function block. When this function block is completed, the stack is restored with the values saved at the point of call, and control is returned to the calling routine. Accumulators are common to function blocks and the calling routine. No save or restore operation is performed on accumulators due to function block use. When a function block is called more than once in the same cycle, the edge up, edge down, timer, ALT, ALTP and counter instructions should not be used.

Error conditions with ENO = 0	SM bits affected
• 0006H Indirect address • 008H Maximum function block nesting exceeded	None

Input / output	Data type	Operand
FB_n	WORD	Constant: 0-127
IN	BOOL	V, I, Q, M, SM[2], S, T, C, L, Power Flow (LAD), Logic flow (FBD)
	BYTE	VB, IB, QB, MB, SMB[2], SB, LB, AC, *VD, *LD, *AC[1], Constant
	WORD, INT	VW, T, C, IW, QW, MW, SMW[2], SW, LW, AC, AIW, *VD, *LD, *AC[1], Constant
	DWORD, DINT	VD, ID, QD, MD, SMD[2], SD, LD, AC, HC, *VD, *LD, *AC[1], &VB, &IB, &QB, &MB, &T, &C, &SB, &AI, &AQ, &SMB, Constant
	REAL	VD, ID, QD, MD, SMD[2], SD, LD, AC, *VD, *LD, *AC[1], &VB, &IB, &QB, &MB, &T, &C, &SB, &AI, &AQ, &SMB, Constant
	STRING	*VD, *LD, *AC[1], Constant
	ARRAY, UDT	

Input / output	Data type	Operand
IN_OUT	BOOL	V, I, Q, M, SM[2], S, T, C, L
	BYTE	VB, IB, QB, MB, SMB[2], SB, LB, AC, *VD, *LD, *AC[1]
	WORD, INT	VW, T, C, IW, QW, MW, SMW[2], SW, LW, AC, *VD, *LD, *AC[1]
	DWORD, DINT	VD, ID, QD, MD, SMD[2], SD, LD, AC, *VD, *LD, *AC[1]
	REAL	VD, ID, QD, MD, SMD[2], SD, LD, AC, *VD, *LD, *AC[1]
	STRING	*VD, *LD, *AC[1]
	ARRAY, UDT	
OUT	BOOL	V, I, Q, M, SM[2], S, T, C, L
	BYTE	VB, IB, QB, MB, SMB[2], SB, LB, AC, *VD, *LD, *AC[1]
	WORD, INT	VW, T, C, IW, QW, MW, SMW[2], SW, LW, AC, AQW, *VD, *LD, *AC[1]
	DWORD, DINT	VD, ID, QD, MD, SMD[2], SD, LD, AC, *VD, *LD, *AC[1]
	REAL	VD, ID, QD, MD, SMD[2], SD, LD, AC, *VD, *LD, *AC[1]
	STRING	*VD, *LD, *AC[1]
	ARRAY, UDT	
Static	BOOL	
	BYTE	
	WORD, INT	
	DWORD, DINT	
	REAL	
	STRING	
	ARRAY, UDT	

[1] Only AC1, AC2 or AC3 (AC0 not allowed)

[2] Must be from byte offset 30 to byte offset 999 for read/write access

Note: The input or output of function block can be elements of the UDT or ARRAY.

10.19.2 Function block call parameters

Calling a function block with call parameters

Function blocks have the option of using passed parameters. The parameters are defined in the local variable table of the Function block. Each parameter must be assigned a local variable name (a maximum of 23 characters), a variable type, and a data type. A maximum of sixteen parameters can be passed to or from a Function block. The VAR_Type type field in the variable table defines whether the variable is passed into the Function block (IN), passed into and out of the Function block (IN_OUT), or passed out of the Function block (OUT).

To add a new parameter row, place the cursor on the Var_Type field of the type (IN, IN_OUT, OUT, STATIC or TEMP) that you want to add. Click the right mouse button to get a menu of options. Select the Insert option and then the Row Below option. Another parameter row of the selected type will appear below the current entry.

Temporary (TEMP) parameters can be assigned in the variable table to store data that is valid only within the scope of the Function block execution. Local TEMP data is not passed as a call

parameter. You can also assign TEMP parameters in the main routine and interrupt routines, but only Function blocks can use IN, IN_OUT, and OUT call parameters.

STATIC

Variable table parameter types for a Function block

Parameter	Description
IN	Parameters are passed into the function block. If the parameter is a direct address (such as VB10), the value at the specified location is passed into the function block. If the parameter is an indirect address (such as *AC1), the value at the location pointed to is passed into the function block. If the parameter is a data constant (16#1234) or an address (&VB100), the constant or address value is passed into the function block.
IN_OUT	The value at the specified parameter location is passed into the function block, and the result value from the Function block is returned to the same location. Constants (such as 16#1234) and addresses (such as &VB100) are not allowed for input/output parameters.
OUT	The result value from the Function block is returned to the specified parameter location. Constants (such as 16#1234) and addresses (such as &VB100) are not allowed as output parameters. Since output parameters do not retain the value assigned by the last execution of the Function block, you must assign values to outputs each time the Function block is called.
TEMP	Any local memory that is not used for passed parameters can be used for temporary storage within the function block.
STATIC	Static variables are stored in the FB instance and do not take part in external parameter passing.

Data types allowed for call parameters

- BOOL: This data type is used for single bit inputs and outputs. IN3 in the following example is a Boolean input assigned to a direct address.
- BYTE, WORD, DWORD: These data types identify an unsigned input or output parameter of 1, 2, or 4 bytes, respectively.
- INT, DINT: These data types identify signed input or output parameters of 2 or 4 bytes, respectively.
- REAL: This data type identifies a single precision (4 byte) IEEE floating-point value.
- STRING: This data type is used as a four-byte pointer to a string.
- ARRAY or UDT: The input or output of function block can be elements of the UDT or ARRAY.

Example variable table

Local Variable Table

	Variable Name	Address	Var Type	Data T...	Defalt Val...	Retain	Comment
1	SetCom		IN	BOOL	OFF	Non-retain	
2	ResetCom		IN	BOOL	OFF	Non-retain	
3	Input		IN	INT	+0	Non-retain	
4			IN_OUT	BOOL	OFF	Non-retain	
5	Output		OUT	INT	+0	Non-retain	
6	Out		OUT	BOOL	OFF	Non-retain	
7	Diff		OUT	INT	+0	Non-retain	
8	SavedInput		STATIC	INT	+0	Non-retain	
9	Resetstate		STATIC	BOOL	OFF	Non-retain	
10							
11			TEMP				

Example: function block call with call parameters

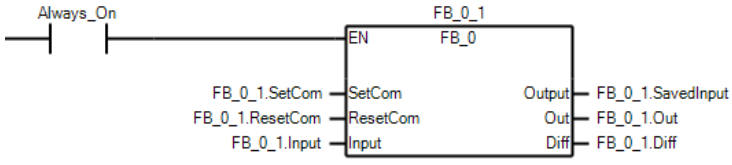
To call a function block, you must create the instance in the FB Instances table first.

Global Variable Table

	Variable Name	Data Type	Initial Value	Retain	Address	Length	Comment
1	FB_0_1	FB_0			DB2.DBB8	14 B	
2	SetCom	BOOL	OFF		DB2.DBX...		
3	ResetCom	BOOL	OFF		DB2.DBX...		
4	Input	INT	+0		DB2.DB...		
5	Output	INT	+0		DB2.DB...		
6	Out	BOOL	OFF		DB2.DBX...		
7	Diff	INT	+0		DB2.DB...		
8	SavedInput	INT	+0		DB2.DB...		
9	Resetstate	BOOL	OFF		DB2.DBX...		
10	FB_0_2	FB_0			DB2.DBB...	14 B	
11	FB_0_3	FB_0			DB2.DBB...	14 B	

Variables | System Defined Variable Table | FB Instance Table | System Motion Variable Table

LAD



STL

```
LD      Always_On
FBCALL FB_0, FB_0_1, FB_0_1.SetCom, FB_0_1.ResetCom, FB_0_1.Input, FB_0_1.SavedInput,
FB_0_1.Out, FB_0_1.Diff
```

Address parameters such as IN4 (&VB100) are passed into a function block as a DWORD (unsigned double word) value. The type of a constant parameter must be specified for the parameter in the calling routine with a constant descriptor in front of the constant value. For example, to pass an unsigned double word constant with a value of 12,345 as a parameter, the constant parameter must be specified as DW#12345. If the constant describer is omitted from the parameter, the constant can be assumed to be a different type.

There are no automatic data type conversions performed on the input or output parameters. For example, if the variable table specifies that a parameter has the data type REAL, and in the calling routine a double word (DWORD) is specified for that parameter, the value in the function block will be a double word.

When values are passed to a function block, they are placed into the local memory of the function block. The left-most column of the variable table shows the local memory address for each passed parameter. Input parameter values are copied to the function block's local memory when the function block is called. Output parameter values are copied from the function block's local memory to the specified output parameter addresses when the function block execution is complete.

The data element size and type are represented in the coding of the parameters. Assignment of parameter values to local memory in the function block is as follows:

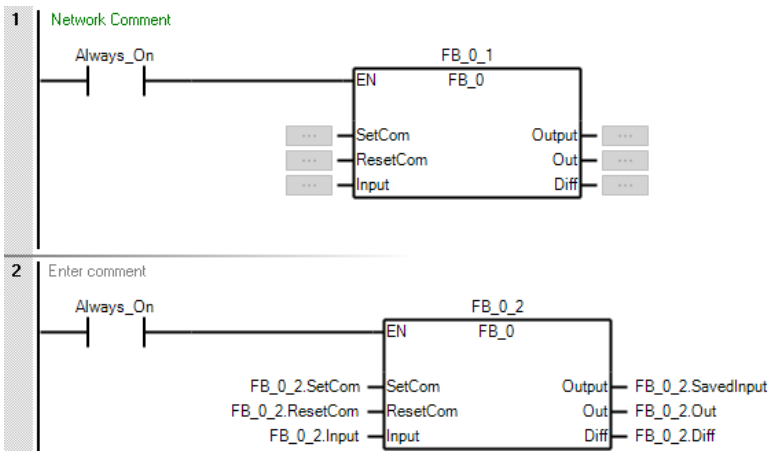
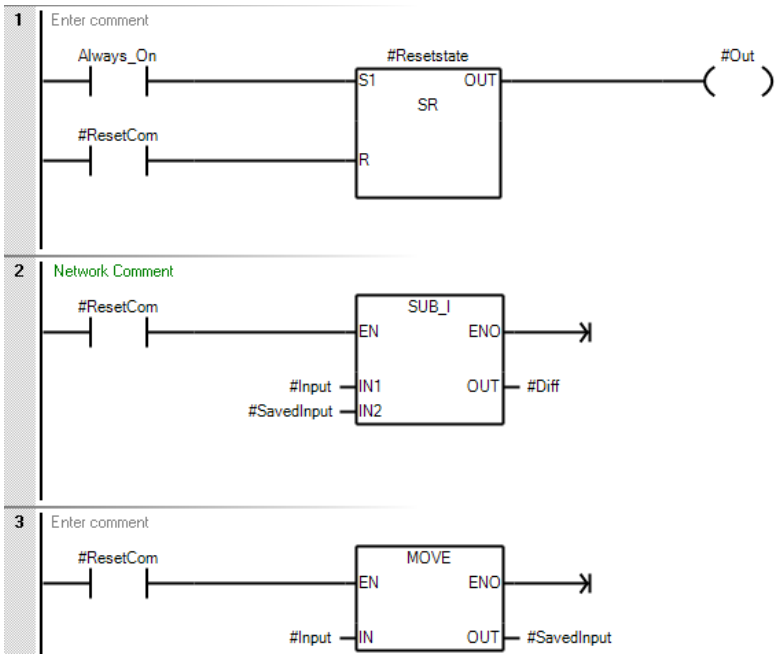
- Parameter values are assigned to local memory in the order specified by the call function block instruction with parameters starting at L 0.0.
- One to eight consecutive bit parameter values are assigned to a single byte starting with Lx.0 and continuing to Lx.7.
- Byte, word, and double word values are assigned to local memory on byte boundaries (LBx, LWx, or LDx).

In the Call function block instruction with parameters, parameters must be arranged in order with input parameters first, followed by input/output parameters, and then followed by output parameters.

If you are programming in STL, the format of the CALL instruction is:

```
CALL function block number, parameter 1, parameter 2, ... ,
parameter 16
```

Example: function block call using string parameter

LAD		STL
MAIN[1]		**Network 1** **LD Always_On** **FBCALL FB_0, FB_0_1,** **FB_0_1.SetCom,** **FB_0_1.ResetCom,** **FB_0_1.Input,** **FB_0_1.Output,** **FB_0_1.Out, FB_0_1.Diff** **Network 2** **LD Always_On** **FBCALL FB_0, FB_0_2,** **FB_0_2.SetCom,** **FB_0_2.ResetCom,** **FB_0_2.Input,** **FB_0_2.SavedInput,** **FB_0_2.Out, FB_0_2.Diff**
FB0		**Network 1** **LD Always_On** **LD #ResetCom** **NOT** **A #Resetstate** **OLD** **= #Resetstate** **= #Out** **Network 2** **LD #ResetCom** **-I #Input,** **#SavedInput, #Diff** **Network 3** **LD #ResetCom** **MOVE #Input,** **#SavedInput**

[1] When you call a function block (FB) instance, if you don't assign variables to the pins, the system automatically assigns variables to the pins.

10.20 PROFINET

10.20.1 Features of the programming instruction "PROFINET"

There are two groups of programming instructions under the folder "PROFINET" :

- RDREC and WRREC (Page 470): Read a data record from any connected PROFINET device or write a data record to any connected PROFINET device.
- BLKMOV_BIR and BLKMOV_BIW (Page 474): Read multiple bytes immediately from PROFINET device or write multiple bytes immediately to PROFINET device.

Note

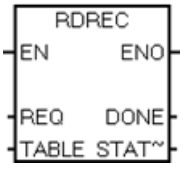
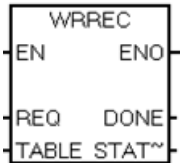
For any legacy instruction that can access the I or Q memory area, it accesses to the I or Q memory area of a PROFINET network.

10.20.2 Read and Write data record

10.20.2.1 Input and output interface of RDREC and WRREC instruction

The RDREC and WRREC instructions are as follows:

Table 10-21 RDREC and WRREC

LAD/FBD	STL	Description
	RDREC Req, Table, Done, Status	Use the RDREC instruction to read a data record from PROFINET device.
	WRREC Req, Table, Done, Status	Use the WRREC instruction to write a data record to PROFINET device.

The parameters of the RDREC and WRREC instructions are as follows:

Table 10-22 Parameters of RDREC and WRREC instruction

Parameter and type		Data type	Operand	Description
REQ	IN	BOOL	I, Q, V, M, T, C, SM, S, L	REQ=1: Transfer data record
TABLE	IN/OUT	BYTE	Q, V, M, SM, S, L, *AC,*VD,*LD	A table of parameters you set for the data read/ write record. For detailed information, refer to Definition of "TABLE" parameters (Page 471).
DONE	OUT	BOOL	I, Q, V, M, SM, S, L	The instruction is completed.
STATUS	OUT	BYTE	I, Q, V, M, SM, S, L	The status of the current operation. For detailed information, refer to Definition of "STATUS" parameters (Page 472).

Note

The supported maximum data record length is 1024 bytes.

Definition of "TABLE" parameters

The following table lists the parameter information of "TABLE":

Table 10-23 TABLE

Byte Offset	Parameter and Type		Comment
0	Device Number	Input **Note:** The value range is from 1 to 8.	Device Number, API Number, Slot Number and SubSlot Number are used to address a submodule. You can find the Device Number, API Number, Slot Number and SubSlot Number in the PROFINET wizard.
1			
2	API Number	Input	
3			
4			
5			
6	Slot Number	Input	
7			
8	SubSlot Number	Input	
9			
10	Record Index	Input	The Record Index includes the record index from protocol or the user-defined record index. For detailed information of the index from the protocol, refer to *Technical Specification for PROFINET IO (Version 2.3)*.
11			
12	Buffer Length	Input	This parameter refers to the number of bytes of the buffer. The buffer stores the data record read from or written to the device. The value range: from 1 to 1024.
13			

Byte Offset	Parameter and Type		Comment
14	Data Address	Input	Address of the buffer read from or written to the device. **Note:** If the buffer length is greater than the actual record data length, the buffer contains all the record data. If the buffer length is smaller than actual record data length, the buffer contains partial record data and an error occurs.
15			
16			
17			
18	Actual record data length	Output	This parameter is valid for the instruction RDREC and returns the actual data length specified by the device.
19			
20	PROFINET Error Code	Output	The error code defined by the PROFINET protocol. 0 = no error If the value is not 0, check the specific error code in *Technical Specification for PROFINET IO (Version 2.3)*.
21			
22			
23			

Definition of "STATUS" parameters

The following table lists the parameter information of "STATUS":

Table 10-24 STATUS

Byte Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	A [1]	E [2]	Error code [3]					

[1] A : 1 = a request is in process

[2] E : 1= an error occurs

[3] Error code: The system error code. For detailed information, refer to System-defined error code of the instructions RDREC and WRREC (Page 472).

10.20.2.2 System-defined error code of the instructions RDREC and WRREC

The error codes are as follows:

Error code	Description
0	No error.
1	The data length parameter is 0 or is greater than the supported maximum length (1024 bytes).
2	The data buffer is not in I, Q, M, or V memory areas.
3	The data buffer does not fit in the memory area.
4	The table doesn't match with the memory.
5	The device number is invalid and not within the range: from 1 to 8.

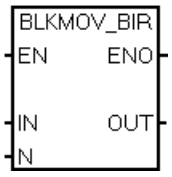
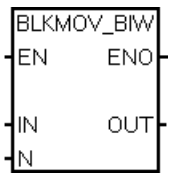
Error code	Description
6	An instance mismatch: The connection is busy with another instance, whose device number, API number, slot number and subslot number are same as the requested instance, but with a different buffer size and data address.
7	The PROFINET device is not connected.
8	The size of the received buffer exceeds 1024 bytes.
9	Call sequence is invalid.
10	Parameters are invalid (for example, out-of-range).
11	The AR is created afresh in the meanwhile.
12	The RPC reports a timeout error.
13	The RPC reports a communication error.
14	The RPC Server of the IOD signaled “busy” (for example, the call can be repeated later).
15	CLRPC reports an error or the PDU cannot be parsed.
16	CM response is OK, but has a PROFINET protocol defined error.
17	The instruction parameter is invalid.
24	REQ is not enabled.
25	The buffer length is smaller than the actual data record length.
63	Unknown error.

10.20.3 Read and Write multiple bytes between physical PROFINET and memory address

10.20.3.1 Input and output interface of BLKMOV_BIR and BLKMOV_BIW

The BLKMOV_BIR and BLKMOV_BIW instruction is as follows:

Table 10-25 BLKMOV_BIR **and** BLKMOV_BIW

LAD/FBD	STL	Description
	BMIR In, Out, N	The "BLKMOV_BIR Immediate Read" instruction reads multiple bytes of physical PROFINET input IN and writes the result to the memory address OUT, but the process image register is not updated. The input signal N defines the number of bytes. **Note**: • N ≤ 128 • N bytes cannot exceed the submodule boundary.
	BMIW In, Out, N	The "BLKMOV_BIW Immediate Write" instruction reads multiple bytes from the memory address IN and writes to physical PROFINET output OUT, and the corresponding process image location is updated. The input signal N defines the number of bytes. **Note**: • N ≤ 128 • N bytes cannot exceed the submodule boundary.

The parameters of BLKMOV_BIR and BLKMOV_BIW are as follows:

Parameter and type		Data type	Operand
IN (BLKMOV_BIR)	IN	BYTE	IB, *VD, *LD, *AC
IN (BLKMOV_BIW)	IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC
OUT (BLKMOV_BIR)	OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC
OUT (BLKMOV_BIW)	OUT	BYTE	QB, *VD, *LD, *AC
N	IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, Constant, *VD, *LD, *AC

10.20.3.2 Error code of the instructions BLKMOV_BIR and BLKMOV_BIW

Non-fatal errors with ENO = 0	SM bits affected
● 0006H Indirect address ● Unable to access expansion module	None

Communication 11

The S7-200 SMART offers several types of communication between CPUs, programming devices, and HMIs:

- Ethernet:
  - Exchange of data from the programming device to the CPU
  - Exchange of data between HMIs and the CPU
  - S7 peer-to-peer communication with other S7-200 SMART CPUs
  - Open User Communication (OUC) with other Ethernet-capable devices
  - PROFINET communication with PROFINET devices
- PROFIBUS:
  - High speed communications for distributed I/O (up to 12 Mbps)
  - One bus master connects to many I/O devices (supports 126 addressable devices).
  - Exchange of data between the master and I/O devices
  - EM DP01 module is a PROFIBUS I/O device.
- RS485:
  - Provides a STEP 7-Micro/WIN SMART V3 connection for programming when using a USB-PPI cable
  - Supports a total of 126 addressable devices (32 devices per network segment)
  - Supports PPI (point-to-point interface) protocol
  - Exchange of data between HMIs and the CPU
  - Exchange of data between devices and the CPU using Freeport (XMT/RCV instructions)
- RS232:
  - Supports a point-to-point connection to one device
  - Supports PPI protocol
  - Exchange of data between HMIs and the CPU
  - Exchange of data between devices and the CPU using Freeport (XMT/RCV instructions)

11.1 CPU communication connections

The CPU supports the following maximum number of simultaneous, asynchronous communication connections:

- Ethernet programming port:
  - Open User Communication (OUC) connections: Eight active (client) connections and eight passive (server) connections to support S7-200 SMART V3.0 CPUs or other Ethernet devices.
  - HMI/OPC connections: Eight dedicated HMI/OPC server connections.
  - PG connections: One programming device (PG) connection.
  - Peer-to-peer (GET/PUT) connections: Eight active (client) connections and eight passive (server) connections to support S7-200 SMART V3.0 CPUs or network devices.
  - PROFINET connections: Each PROFINET controller can support eight connections (IO device or drive).

  Note

  You can program an S7-200 SMART V3.0 CPU through the Ethernet port. A single device equipped with SETP-7 Micro/WIN SMART V3 can only program or monitor one CPU at a time.

  Note

  The S7-200 SMART CPU uses the GET and PUT instructions for CPU-to-CPU communications.

- Integrated RS485 port (Port 0): Four connections to support HMI devices and one connection reserved for programming with STEP 7-Micro/WIN SMART V3.

  Note

  You can make the following RS485 communication connections:
  - Use a USB-PPI cable to program all CPU models through any serial port, including the RS485 port, the signal board port, and the DP01 PROFIBUS port.
  - Use the RS485 and RS232 ports for HMI access (Data read/write) and Freeport communications.

- PROFIBUS port: Each EM DP01 PROFIBUS DP module can support six connections.
- CM01 Signal Board (SB) RS232/RS485 port (Port 1, Port 2): Four connections to support HMI devices.

11.2 CPU communication ports

S7-200 SMART V3.0 CPU provides interfaces with the following types of communication:

- Ethernet port:
  - STEP 7-Micro/WIN SMART programming
  - GET/PUT communication
  - HMI: Ethernet type
  - Open User Communication (OUC) over UDP, TCP, or ISO-on-TCP
  - PROFINET Communication
- RS485 port (Port 0):
  - STEP 7-Micro/WIN SMART V3 programming when using a USB-PPI cable
  - TDs/HMI: RS485 type
  - Freeport (XMT/RCV) including Siemens-provided USS and Modbus RTU libraries
- PROFIBUS port: The S7-200 SMART CPUs can support two EM DP01 modules for PROFIBUS DP and HMI communication.
- RS485/RS232 signal board (SB) :
  - TDs/HMIs: RS485 or RS232 type
  - Freeport (XMT/RCV) including Siemens-provided USS (RS485 only) and Modbus RTU (RS485 or RS232) libraries

11.3 HMIs and communication drivers

HMIs

The S7-200 SMART supports the HMIs from the following Siemens HMI families:

- COMFORT HMIs (PROFINET and PROFIBUS):
  - SIMATIC HMI TP700 COMFORT
  - SIMATIC HMI TP900 COMFORT
  - SIMATIC HMI TP1200 COMFORT
  - SIMATIC HMI KP400 COMFORT
  - SIMATIC HMI KP700 COMFORT
  - SIMATIC HMI KP900 COMFORT
  - SIMATIC HMI KP1200 COMFORT
  - SIMATIC HMI KTP400 COMFORT
- SMART HMIs (Ethernet, PPI and MPI):
  - SMART 700 IE
  - SMART 1000 IE

- BASIC HMIs (Industry Ethernet Interface & RS485 Interface):
  - SIMATIC HMI KTP400 BASIC MONO PN
  - SIMATIC HMI KTP600 BASIC MONO PN
  - SIMATIC HMI KTP600 BASIC COLOR PN
  - SIMATIC HMI KTP1000 BASIC COLOR PN
  - SIMATIC HMI TP1500 BASIC COLOR PN
  - SIMATIC HMI KP300 BASIC MONO PN
- BASIC HMIs (PROFIBUS):
  - SIMATIC HMI KTP600 BASIC COLOR DP
  - SIMATIC HMI KTP1000 BASIC COLOR DP
- Micro HMIs (PROFIBUS):
  - TD 400C TEXT DISPLAY, 4 LINES

Communication drivers

Communication drivers for your S7-200 SMART HMIs can be selected in two locations:

- WinCC Flexible SMART
- TIA portal

WinCC Flexible

In the WinCC Flexible SMART package, you can select the required "Communication driver" with the following menu selections:

- Communications
- Connections table

In the "Connections table", select the "SIMATIC S7 200 SMART" driver. If the SMART driver is not in the list, select the "SIMATIC S7 200" driver.

TIA portal

In the TIA portal, you can select the required "Communication driver" with the following menu selections:

- HMI tags
- Connections

In "Connections", select the "SIMATIC S7 200" driver.

Note

If the HMI panel is using a PROFIBUS DP connection (RS485), then also set the "Network Profile" to PPI.

11.4 Ethernet

11.4.1 Overview

An Ethernet network is a differential (multi-point) network that can have up to 32 segments and 1,024 nodes. Ethernet allows for data transfer at a high speed (up to 100 Mbit/s) and long distances (Copper: Maximum approximately 1.5km; Optical: Maximum approximately 4.3km).

Possible Ethernet connections include connections for the following:

- Programming devices
- CPU-to-CPU GET/PUT communication
- HMI displays
- Open User Communication (OUC)
- PROFINET Communication
- Web server (HTTPS)

11.4.2 Local/partner connection

A Local / Partner (remote) connection defines a logical assignment of two communication partners to establish a communication connection. A connection is defined by the following:

- Communication partners involved (one active, one passive)
- Type of connection (programming device, HMI, CPU, or other device) You can enable/disable secure communication to STEP 7-Micro/WIN SMART V3 or HMI in Configuring PLC security (Page 49).
- Connection path (network, IP address, subnet mask, gateway)

The communication partners set up and establish the communication connection. The active device establishes the connection, and the passive device either accepts or rejects the connection request from the active device. After a connection is established, it is automatically maintained by the active device and monitored by both the devices.

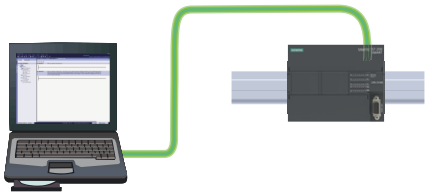
If the connection is terminated (for example, due to a line break or one of the partners breaks the connection), the active partner attempts to re-establish the connection. The passive device will also note the termination of the connection and take actions (for example, revoking the password privileges of the now disconnected active partner).

The S7-200 SMART CPUs are both active and passive devices. When an active device (for example, a computer running STEP 7-MicroWIN SMART or an HMI) establishes a connection, the CPU decides whether to accept or reject the connection request, based upon the type of the connection and how many connections of a given type are allowed.

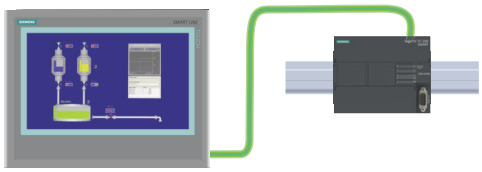
11.4.3 Sample Ethernet network configurations

You have three different types of communication options when using the S7-200 SMART V3.0 CPU Ethernet network:

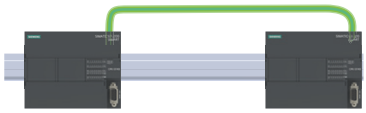
- Connecting a CPU to a programming device
- Connecting a CPU to an HMI
- Connecting a CPU to another S7-200 SMART CPU



Programming device connected to the CPU



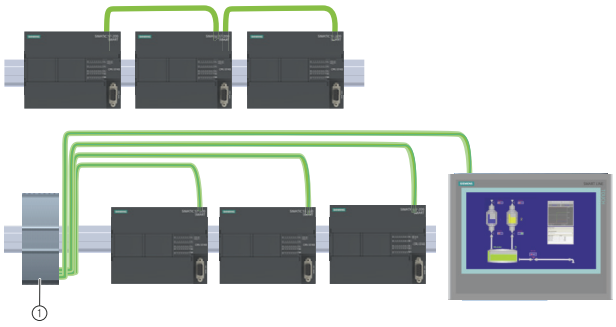
HMI connected to the CPU



A CPU connected to another CPU

Ethernet switching

S7-200 SMART V3.0 CPU has two network ports and an built-in Ethernet switch. When setting up the network, you can choose to connect devices in a linear topology or use an additional switch for a star topology.



① CSM1277 Ethernet switch

You can use the rack-mounted CSM1277 4-port Ethernet switch for connecting multiple CPUs and HMI devices.

11.4.4 Assigning Internet Protocol (IP) addresses

11.4.4.1 Assigning IP addresses to programming and network devices

If your programming device is using a network adapter card connected to your plant LAN (and possibly the World Wide Web), both the programming device and the CPU must exist on the same subnet. The subnet is specified as a combination of the IP Address and subnet mask for the device. Please see your local network administrator for help.

The Network ID is the first part of the IP address (first three octets) (for example, **211.154.184.16**) that determines what IP network you are on. The subnet mask normally has a value of **255.255.255.0**; however, since your computer is on a plant LAN, the subnet mask may have various values (for example, **255.255.254.0**) in order to set up unique subnets. The subnet mask, when combined with the device IP address in a logical AND operation, defines the boundaries of an IP subnet.

Note

In a World Wide Web scenario, where your programming devices, network devices, and IP routers will communicate with the world, unique IP addresses must be assigned to avoid conflict with other network users. Contact your company IT department personnel, who are familiar with your plant networks, for assignment of your IP addresses.

Note

A secondary network adapter card is useful when you do not want your CPU on your company LAN. During initial testing or commissioning tests, this arrangement is particularly useful.

11.4.4.2 Configuring or changing an IP address for a CPU or device in your project

You must enter the following IP information for each S7-200 SMART CPU that is attached to your Ethernet network:

- IP address: Each CPU or device must have an Internet Protocol (IP) address. The CPU or device uses this address to deliver data on a more complex, routed network. Each IP address is divided into four 8-bit segments and is expressed in a dotted, decimal format (for example, 211.154.184.16). The first part of the IP address is used for the Network ID (What network are you on?), and the second part of the address is for the Host ID (unique for each device on the network). An IP address of 192.168.x.y is a standard designation recognized as part of a private network that is not routed on the Internet.

  Note

  All S7-200 SMART CPUs have a default IP address of: 192.168.2.1

  Note

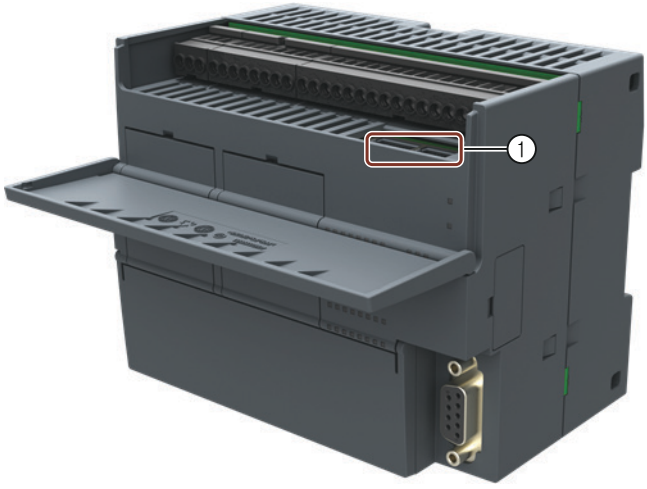
  You must have a unique IP address for each device on your network.

- Subnet mask: A subnet is a logical grouping of connected network devices. Nodes on a subnet are usually located in close physical proximity to each other on a Local Area Network (LAN). The subnet mask defines the boundaries of an IP subnet.

  Note

  A subnet mask of 255.255.255.0 is generally suitable for a local network.

- Default gateway IP address: Gateways (or IP routers) are the link between LANs. Using a gateway, a computer in a LAN can send messages to other networks, which might have other LANs behind them. If the data destination is not within the LAN, the gateway forwards the data to another network or group of networks where it can be delivered to its destination. Gateways rely on IP addresses to deliver and receive data packets.



① PROFINET (LAN) port

There are three ways to configure or change the IP information for the onboard Ethernet port of a CPU or device:

- Configuring the IP information in the "Communications" dialog (dynamic IP information)
- Configuring the IP information in the "System Block" dialog (static IP information)
- Configuring the IP information in the user program (dynamic IP information)

Note

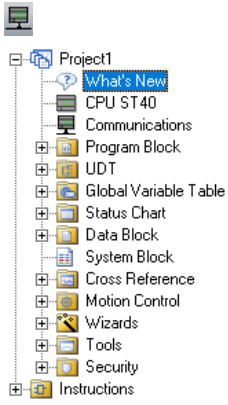
You can have static or dynamic IP information in the CPU:

- Static IP information: If the "IP address data is fixed to the values below and cannot be changed by other means" checkbox in the system block is checked, then the Ethernet network information that you enter is static. Static IP information must be downloaded to the CPU before it is active in the CPU, and, if you want to change the IP information, this IP information can only be changed in the system block dialog and once again downloaded to the CPU.
- Dynamic IP information: If the "IP address data is fixed to the values below and cannot be changed by other means" checkbox in the system block is not checked, then you change the IP address of the CPU through other means and this IP address information is considered to be dynamic. You can change the IP address information in the Communications dialog or with the SIP_ADDR instruction in the user program.
- For both static and dynamic IP, the information is stored in persistent memory.

Configuring the IP information in the Communications dialog (dynamic IP information)

IP information changes done through the Communications dialog are immediate and do not require a download of the project.

To access this dialog, perform one of the following:



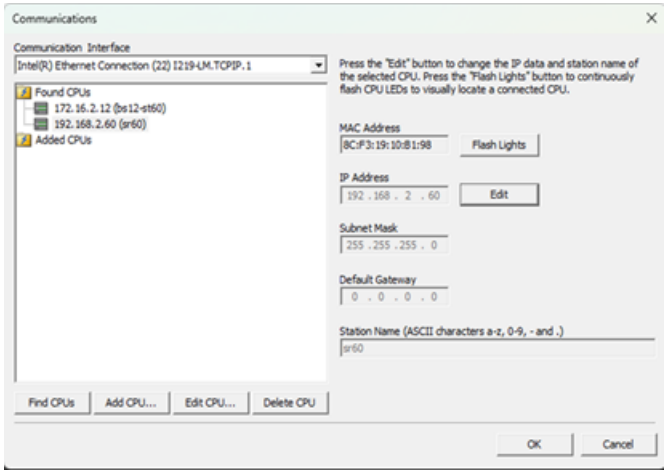
- In the Navigation bar, click the "Communications" button.
- In the Project tree, select the "Communications" node, then press Enter; or double-click the "Communications" node.

You can access CPUs in one of two ways:

- "Found CPUs": CPUs located on your local network
- "Added CPUs": CPUs on the local or remote networks (for example, CPUs accessed on another network through a router)

For "Found CPUs" (CPUs located on your local network), use the "Communications dialog" to connect with your CPU:

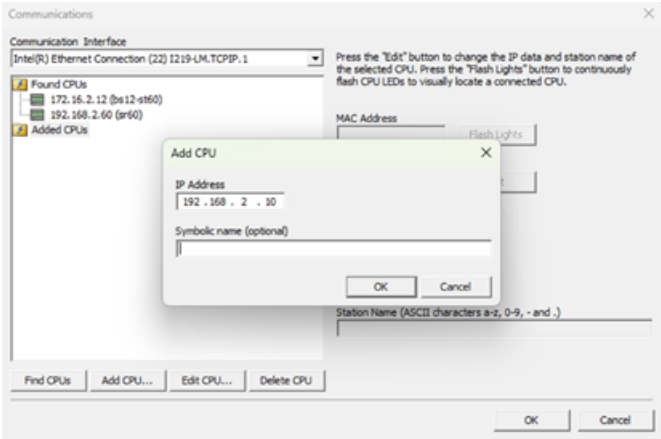
- Click the "Communication Interface" dropdown list, and select the "TCP/IP" Network Interface Card (NIC) for your programming device.
- Click the "Find CPUs" button to display all operational CPUs ("Found CPUs") on the local Ethernet network. All CPUs have a default IP address.
- Highlight a CPU, and then click "OK".



For "Added CPUs" (CPUs on the local or remote networks), use the "Communications dialog" to connect with your CPU:

- Click the "Communication Interface" dropdown list, and select the "TCP/IP" Network Interface Card (NIC) for your programming device.
- Click the "Add CPU" button to do one of the following:
  - Enter the IP address of a CPU that is accessible from the programming device, but is not on the local network.
    You can add these CPUs, select them as the communication partner in STEP 7-Micro/WIN SMART, and program and operate these CPUs in the same way you would a CPU on the local network. As long as there is a valid network path through routers, STEP 7-Micro/WIN SMART can communicate with any S7-200 SMART CPU.
  - Enter the IP address of a CPU directly that is on the local network.

  You can add multiple CPUs, on the local network and/or remote network. As always, STEP 7-Micro/WIN SMART communicates with one CPU at a time. All CPUs have a default IP address.
- Highlight a CPU, and then click "OK".



To enter or change IP information, perform the following:

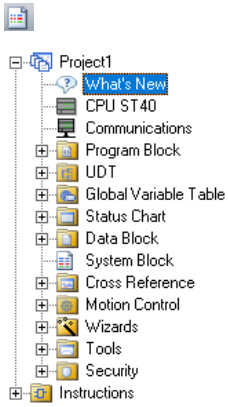
- Click the required CPU.
- If you need to identify which CPU to configure or change, click the "Flash Lights" button. This button flashes the STOP, RUN, and FAULT lights for the highlighted CPU in the list.
- Click the "Edit" button to make changes in the IP information.
- Change the following IP information:
  - IP address
  - Subnet mask
  - Default gateway
  - Station name
- Press the "Set" button. When the "Set" button is pressed, these values are updated within the CPU.
- When finished, click "OK".

When you configure IP information for the onboard Ethernet port in the Communications dialog, this information is "dynamic". If the "IP address data is fixed to the values below and cannot be changed by other means" checkbox in the system block dialog is not checked, then you must enter IP information in the Communications dialog. You can enter new IP address information and update this information in the CPU by clicking the "Set" button.

Configuring the IP information in the System Block dialog (static IP information)

IP information configuration or changes done in the system block are part of the project and do not become active until you download your project to the CPU.

To access this dialog, perform one of the following:



- In the Navigation bar, click the "System Block" button.
- In the Project tree, select the "System Block" node, then press Enter; or double-click the "System Block" node.

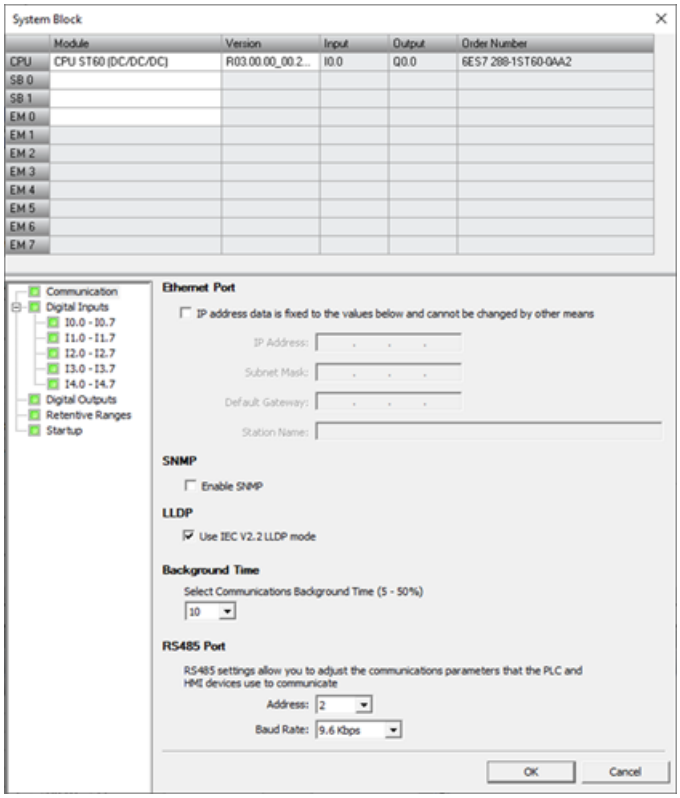
To enter or change IP information, perform the following:

- If not already checked, click the "IP address data is fixed to the values below and cannot be changed by other means" checkbox. The Ethernet port IP information fields are enabled.
- Enter or change the following IP information:
  - IP address
  - Subnet mask
  - Default gateway
  - Station name

Note

The Station Name follows the standard DNS (Domain Name System) naming conventions. The S7-200 SMART CPUs limit the Station Name to a maximum of 63 characters. The Station Name can consist of the lower case letters a through z, the digits 0 through 9, the hyphen character (minus sign), and the period character.

Certain names are not allowed and this can depend on the tool used to set the Station Name. The Station Name must not have the format "n.n.n.n" where n is a value of 0 through 999. The Station Name cannot begin with the string "port-nnn" or the string "port-nnn-nnnnn" where "n" is a digit 0 through 9 (for example, "port-123" and "port-123-45678" are illegal). The Station Name cannot start or end with the hyphen or period.



When you check the "IP address data is fixed to the values below and cannot be changed by other means" checkbox in the system block dialog, the IP information that you enter for the

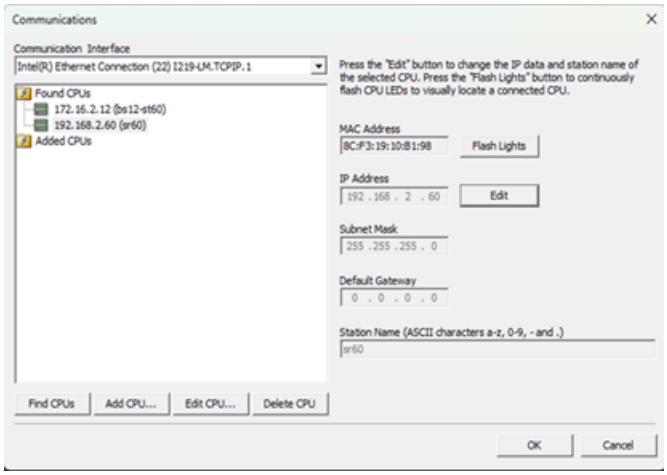
onboard Ethernet port is static. Static IP information must be downloaded to the CPU before it is active in the CPU. If you want to change the IP information, this IP information can only be changed in the system block dialog and once again downloaded to the CPU.

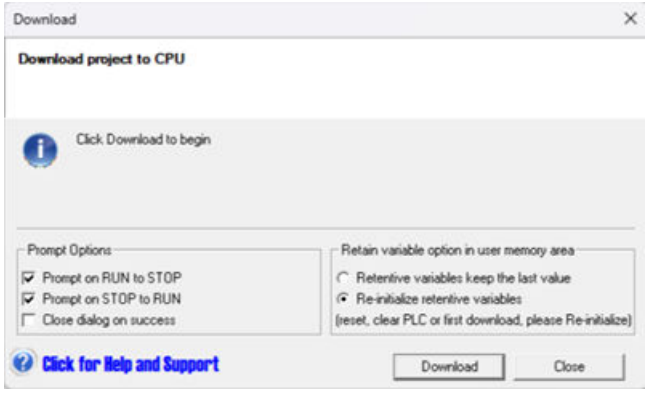
Note

If the "IP address data is fixed to the values below and cannot be changed by other means" checkbox is checked, then the IP information cannot be set in the Communications dialog.

In order to use the SIP_ADDR instruction, the "IP address data is fixed to the values below and cannot be changed by other means" checkbox must be unchecked.

After completing the IP information configuration, download the project to the CPU. All CPUs and devices that have valid IP addresses are displayed in the Communications dialog.





Configuring the IP information in the user program (dynamic IP information)

The SIP_ADDR instruction sets the CPU's IP address to the value found in its "ADDR" input, the CPU's subnet mask to the value found in its "MASK" input, and the CPU's gateway to the value found in its "GATE" input.

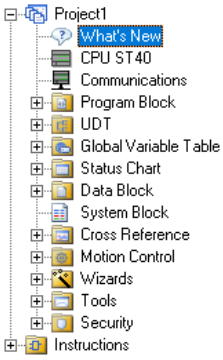
IP information or changes done through the SIP_ADDR instruction are immediate and do not require a download of the project. The IP address information set with the SIP_ADDR instruction is stored in permanent memory in the CPU.

Refer to the "Get IP address and set IP address (Ethernet)" (Page 283) instructions for more information.

11.4.4.3 Searching for CPUs and devices on your Ethernet network

You can search for and identify the S7-200 SMART CPUs that are attached to your Ethernet network in the "Communications" dialog. To access this dialog, click one of the following:

Communications button in the navigation bar



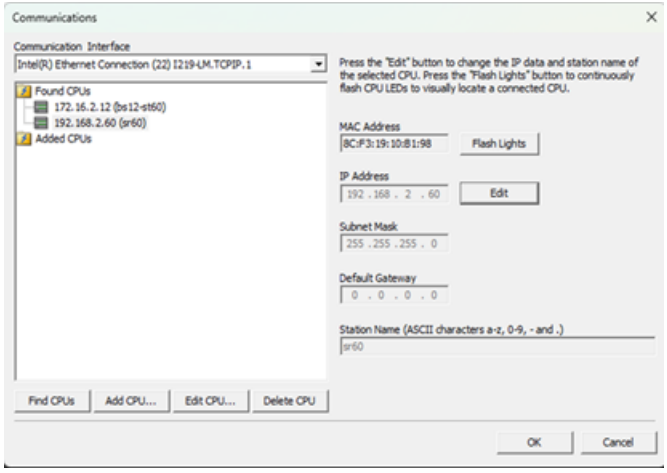
Communications in the project tree

Communications from the Component drop-down list in the Windows area of the View menu ribbon strip

The "Communications" dialog will autodetect all connected and available S7-200 SMART CPUs on a given Ethernet network by creating a lifelist. (See the figure below.) After selecting a CPU, the following detailed information about the CPU is listed:

- MAC address
- IP information
- Station name

The IP address of a CPU is not associated with a STEP 7-Micro/WIN SMART project. Opening a STEP 7-Micro/WIN SMART project does not automatically select an IP address or establish a connection to a CPU. Every time you create a new or open an existing STEP 7-Micro/WIN SMART project, you must go to the Communications dialog to establish a connection to a CPU. The Communications dialog will show the last selected CPU.



11.4.5 Locating the Ethernet (MAC) address on the CPU

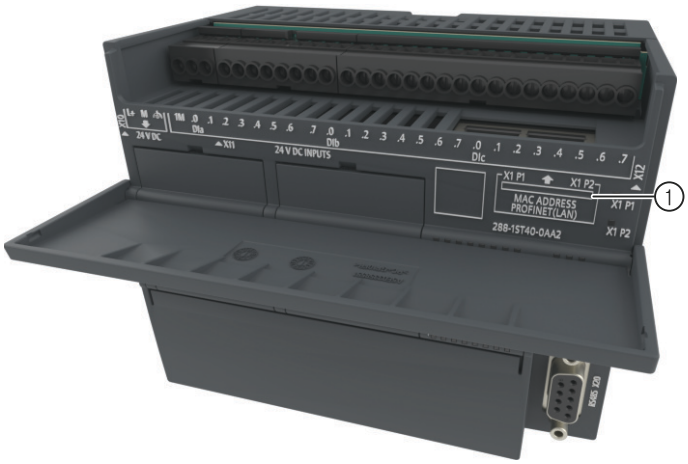
In Ethernet networking, a Media Access Control address (MAC address) is an identifier assigned to the network interface by the manufacturer for identification. A MAC address usually encodes the manufacturer's registered identification number.

The standard (IEEE 802.3) format for printing MAC addresses in human-friendly form is six groups of two hexadecimal digits separated by hyphens (-) or colons (:), for example, 01-23-45-67-89-ab or 01:23:45:67:89:ab.

Note

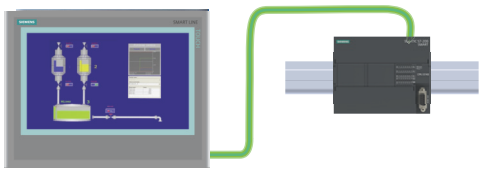
Each CPU is loaded at the factory with two permanent, unique MAC address. You cannot change the MAC address of a CPU.

The MAC address is printed on the front, upper-right corner of the CPU. Note that you have to open the upper door to see the MAC address information.



① MAC address

11.4.6 HMI-to-CPU communication



The CPU supports Ethernet communication connections to HMIs.

The following requirements must be considered when setting up communications between CPUs and HMIs:

- Configuration/Setup:
  - The CPU must be configured with an IP address.
  - The HMI must be setup and configured to connect with the IP address of the CPU.

Note

The rack-mounted CSM1277 4-port Ethernet switch can be used to connect your CPUs and HMI devices.

- Supported functions:
  - The HMI can read/write data to the CPU.
  - Messages can be triggered, based upon information retrieved from the CPU.
  - System diagnostics

To ensure that your CPU and HMI are communicating properly, follow the sequence of steps in the table below:

Table 11-1 Required steps in configuring communications between an HMI and a CPU

Step	Task
1	Establishing the hardware communications connection An Ethernet interface establishes the physical connection between an HMI and a CPU. Since Auto-Cross-Over functionality is built into the CPU, you can use either a standard or crossover Ethernet cable for the interface. Refer to "Establishing the hardware communications connection" (Page 71) for more information.
2	If you have already created a project with a CPU, open your project in STEP 7-Micro/WIN SMART. If not, create a project and insert a CPU In the project.
3	Configuring an IP address in your project Use the same configuration process; however, you must configure IP addresses for the HMI and the CPU. You must download the configuration for each CPU and HMI device. Refer to "Configuring or changing an IP address for a CPU or device in your project" (Page 283) for more information.

Note

You can restrict communication writes to a specific range of V memory; this can affect HMI communications. See "Configuring system security" (Page 49) for more information.

See also

Configuring PLC security (Page 49)

11.4.7 Open user communication

11.4.7.1 Protocols

Open User Communication (OUC) provides a mechanism for your program to transmit and receive messages over an Ethernet network. You can select the Ethernet protocol used as the transport mechanism: UDP, TCP, or ISO-on-TCP

User Datagram Protocol (UDP) uses a simple connectionless transmission model with a minimum of protocol overhead. There is no handshake mechanism in UDP protocol so the protocol is only as reliable as the underlying network. There is no guarantee of delivery, ordering, or duplicate message protection. UDP does provide checksums for data integrity and generally uses different port numbers to address different functions.

Transmission Control Protocol (TCP) is a core internet protocol. TCP provides a reliable, ordered, and error-checked delivery of messages between applications running on hosts communicating over an Ethernet network. TCP guarantees that all bytes received are identical with the bytes sent, and in the original order. TCP protocol creates a connection between an active device (the device that initiates the connection) and a passive device (the device that accepts the connection). Once a connection is established, either device can initiate a transfer of data.

TCP protocol is a "streaming" protocol. This means that there is no end sign included in a message. All the messages that are received are considered to be part of a "stream" of data. For example, the client device sends three messages of 20 bytes each to a server. The server sees only a "stream" of 60 bytes received (assuming the server executes a receive operation after the three messages are received.)

ISO-on-TCP is a protocol add-on using RFC 1006. The main advantage of ISO-on-TCP is that you have a clear data end sign in the data so that you know when the entire message has been received. SPS7 protocol (Put/Get) utilizes ISO-on-TCP protocol. ISO-on-TCP only uses port 102 and utilizes TSAPs (Transport Services Access Point) to route the message to the proper receiver instead of a port as in TCP.

The ISO-on-TCP protocol delineates each message received. For example, a client sends three messages to a server using the ISO-on-TCP protocol. Even if the server waits for all messages to accumulate before checking for a received message, the server will get each message as it was sent and see three distinct messages. This is the difference between TCP protocol and the ISO-on-TCP protocol.

11.4.7.2 Connections

The S7-200 SMART CPU has two OUC instructions to perform connection management:

- The TCON instruction to establish an active connection (client) or open a passive connection (server)
- The TDCON instruction to force a disconnection (for example, close a connection). A RUN-to-STOP transition forces the closure of all open connections by the CPU.

The CPU supports two types of connections for OUC:

- Active: A connection established and maintained by the local CPU. The local CPU is responsible for issuing the connection request to another device and maintaining the connection so that the connection does not time out due to inactivity.
- Passive: A passive connection is one in which the local CPU opens a port and/or TSAP to accept the connection request from another device.

The CPU supports eight active and eight passive connections.

The CPU creates a passive or active connection based upon the connection table passed to the TCON instruction. UDP connections are always passive. TCP and ISO-on-TCP protocol connections use a configuration parameter to determine the type of connection.

11.4.7.3 Ports and TSAPs

Ports and Transport Service Access Points (TSAPs) provide for the routing of messages to the proper receiver within the CPU or other device.

The UDP and TCP protocols allow you to select the local and remote port numbers. The port number is fixed at port 102 when you select the ISO-on-TCP protocol.

Port numbers must be in the range of 1 to 49151. It is recommended that port numbers be in the range of 2000 to 5000. The S7-200 SMART CPU range and restriction rules for port numbers are shown in the tables below:

Port number	Description
1 to 1999	• You can use these numbers; however, they are outside the recommended range. • Some ports are excluded (see below).
2000 to 5000	Recommended range
5001 to 49151	• You can use these numbers; however, they are outside the recommended range. • Some ports are excluded (see below).
49152 to 65535	• These are dynamic or private ports. • The use of these port numbers is restricted.

You cannot use the port numbers shown in the following table for local port numbers in the S7-200 SMART CPU. You have no restrictions for the remote port number:

Port number	Description
20	FTP data transmission
21	FTP control
25	SMTP
80	Web server
102	ISO-on-TCP
135	DCE for PROFINET
161	SNMP
162	SNMP Trap
443	HTTPS
34962 to 34964	PROFINET

You can have multiple active connections use the same port number for the local and remote port numbers. For example, a TCP client can connect to multiple servers on port 2500. Both the local and remote ports would normally be port 2500 for the active connection.

You cannot have multiple passive connections use the same port number as the local port number. For example, the CPU does not allow multiple TCP servers (multiple passive connections) to be on local port 2500. The CPU does not know to which of the multiple 2500 ports to route the messages.

Using Transport Service Access Points (TSAPs), ISO-on-TCP protocol allows multiple connections to a single IP address. TSAPs uniquely identify these communication end point connections to an IP address.

The ISO-on-TCP protocol utilizes port 102 exclusively. You cannot set the port number for this protocol; however, you can set the TSAP for the local and remote partner.

The TSAP rules are shown below:

- You always enter TSAPs as an S7-200 SMART string data type (a length byte followed by the string characters).
- TSAPs must be at least 2 characters and no more than 16 ASCII characters in length.

- The local TSAP cannot start with string "SIMATIC-"
- The local TSAP must begin with the hexadecimal character "0xE0" if the TSAP is exactly 2 characters long. For example, the TSAP "$E0$01" is legal but the TSAP "$01$01" is not legal. (The "$" character denotes that the following value is a hexadecimal character.)

11.4.8 PROFINET Communication

11.4.8.1 Introduction

What is PROFINET IO?

PROFINET IO is the Ethernet-based automation standard of PROFIBUS International. It defines a cross-vendor communication, automation, and engineering model.

With PROFINET IO, a switching technology allows all stations to access the network at any time. In this way, the network can be used much more efficiently through the simultaneous data transfer of several nodes. Simultaneous sending and receiving is enabled through the full-duplex operation of Switched Ethernet, with a bandwidth of 100 Mbps.

A PROFINET IO system consists of the following devices:

- PROFINET Controller: Device that addresses the connected IO devices and exchanges input and output signals with field devices.
- PROFINET IO Device: A field device that is monitored and controlled by a PROFINET Controller. A PROFINET IO Device consists of several modules and submodules.

Note

You can have a maximum of 8 PROFINET IO devices and 64 modules on your S7-200 SMART PROFINET network.

Objectives of PROFINET

The objectives of PROFINET are:

- Industrial networking, based on Industrial Ethernet (open Ethernet standard)
- Compatibility of Industrial Ethernet and standard Ethernet components
- High robustness due to Industrial Ethernet devices. Industrial Ethernet devices are suited to the industrial environment (such as temperature and noise immunity).
- Real-time capability
- Seamless integration of other fieldbus systems

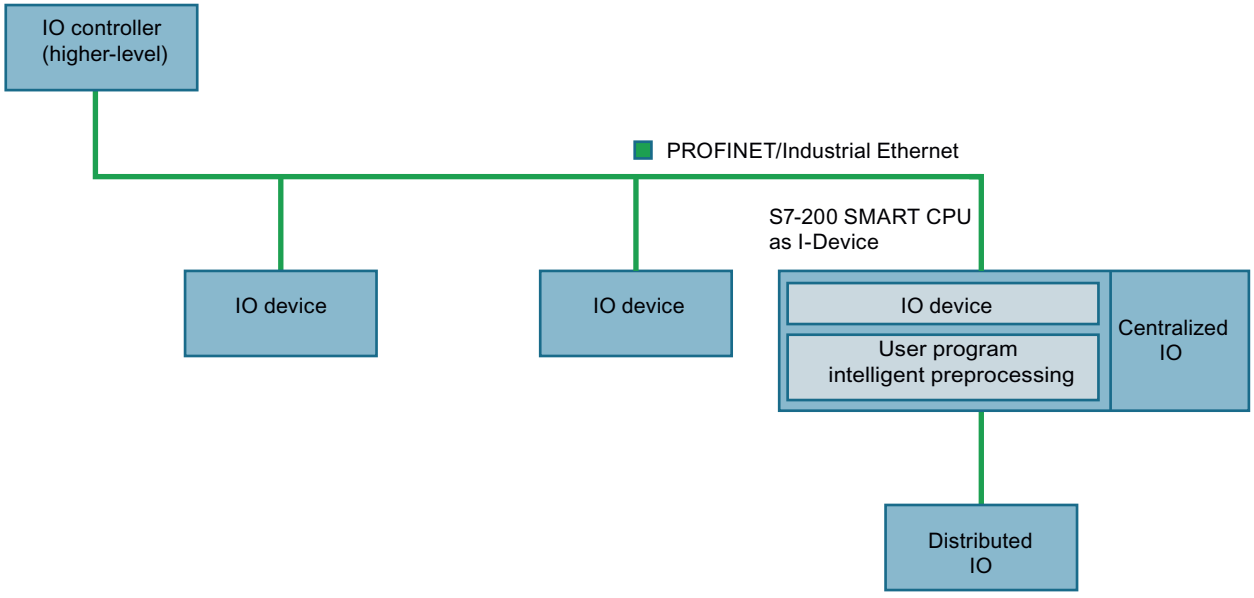
11.4.8.2 I-Device

I-Device functionality

I-Device is a CPU with the "Intelligent IO device" configuration.

The I-Device functionality of a CPU enables data exchange between a CPU and an IO controller, and CPU can work as intelligent preprocessing unit of sub processes.

The preprocessing is handled by the user program on the CPU. The process values acquired in the centralized or distributed (PROFINET IO) I/O are preprocessed by the user program and made available through a PROFINET IO interface to the higher-level IO controller.

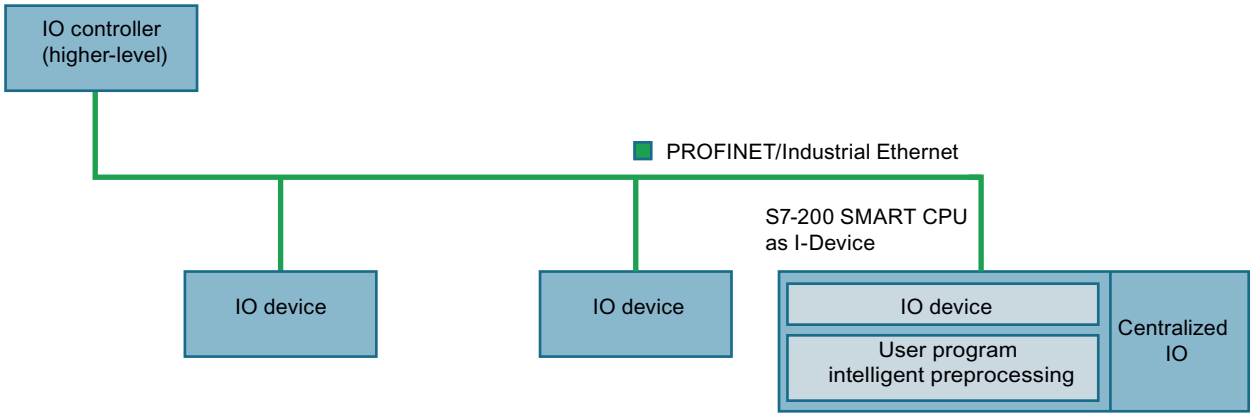


Two types of I-Device are supported for S7-200 SMART V3.0 or later version:

- I-Device without lower-level PROFINET IO system
- I-Device with lower-level PROFINET IO system

I-Device without lower-level PROFINET IO system

The I-Device does not have its own distributed IO. The configuration and parameter assignment of I-Device in the role of an IO device is the same as for a distributed I/O system.



I-Device with lower-level PROFINET IO system

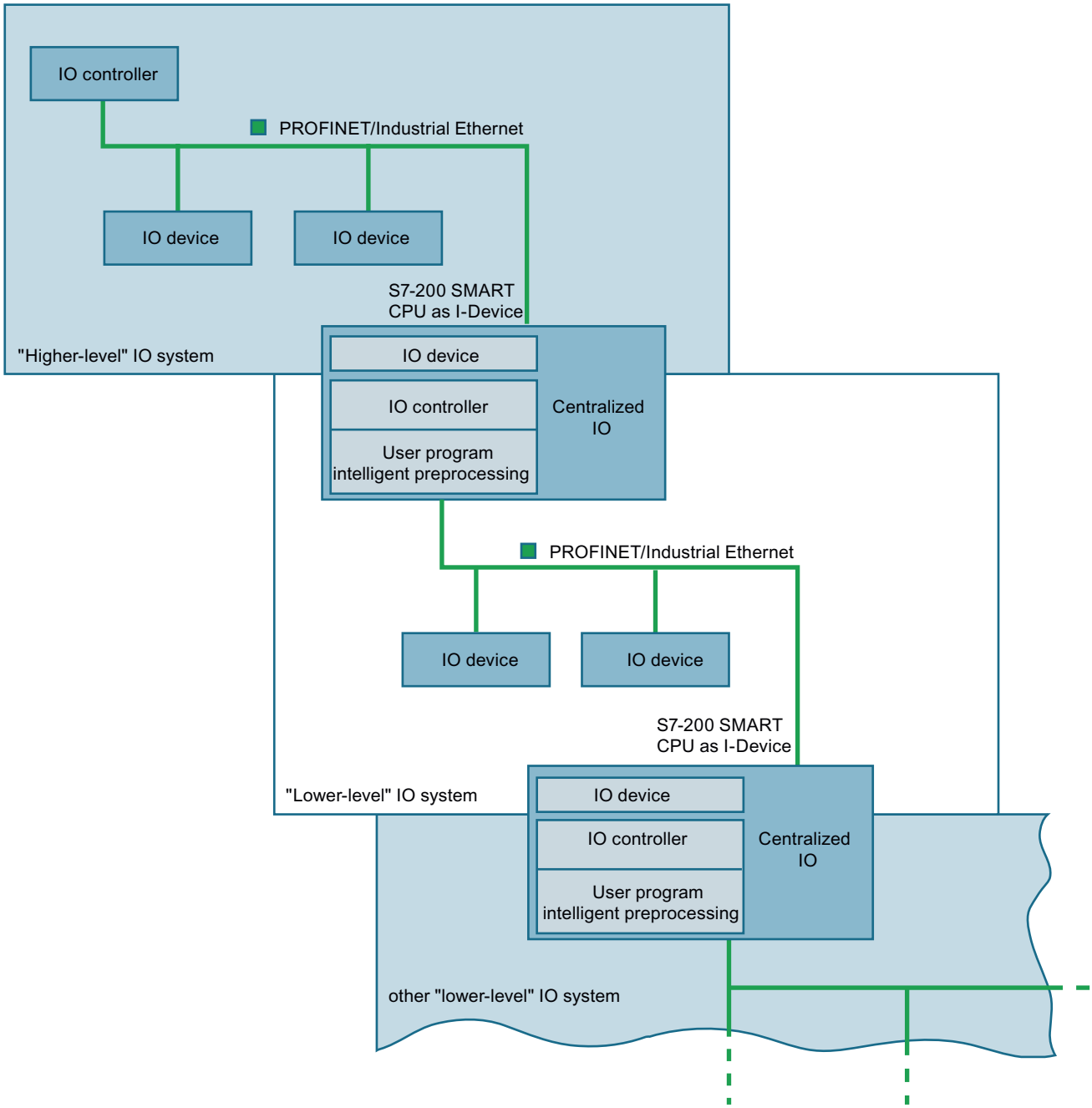
In addition to the role as an IO device, an I-Device can also work as an IO controller on a PROFINET interface.

This means that the I-Device can be part of a higher-level IO system through its PROFINET interface and work as an IO controller that supports its own lower-level IO system.

The lower-level IO system can contain I-Devices. This makes hierarchically structured IO systems possible.

Note

If S7-200 SMART CPU works as I-Device with lower-level PROFINET IO system, the maximum number of supported lower-level IO devices is eight, and only one higher-level controller is supported.



11.4.8.3 Device database file in XML: GSDML

The properties of PROFINET devices are stored in an XML file whose structure and rules are determined by a GSDML schema.

The language used to describe the GSD files is GSDML (Generic Station Description Markup Language). You can get the GSDML schemas (as schema files) from PROFINET device manufacturers.

STEP 7-Micro/WIN SMART V3 supports GSDML Specification 2.44 or earlier versions. When you import a GSDML file, STEP 7-Micro/WIN SMART V3 validates the GSDML file. If the objects pass validation, STEP 7-Micro/WIN SMART V3 puts them into the PROFINET device catalog tree (Page 513).

Before adding an IO device to the PROFINET network, you can import its GSDML files (Page 499) to STEP 7-Micro/WIN SMART V3.

11.4.8.4 GSDML Management

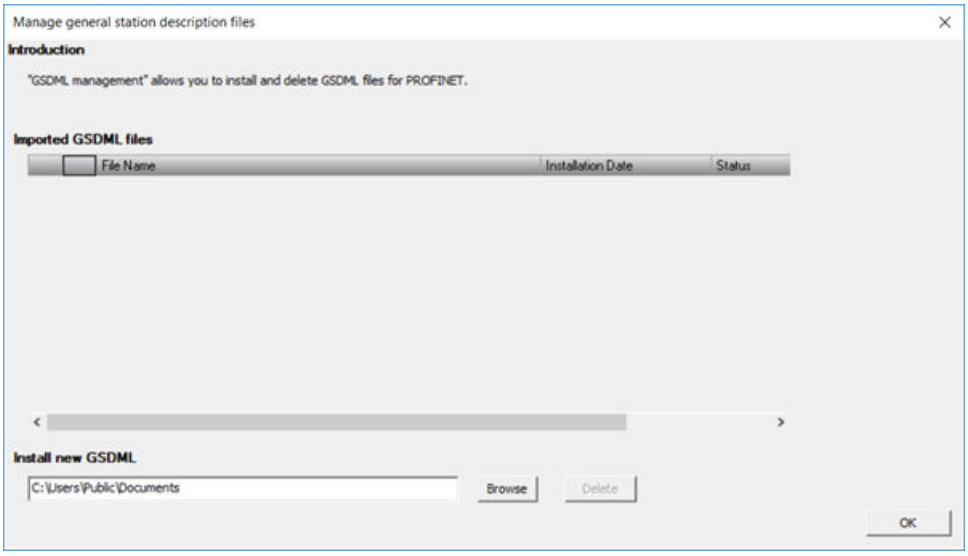
"GSDML Management" allows you to import or delete GSDML files. GSDML files for PROFINET describe the features of PROFINET device and related modules.

Importing GSDML files

To import GSDML files, follow these steps:

1. Click the "GSDML Management" button from the GSDML section of the File menu ribbon strip.

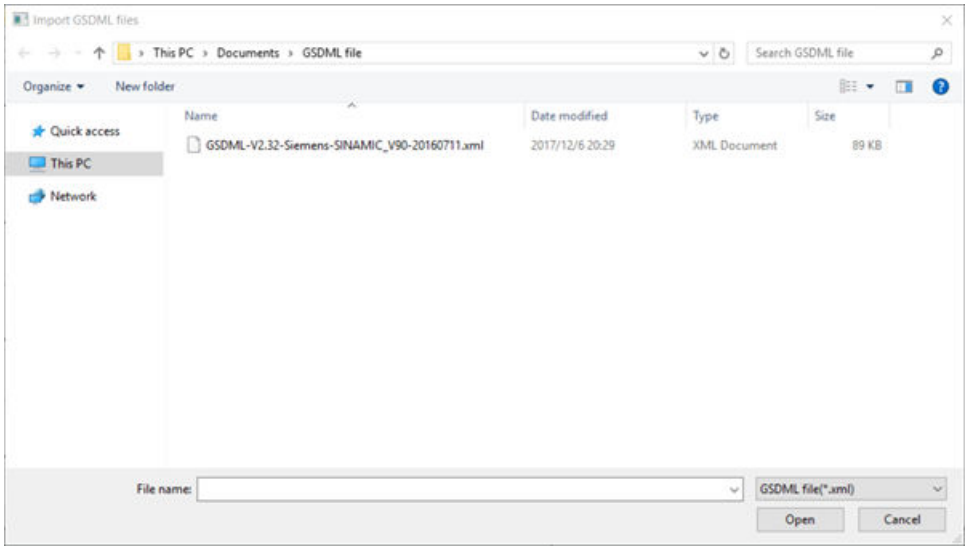
2. Click the "Browse" button in the "Manage general station description files" dialog.



Note

The next time you import GSDML files, the dialog displays the file path you navigate to last time.

3. Navigate to the folder where you save GSDML files.



4. Select the GSDML file you want to import. You can also import multiple GSDML files.

Note

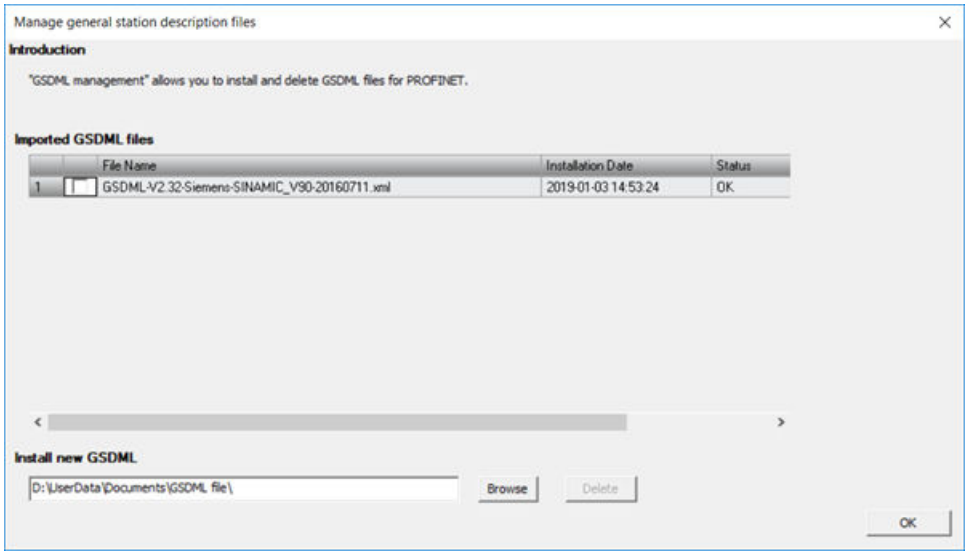
The GSDML file must conform to the GSDML Specification.

For any issue about GSDML files, contact the manufacture.

Note

The maximum number of imported GSDML file is 20.

5. Click the "Open" button. The GSDML file and the installation date displays in the "Imported GSDML files" field.

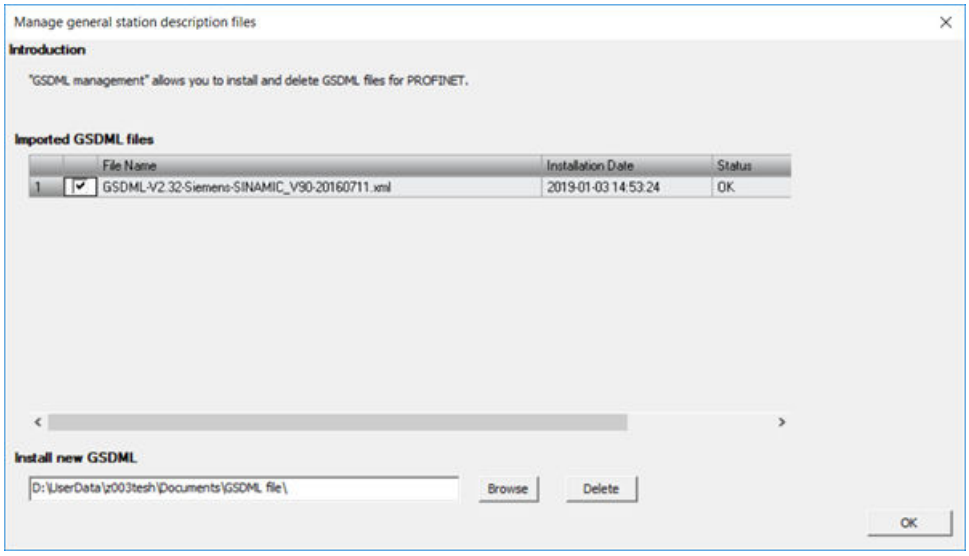


6. Click the "OK" button to close the dialog.

Deleting GSDML files

To delete GSDML files, follow these steps:

1. Click the "GSDML Management" button from the GSDML section of the File menu ribbon strip.
2. Select the GSDML file(s) you want to delete in the "Manage general station description files" dialog.
3. Click the checkbox for your desired GSDML file, and click the "Delete" button. You can also delete multiple GSDML files.



4. Confirm to delete the GSDML file in the pop-up reminder window.
5. Click the "OK" button to close the dialog.

The deleted GSDML file is removed from the "Imported GSDML files" field.

11.4.8.5 PROFINET device naming

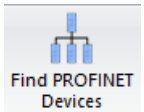
All PROFINET devices **must** have a device name and an IP Address. Use STEP 7-Micro/WIN SMART V3 to define the device names. Device names are assigned to the devices through PROFINET DCP (Discovery and Configuration Protocol).

Make sure that the PROFINET devices and the PC are in the same subnet.

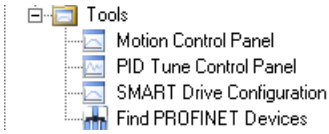
Open the "Find the PROFINET Devices" dialog

To open the "Find PROFINET Devices", use one of the following methods:

- Click the "Find PROFINET Devices" button from the Tools area of the Tools menu ribbon strip.



- Open the Tools folder in the project tree; select the "Find PROFINET Devices" node and press Enter, or double-click the "Find PROFINET Devices" node.

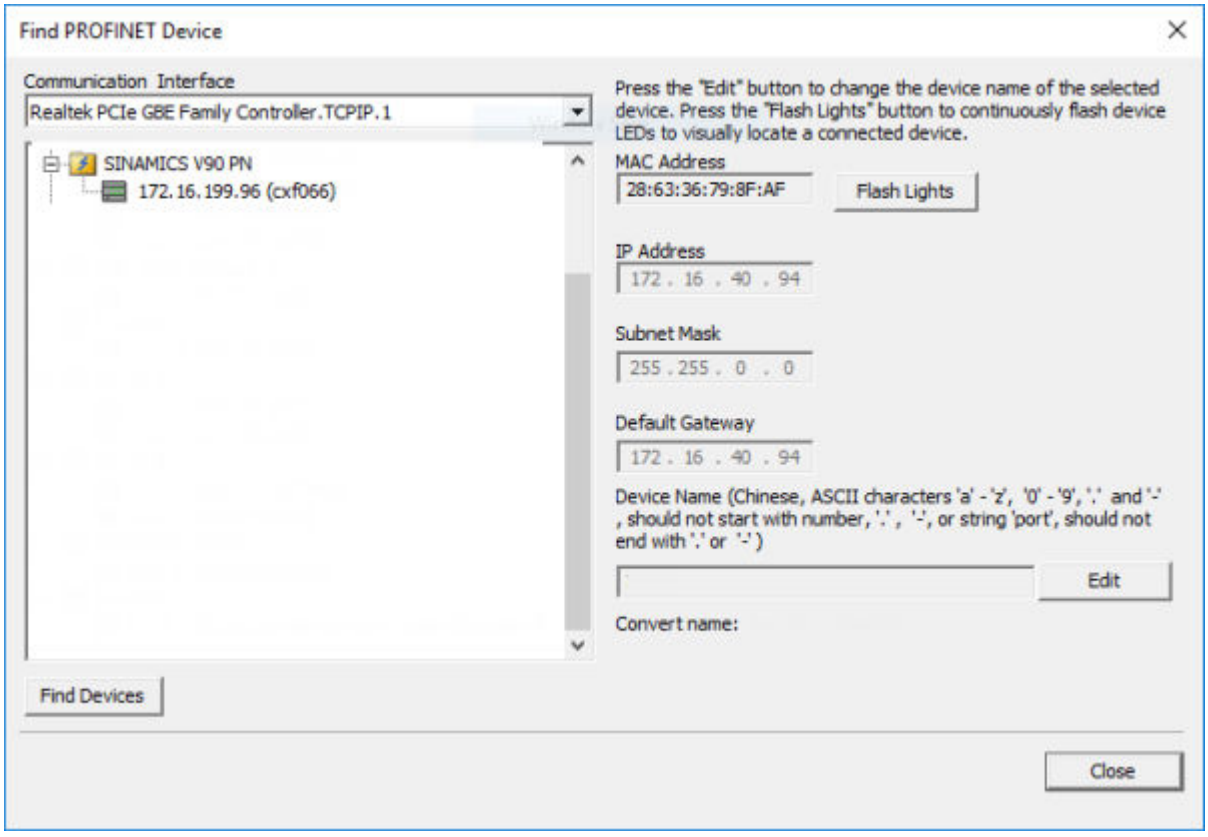


Configuring the PROFINET device name in the "Find PROFINET Devices" dialog

To assign a name to a device, proceed with the following steps:

1. Open the "Find PROFINET Devices" tool.
   The "Find PROFINET Devices" dialog detects all connected and available PROFINET devices on the local Ethernet network. After selecting a PROFINET device, STEP 7-Micro/WIN SMART displays the following detailed information about the device:
   - MAC Address
   - IP Address
   - Subnet Mask
   - Default Gateway
   - Device name
2. Click the "Communication Interface" dropdown list, and select the "TCP/IP" Network Interface Card (NIC) for your programming device.

3. Click the "Find Devices" button to display all available PROFINET devices on the local Ethernet network.



Note: To modify the IP address for the PN devices, go to Configuring CPU as a controller (Page 505).

4. Click the required device.
If you need to identify which device to configure or change, click the "Flash Lights" button. The "Flash Lights" function is work for the devices that accord with DCP standard.

5. Click the "Edit" button to change the device name.

Note

The device name follows the standard DNS (Domain Name System) naming conventions. The naming rule is as follows:

- The maximum supported characters is 63.
- The device name can consist of the lower case letters a through z, the digits 0 through 9, the hyphen character "-", and the period character ".".
- The device name can consist of the Chinese characters (Simplified or Traditional).
- The device name must not have the format n.n.n.n where n is a value of 0 through 999.
- You cannot begin the device name with the string port-nnn or the string port-nnn-nnnnn where n is a digit 0 through 9. For example, port-123 and port-123-45678 are illegal device names.
- The device name cannot start or end with the hyphen character "-" or the period character ".".
- The device name cannot start with the digit.
- The device name can contain name segments. Name segments are separated by the period character ".". Each name segment follows the rules above. For example, plc.1 and smart-.device are illegal device names.

6. Click the "Set" button. The device name is assigned to the device.
7. When finished, click "OK".

11.4.8.6 LED status indicators for PROFINET network

The CPU status LED "Error" on the front panel indicates the operational state of PROFINET:

- For the IO device, the LED blinks red at 1 Hz rate when either of the following conditions occurs:
  - Any IO device loses connection to IO controller
  - Any IO device has an alarm
- For the I-Device, the LED blinks red at 1 Hz rate when either of the following conditions occurs:
  - Any I-Device loses connection to higher-level controller
  - Any I-Device is connected with higher-level controller, but the configuration of I-Device doesn't match with the configuration of higher-level controller.

11.4.8.7 Configuring CPU as a controller

Requirements

Ensure that the following conditions are met before you start to configure CPU as a controller:

- GSDML files are imported for the PROFINET devices. (Page 499)

  Note

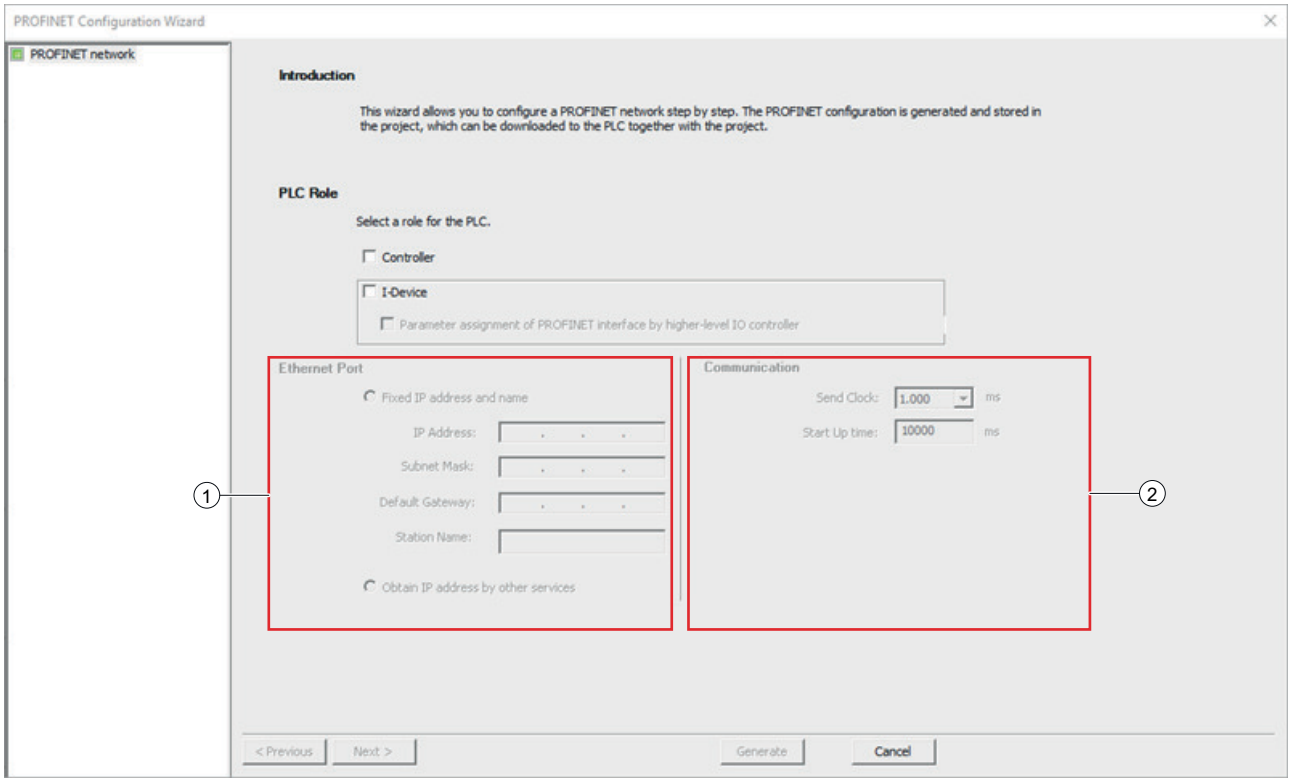
  You can also configure a PROFINET controller through catalog configuration (Page 513).

- The CPU module is selected in the system block dialog (Page 1067).

Procedure

To configure a CPU as a controller, follow these steps:

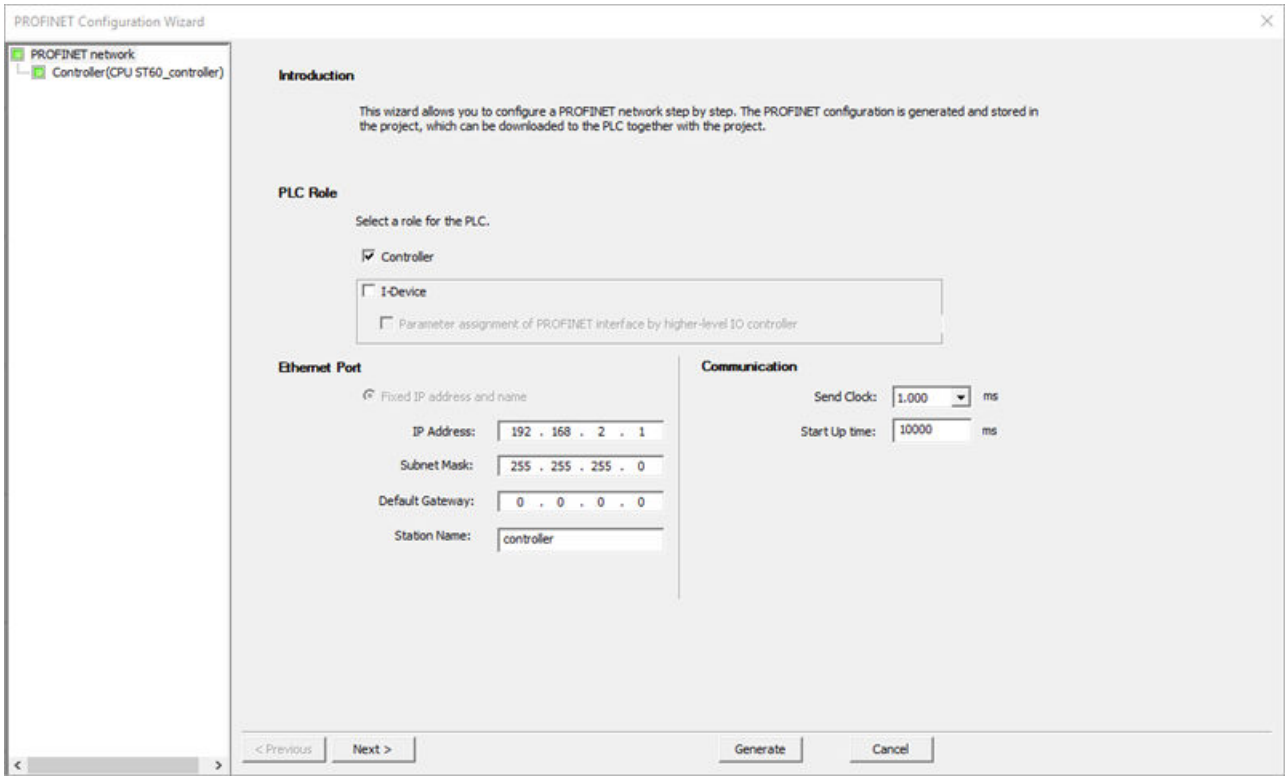
1. Open the PROFINET Wizard.



①	Ethernet Port	There are two methods to configure IP address: • Assign a fixed IP address • Obtain IP address by other services, such as DCP and programming instruction. **Note:** The IP address setting varies as follows: • If you select the CPU as an I-Device, you can select any method above. • If you select the CPU as a controller, you only need to assign a fixed IP address and station name.
②	Communication	You can assign "Send Clock" and "Start up time" for the controller. **Note:** Your setting for the "start up time" impacts the amount of time the CPU takes to switch to RUN mode after the CPU is powered on. The default value for "start up time" is 10 s, and the maximum value is one minute. The situation varies as follows: • If "start up time" is set as zero, the CPU switches to RUN mode immediately. • If the full connection is established within the "start up time", the CPU switches to RUN mode after the connection is established. • If the connection is not established after the "start up time", the CPU switches to RUN mode after the "start up time".

2. Select the "Controller" checkbox.

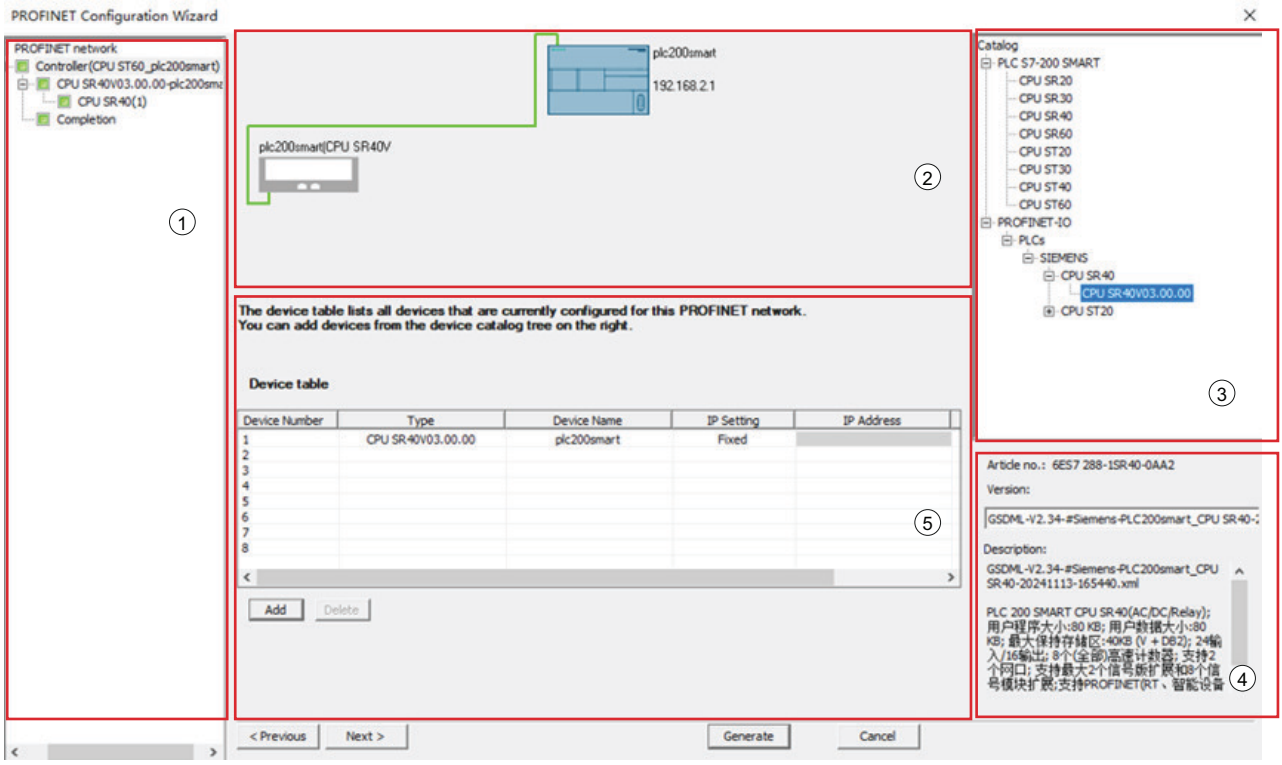
3. Enter a fixed IP address and station name, set send clock and start up time.



Note

Always warm start or power cycle the CPU after changing the name or IP address through PPI download. Otherwise the change will not take effect.

4. Click "Next" in the bottom of the window or click the CPU name in the PROFINET network tree to enter the configuration page.



①	PROFINET network tree	The structure tree of PROFINET network.
②	PROFINET network view	The PROFINET network view.
③	PROFINET device catalog tree	The PROFINET device catalog. There are two nodes in the PROFINET device catalog tree: • PLC S7-200 SMART: S7-200 SMART CPUs are displayed. • PROFINET-IO: Devices defined by the imported GSDML files are displayed. **Note:** If you add S7-200 SMART CPU as the device, you can import GSDML file or just add a specific CPU from the "PLC S7-200 SMART" node (Page 513) in the PROFINET device catalog tree. **Note:** For the unicode character, STEP 7-Micro/WIN SMART V3 supports importing the GSDML file that contains MBCS (Multi-Bytes Character Set) character. If you import the GSDML file that contains special unicode characters, for example, "Ä" and "È", STEP 7-Micro/WIN SMART V3 cannot display GSDML file description and GSDML parameters normally and the question mark "?" shows in the PROFINET wizard.

④	PROFINET device description	The description of the device you select from the PROFINET device catalog tree.
⑤	Device table	List of devices that are configured in the PROFINET network. You can enter the device name or add comments (optional) for the devices in the device table. The address assignment attributes in the "IP setting" drop-down list are as follows: • Fixed: The PROFINET IO Device has a fixed IP for the interface and you do not need to set the IP address here. • Set by user: The IP address of PROFINET IO Device is assigned by DCP protocol through controller, and you must set the IP Address.

5. Select a PROFINET device in the PROFINET device catalog tree and click the "Add" button below the device table. You can also drag and drop the device to the device table.
   Result: The PROFINET device is added in the device table, network view, and PROFINET network tree.
6. Enter the device name, set IP address and add optional comments.

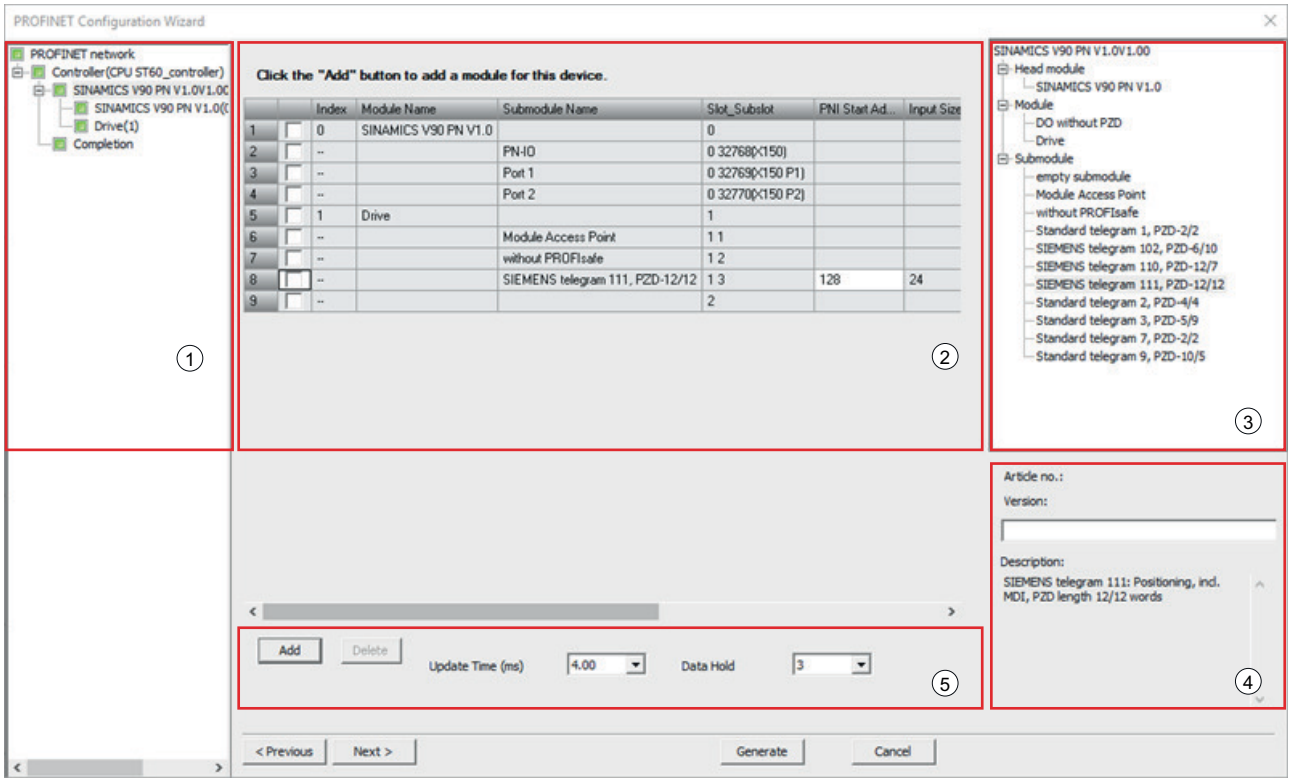
Note

For any PROFINET device, you need to keep the device name in the device table consistent with the actual device name.

Note

The comments that you enter in the PROFINET wizard are not downloaded to PLC. The comments are only saved in the project file.

7. Click the "Next" button or modules in the PROFINET network tree to enter the module configuration dialog.



①	PROFINET network tree	The structure tree of PROFINET network.
②	Submodule table for device	The table lists the module and submodules for the selected devices in the PROFINET network. You can assign the PNI/PNQ start address for the device here. For the range of the PNI/PNQ start address, refer toSystem block (Page 181).
③	Module catalog tree	The list for the selected devices.
④	Device parameters	The parameters for the currently selected device.
⑤	Submodule description	The description of the submodule you select from the PROFINET device catalog tree.

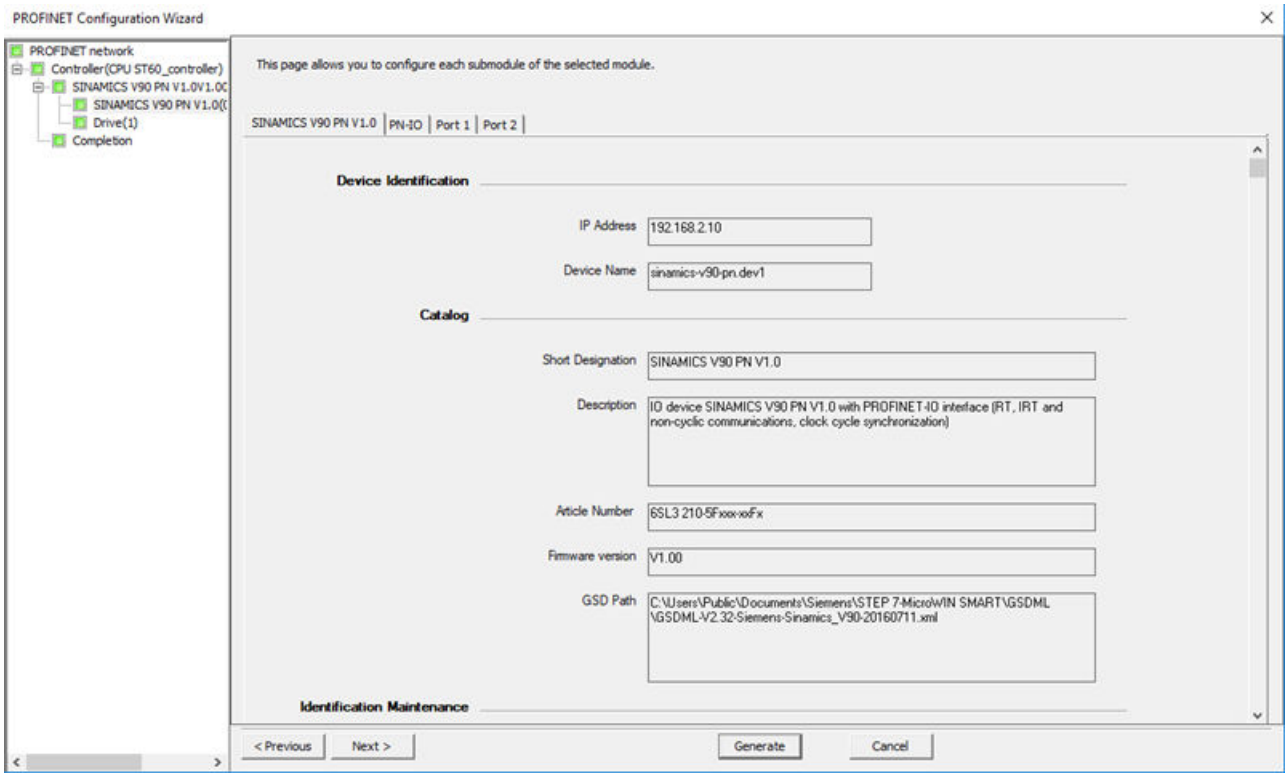
8. Click the module or submodule from the module catalog tree. The slots where you can add a module or submodule turn green.

Note

Make sure the module type you select exists in your real network.

9. Click the "Add" button or drag and drop the module to the slot.

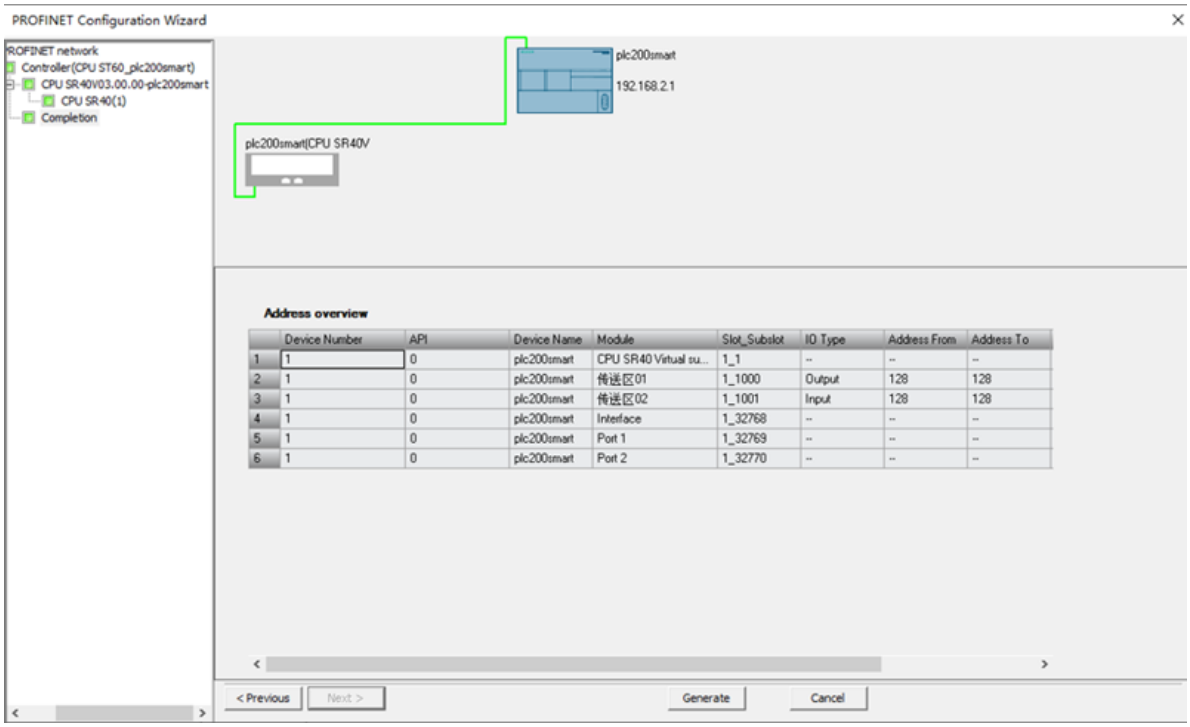
10. Click the "Next" button or the module name in the PROFINET network tree to configure the module or to check the detailed information of the module.



Note

For the "Enable autonegotiation" option in the "Port" tab, you must keep the setting of this option consistent with that of the partner port. Otherwise, the operating parameters of the connected network are not detected, the data transmission rate and transmission mode cannot be optimally set.

11. Verify your PROFINET configuration.



12. Click the "Generate" button to save the configuration.

Result

The PROFINET devices and modules are added to the network and are ready for downloading.

11.4.8.8 Configuring CPU as an I-Device without lower-level PROFINET IO system

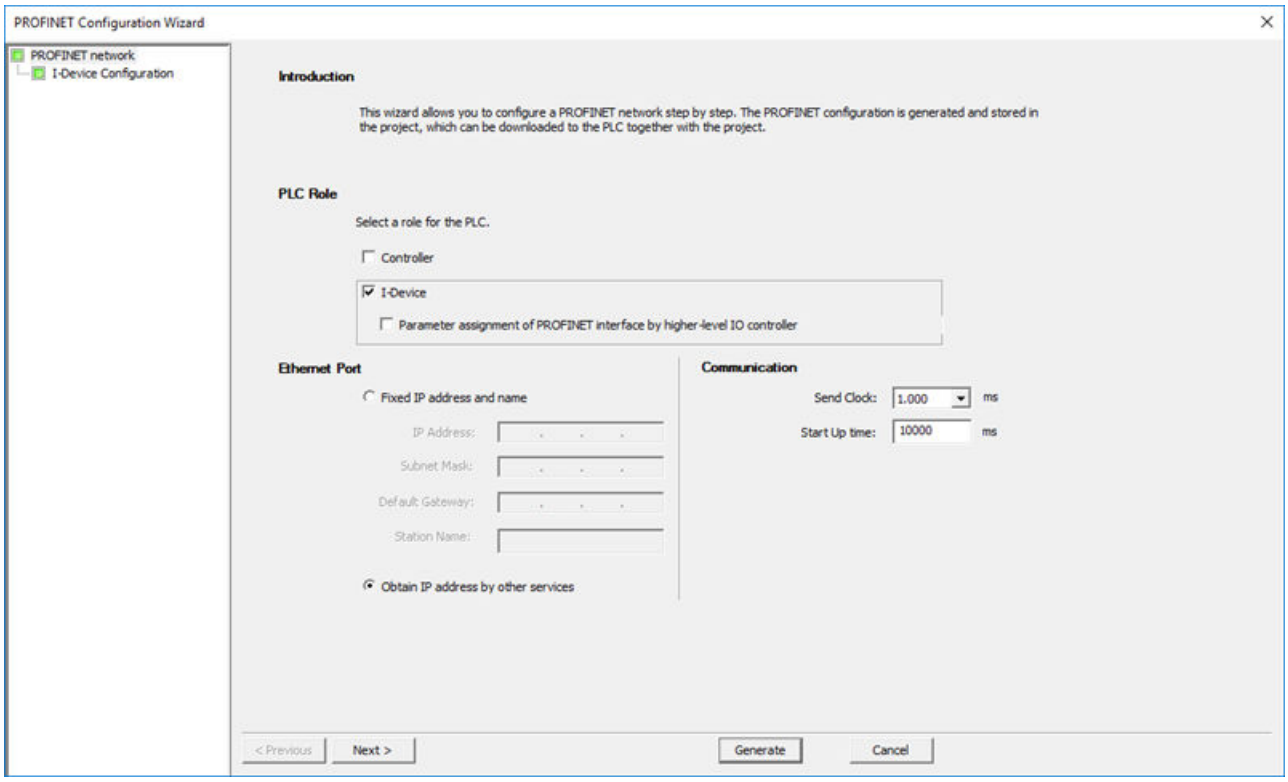
Requirement

Ensure that the CPU module is selected in the system block dialog.

Procedure

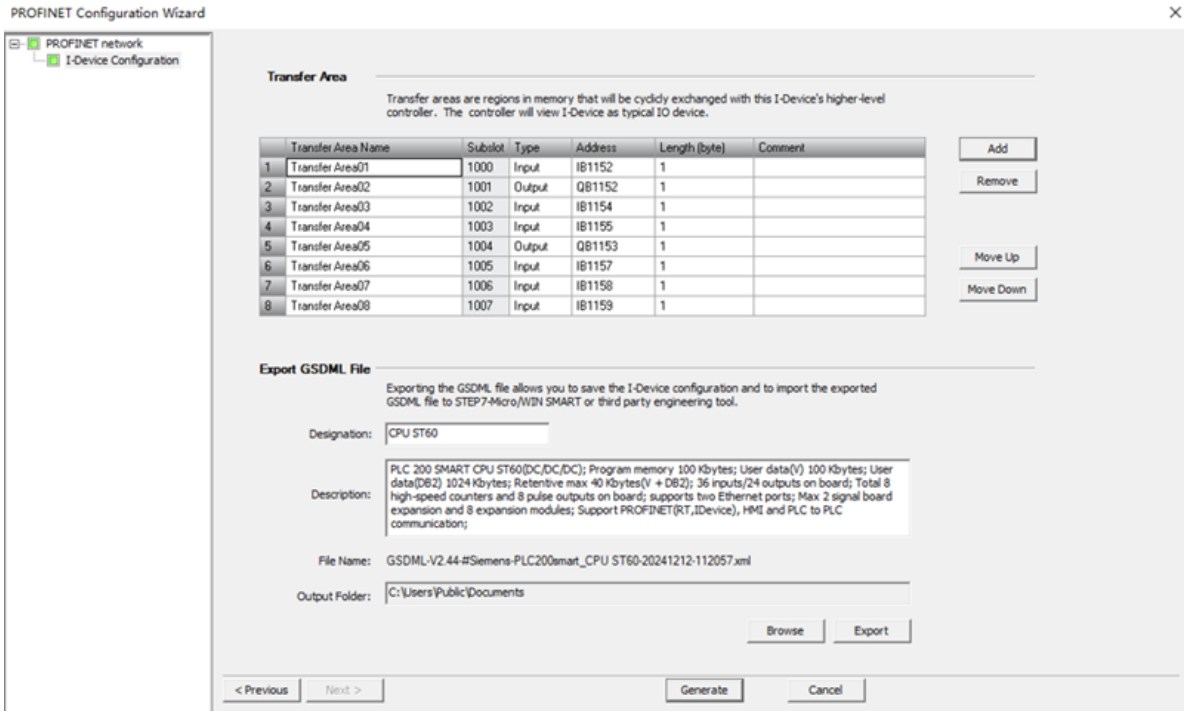
To configure an I-Device project in PROFINET wizard, follow these steps:

1. Open the PROFINET wizard.
2. Select the "I-Device" checkbox.
   The I-Device PROFINET interface parameters and ports can be assigned by the I-Device itself or a higher-level IO controller:
   If you select the "Parameter assignment of PROFINET interface by higher-level IO controller" checkbox, the I-Device PROFINET interface parameters are set by the higher-level controller.



3. Configure an IP address.
   For IP address, you can assign a fixed IP address or obtain IP address by other services.
4. Set "Send Clock" and "Start up time".

5. Click the "Next" button to enter the configuration page. "Transfer area" table and "Export GSDML" field are displayed.



Transfer area work as an interface to the user program of the I-Device CPU and exchange the communication data between higher-level controller and I-Device. Inputs are processed in the user program and outputs are the result of the processing in the user program.

Note

- The maximum number of supported transfer area is 8.
- At least one transfer area must be configured.

The "Transfer Area" table lists the following items:

Items	Description
Transfer area name	Submodule name of one subslot. You can define the transfer area name or use the default transfer area name. Transfer area name is not downloaded to PLC. **Note:** The maximum number of supported characters is 64.
Subslot	Subslot number. The subslot number starts from 1000, and the system automatically generates a sequential digit for the subslot number.
Type	Transfer area type. You can select one transfer area as input or output.

Items	Description
Address	Transfer area IO address. You can enter the input or output address. • The input range is from IB1152 to IB1279. • The output range is from QB1152 to QB1279.
Length	Transfer area IO data length. • The maximum size of transfer area total inputs is 128 bytes. • The maximum size of transfer area total outputs is 128 bytes.
Comment	You can add optional comments for one transfer area.

The "Export GSDML" field lists the following items:

Items	Description
Designation	The device family name is displayed by default. For example, CPU ST60. You can also enter any information as you need: • The Chinese, English and digit character are supported. • The character '|', '\\', '/', ':', '\', '', '<', '>', '*' and '?' are not supported. • The maximum number of supported characters is 20.
Description	The CPU information is displayed by default or you can enter any information as you need.
File name	GSDML file name is automatically generated, and is of an assigned format.
Output folder	The folder where you want to save the GSDML file. You can click the "Browse" button to select one folder or use the default folder.

6. Configure the transfer area.
7. Click the "Export" button to export the GSDML file.
8. Click the "Generate" button to save the configuration.
9. Download this I-Device project to a CPU.

This example describes how to configure a PROFINET network project where one S7-200 SMART CPU works as an I-Device without lower-level PROFINET IO system and one S7-200 SMART CPU works as a higher-level controller.

There are two methods to configure this PROFINET network:

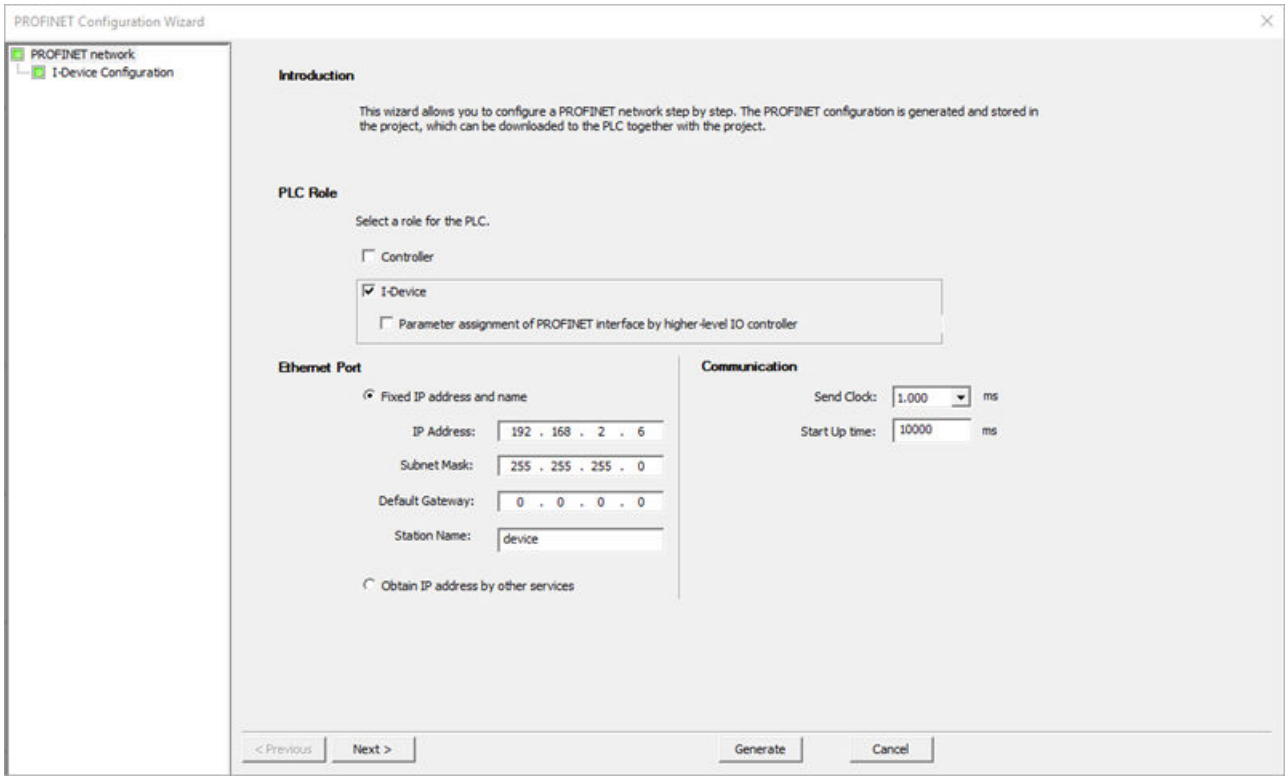
- Method 1: Configuring a PROFINET network by importing GSDML file.
- Method 2: Configuring a PROFINET network through catalog configuration.
  Catalog configuration means that there are built-in S7-200 SMART CPUs under the "PLC S7-200 SMART" node of PROFINET device catalog tree. You can select a CPU model in the "PLC S7-200 SMART" node to add it as IO device of one controller, and you don't need to import the GSDML file.

Preparation

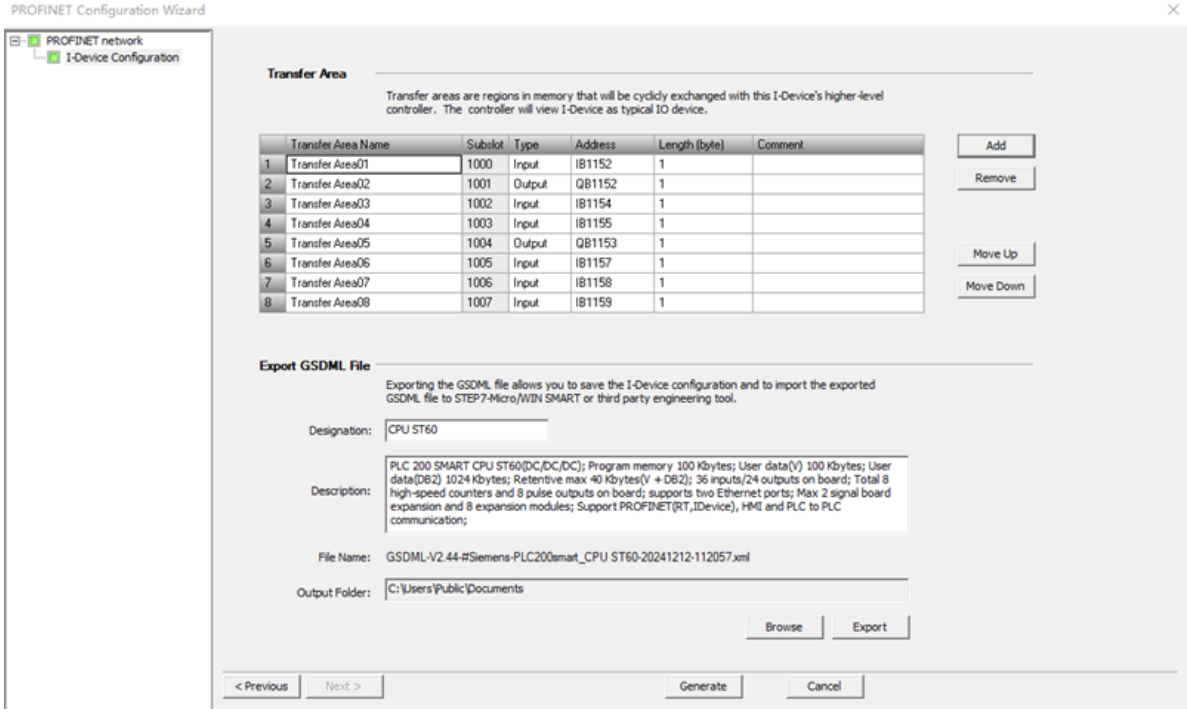
1. Select one ST-60 S7-200 SMART CPU as an I-Device, name it as "device" and assign the following IP address:
   - IP address: 192.168.2.6
   - Subnet Mask: 255.255.255.0
   - Default Gateway: 0.0.0.0
2. Select one ST-60 S7-200 SMART CPU as a controller, name it as "controller" and assign the following IP address:
   - IP address: 192.168.2.1
   - Subnet Mask: 255.255.255.0
   - Default Gateway: 0.0.0.0

You can configure a PROFINET network by importing GSDML file as follows:

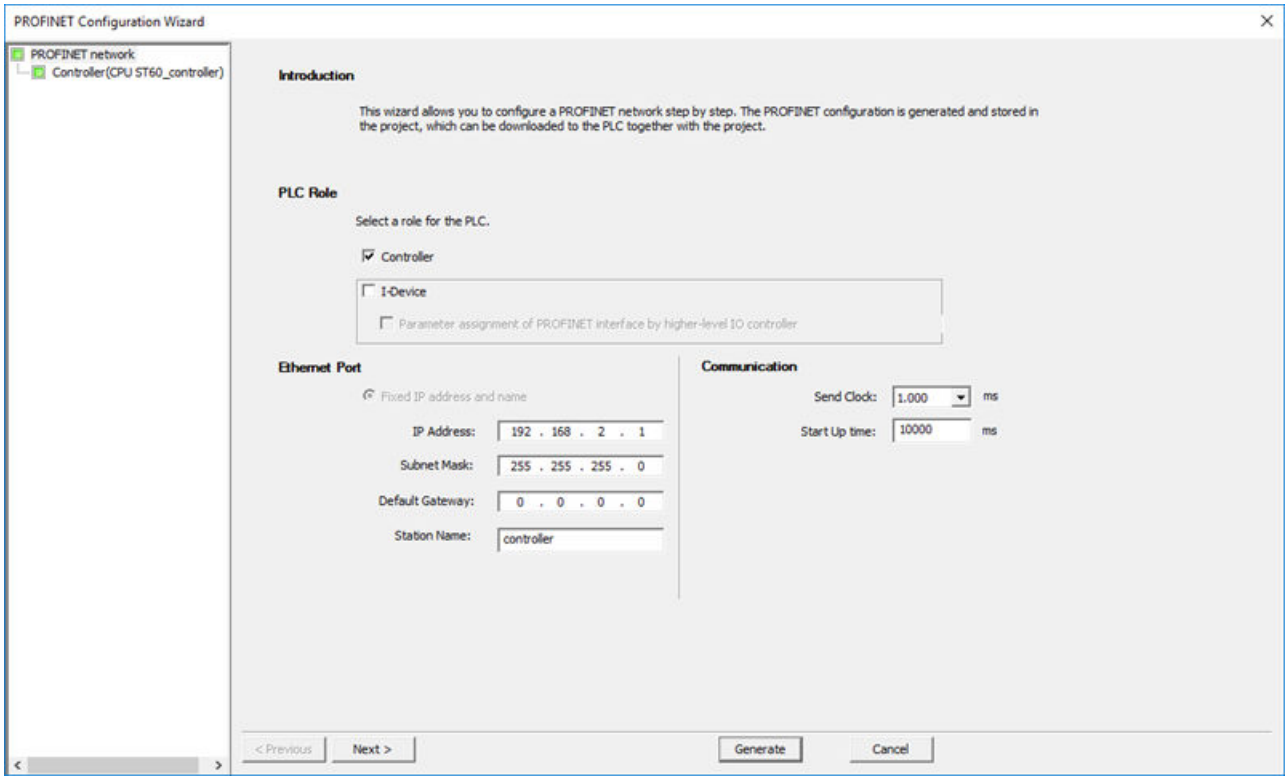
1. Open the PROFINET wizard, select the "I-Device" checkbox and assign a fixed IP address and name as follows:



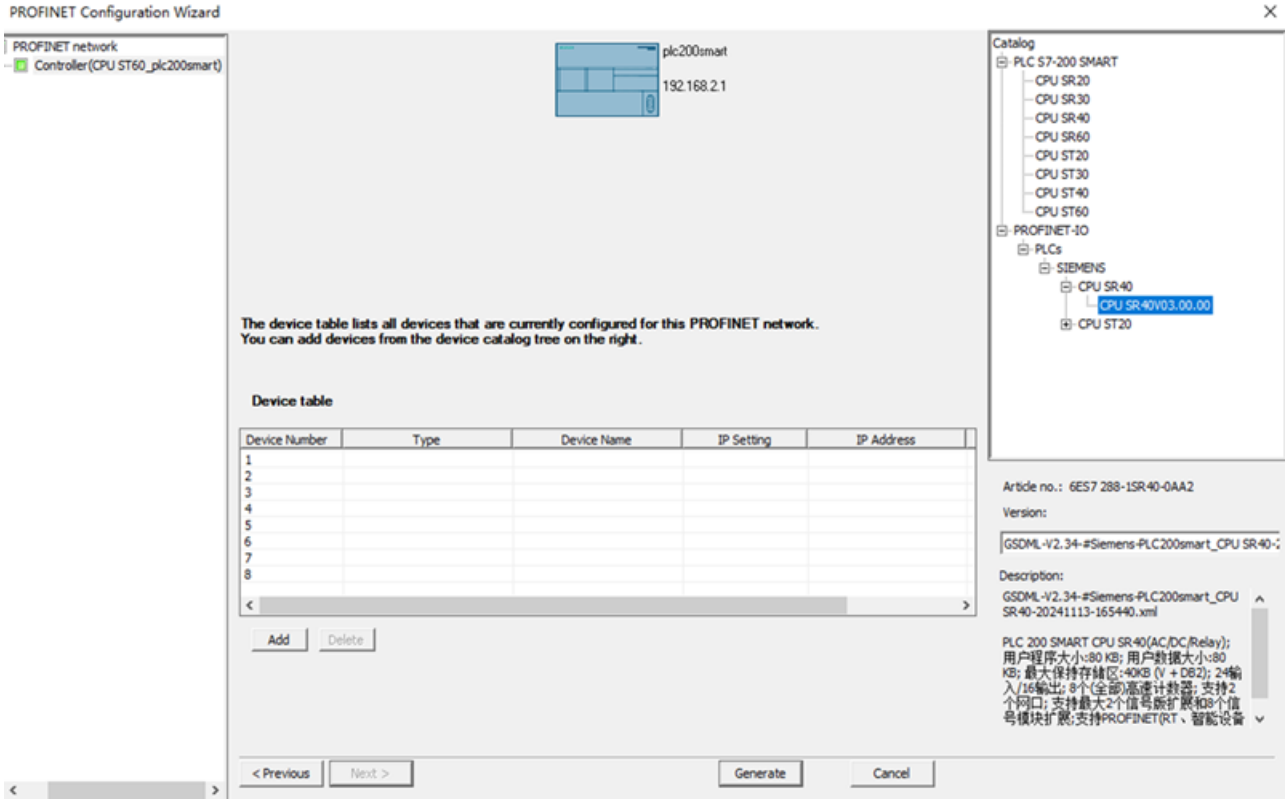
2. Configure the following 8 transfer areas:



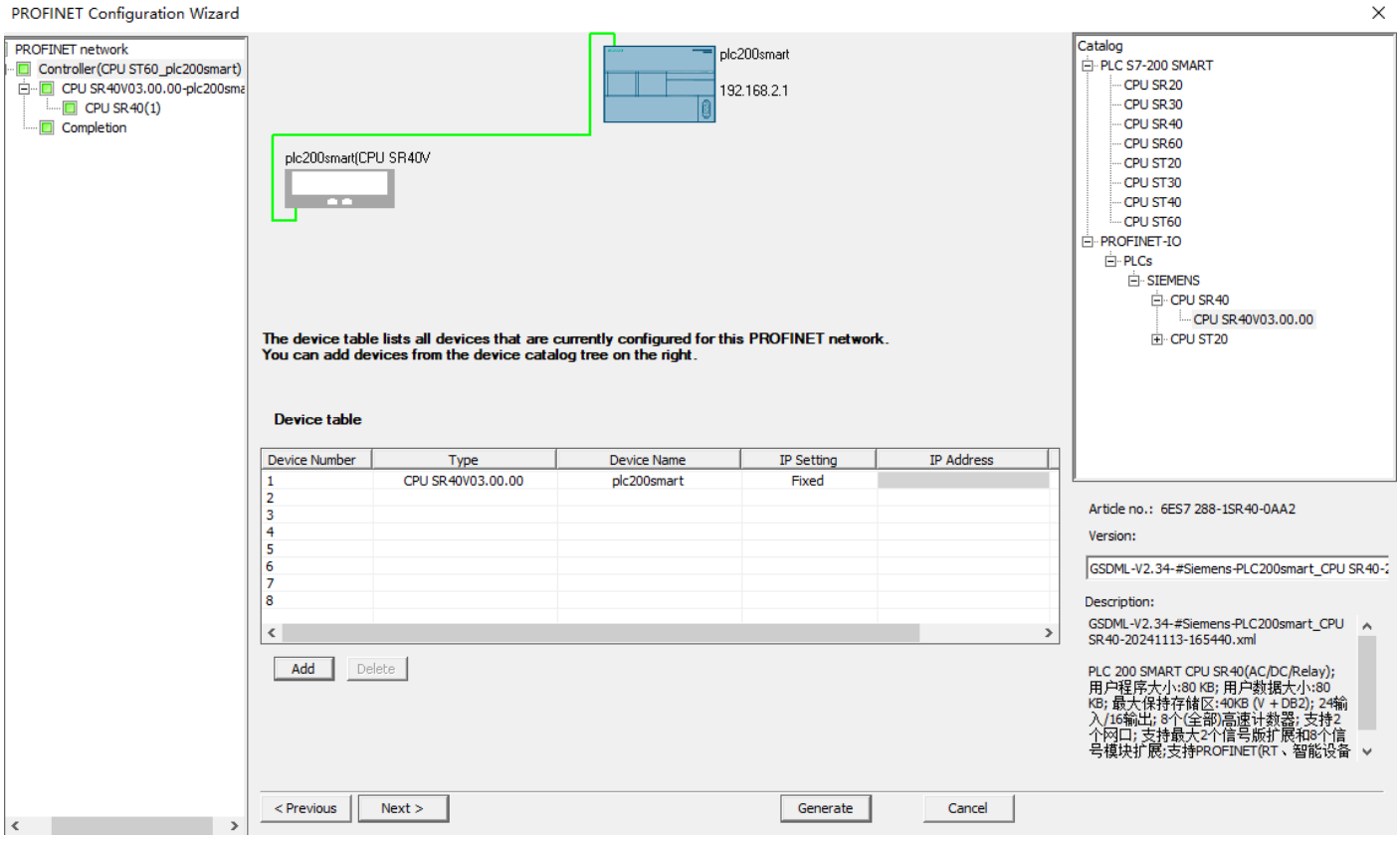
3. Export the GSDML file. An I-Device GSDML file is created.
4. Generate the configuration. Then an I-Device project is created.
5. Download this project to I-Device CPU.
6. Open a new project and import the I-Device GSDML file.
7. Navigate to the PROFINET wizard. Select the "Controller" checkbox, assign a fixed IP address and name as follows:



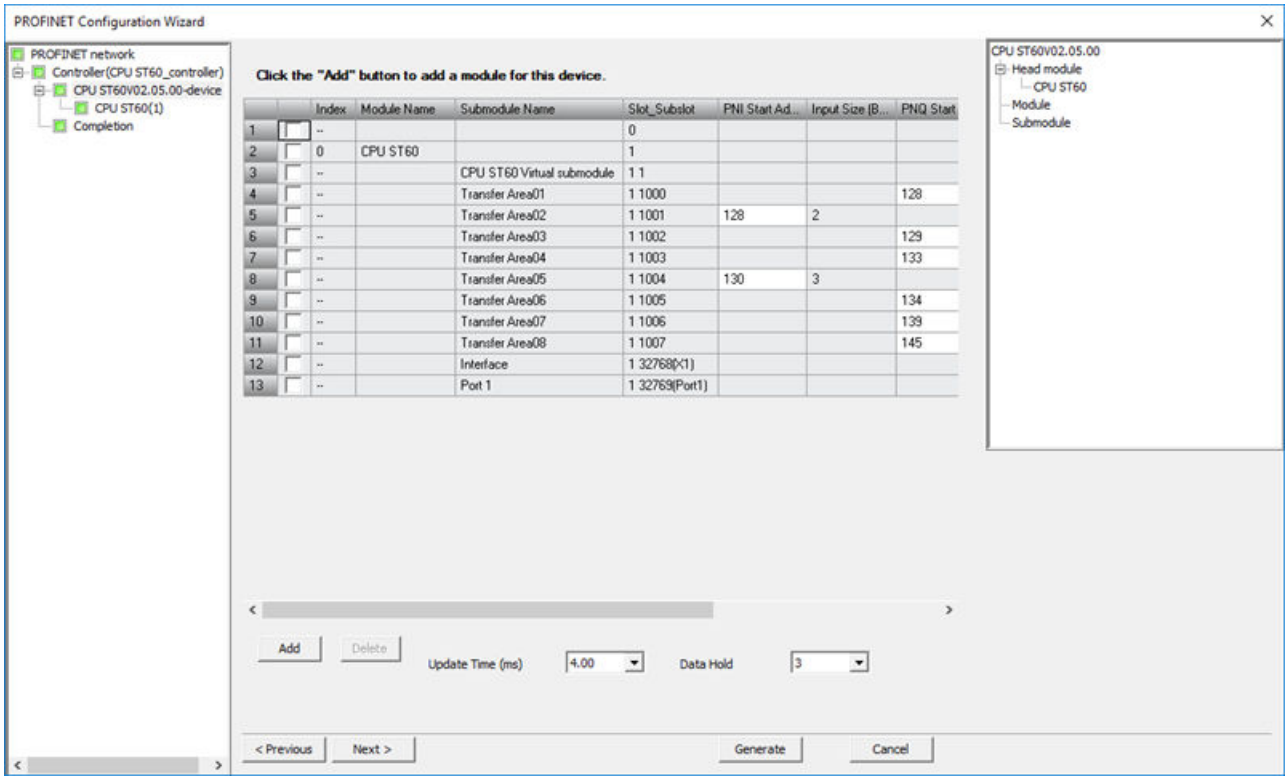
8. Navigate to the next page, and the CPU ST60 defined by the imported GSDML file is displayed in the "PROFINET-IO" node under "PROFINET device catalog tree".



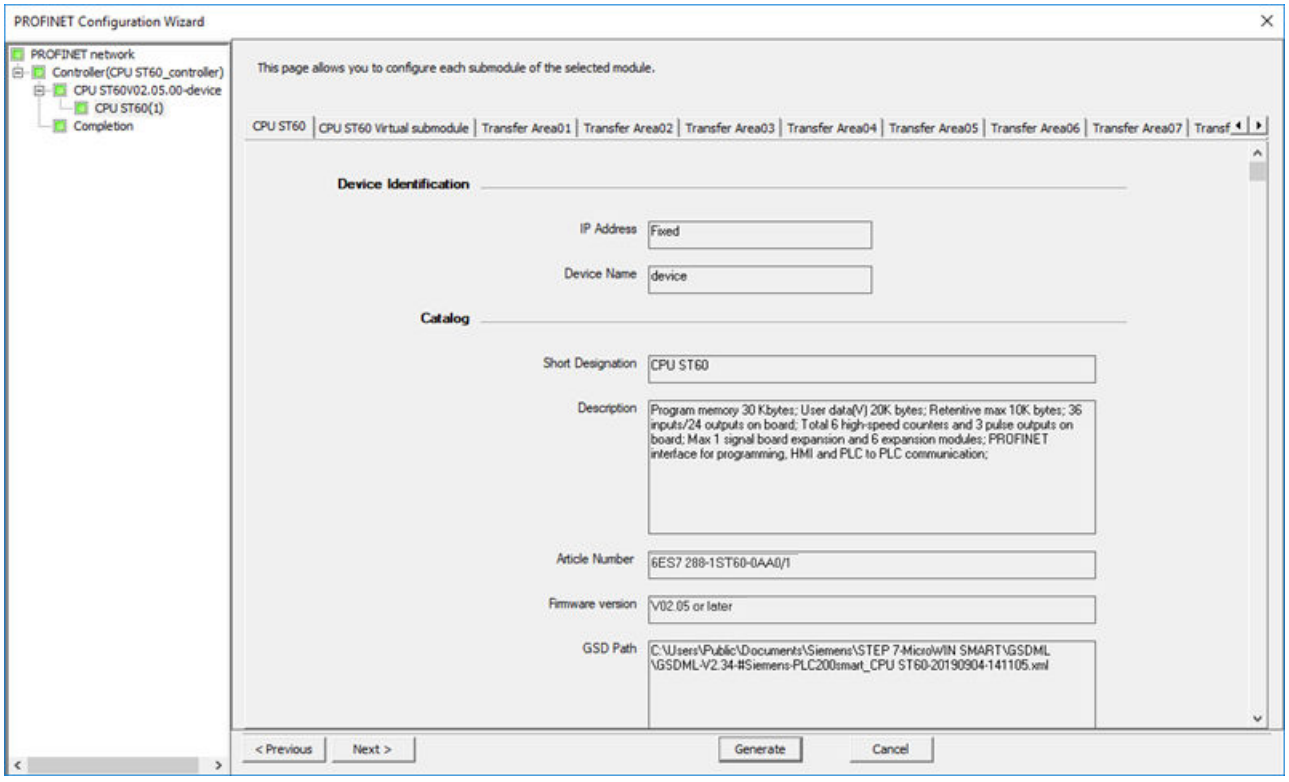
9. Add "CPU ST60" to device table:
   Since "Fixed IP address and name" is selected for I-Device configuration in this example, and you add this I-Device in device table by importing the I-Device GSDML file, you do not need to change the device name and IP address here.



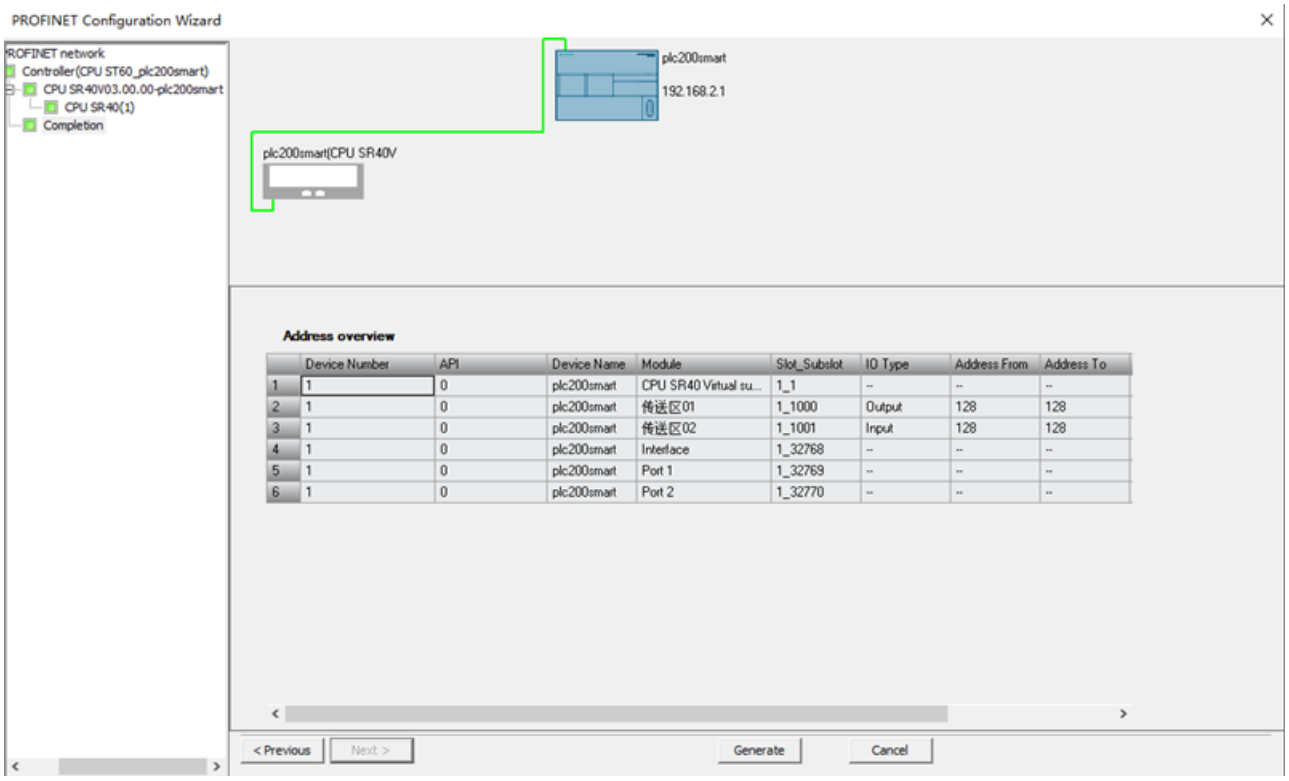
10. Navigate to the next page and the transfer area configuration information is automatically displayed in the module configuration page.



11. Navigate to the next page to check the submodule information.



12. Verify the configuration in the completion page.



13. Click the "Generate" button to save the configuration.
14. Download this project to controller CPU.

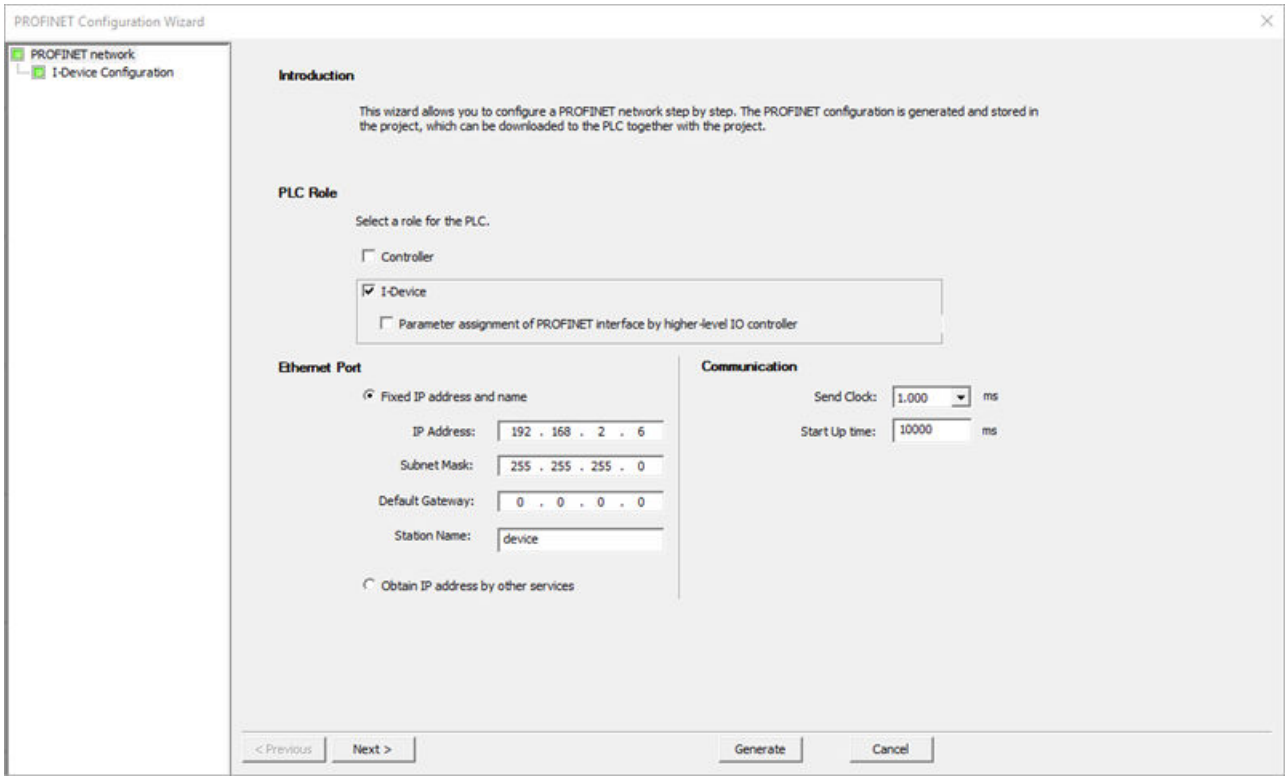
Note

For a S7-200 SMART CPU that works as a controller, if this controller configures I&M data record of its lower-level device and you download the project to the controller, after this controller connects with its lower-level device, the I&M data record will be written to the lower-level device.

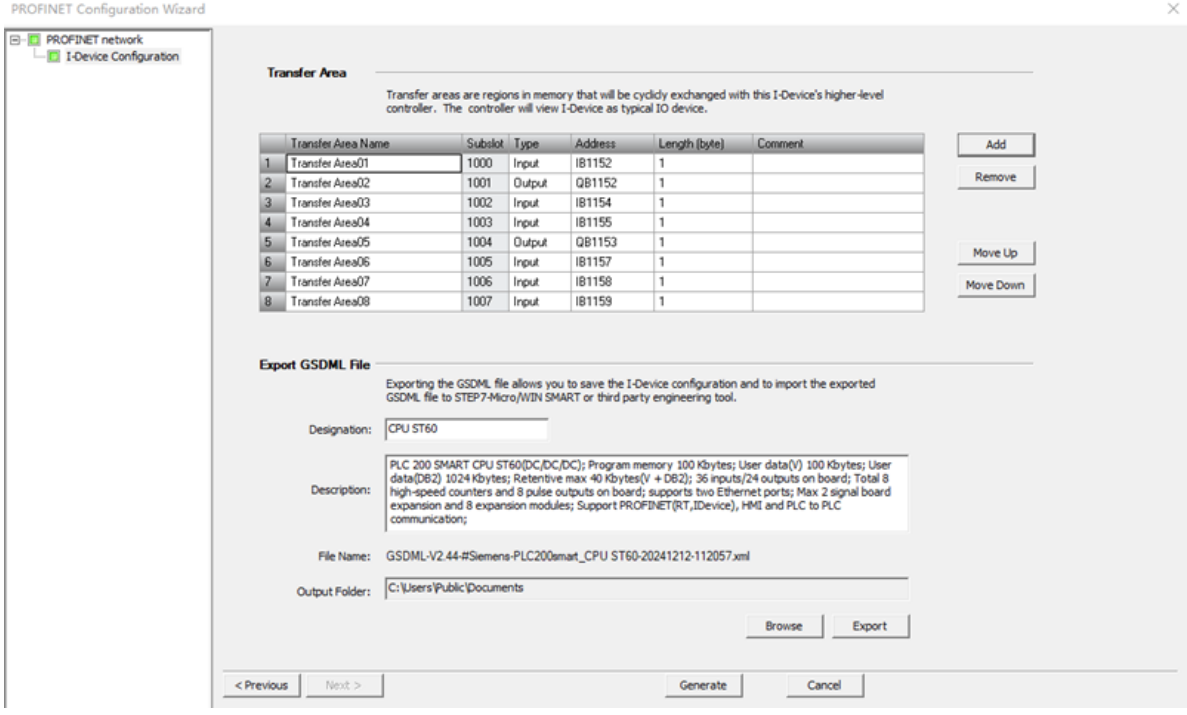
For a S7-200 SMART CPU that works as an I-Device, the I&M data record will be cleared if you download a project to the I-Device.

You can configure a PROFINET network through catalog configuration as follows:

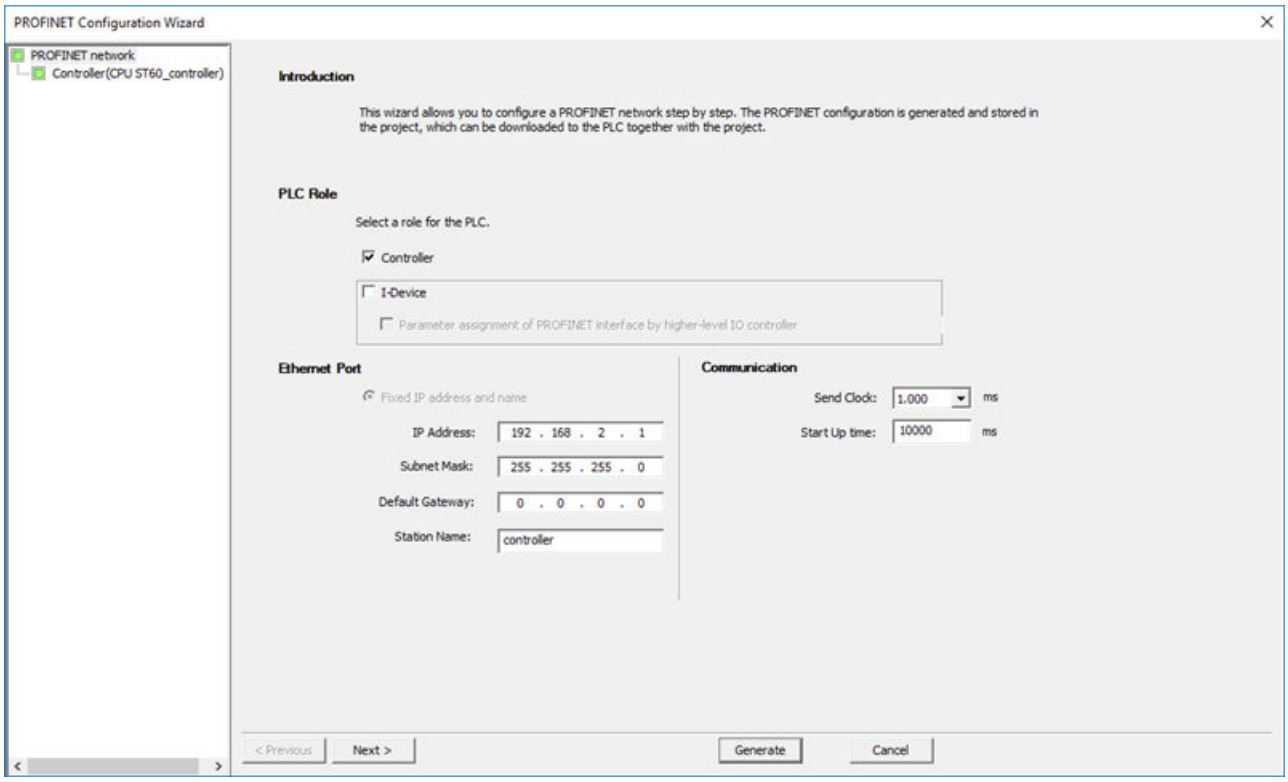
1. Open the PROFINET wizard, select the "I-Device" checkbox and assign a fixed IP address and name as follows:



2. Configure the following 8 transfer areas:



3. Click the "Generate" button to save the configuration.
4. Download this I-Device project to I-Device CPU.
5. Open a new project and navigate to the PROFINET wizard. Select the "Controller" checkbox, and assign a fixed IP address and name as follows:



6. Navigate to the device configuration page. S7-200 SMART CPUs are displayed in the "PLC S7-200 SMART" node of "PROFINET device catalog" tree.

7. Add CPU ST60 in the device table, and configure it as follows:
   - Device name: device

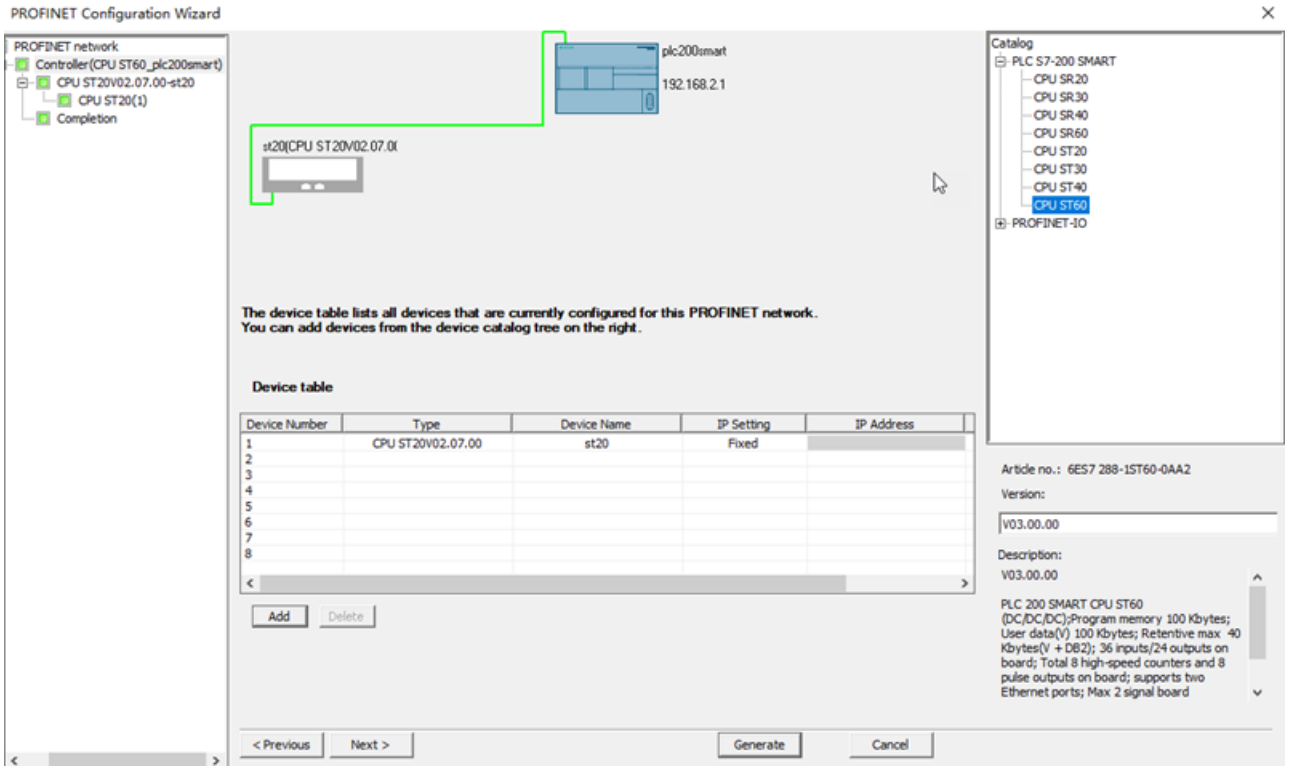
     Note

     If you select "Fixed IP address and name" for I-Device configuration and add this I-Device as IO device of a higher-level controller, keep the "Device Name" here consistent with the "Station Name" of I-Device.

   - IP Setting: Fixed.

     Note

     You need to select the "IP Setting" option as "Fixed" for IP address of I-Device is fixed in the I-Device project.

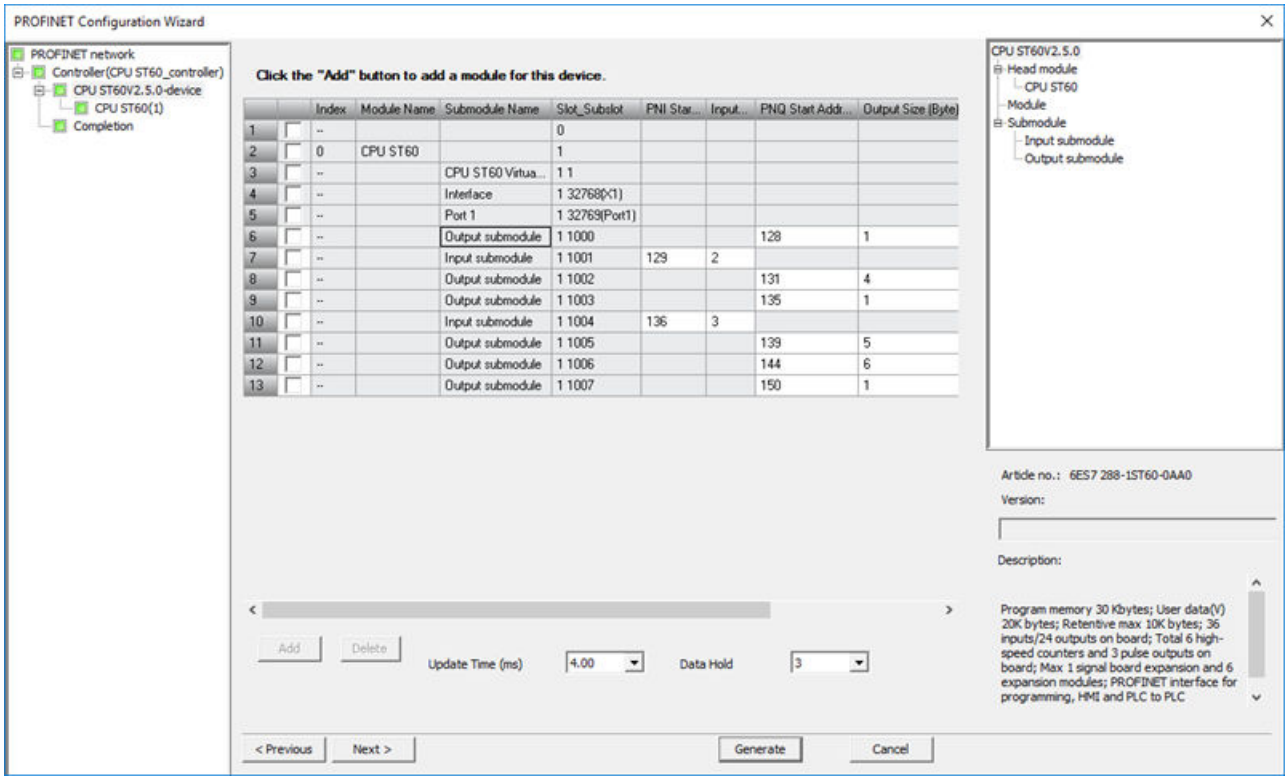


8. Click the "Next" button and configure modules.

Note

For subslots in the "Slot_Subslot" column, select the "Input submodule" and click the "Add" button to add the submodule if you set this slot as "output transfer area" in the I-Device configuration.

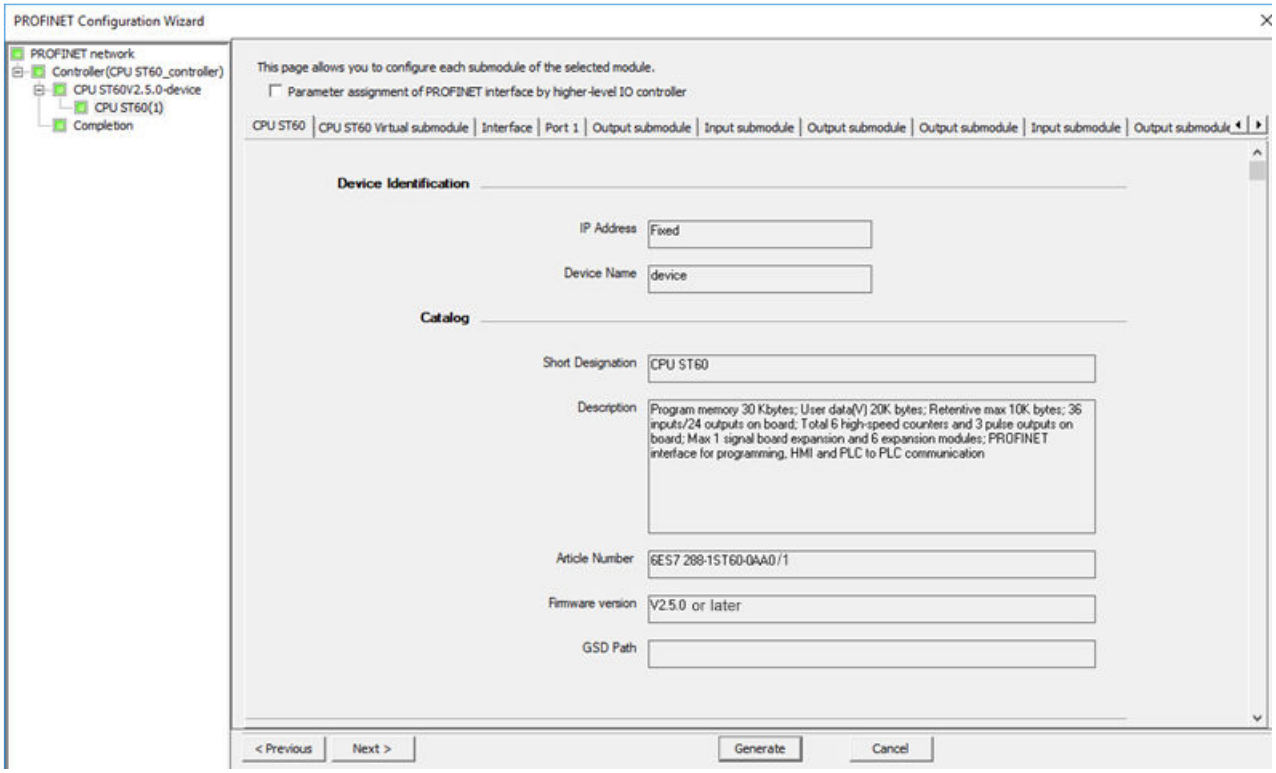
Select the "Output submodule" and click the "Add" button to add the submodule if you set this slot as "input transfer area" in the I-Device configuration. For example, the subslot 1000 is set as "Input" in the transfer area of I-Device project, here you need to set this subslot as "output submodule".



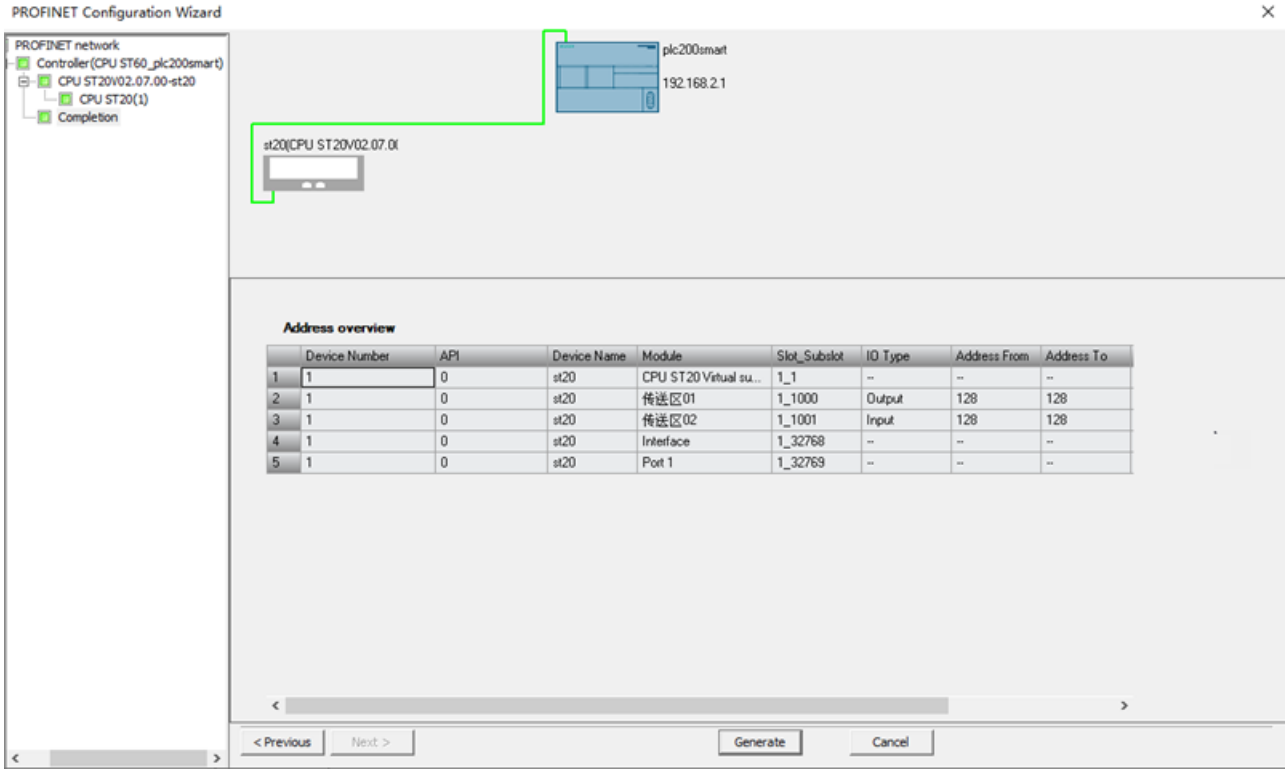
9. Click the "Next" button and configure submodules or check the submodule configuration page.

Note

If you select the checkbox "Parameter assignment of PROFINET interface by higher-level IO controller" when you configure I-Device project, you need to select the checkbox in this page. Then you can configure parameters under "Port 1" tab.



10. Verify the configuration in the completion page.



11. Click the "Generate" button to save the configuration.

12. Download this project to controller CPU.

11.5 PROFIBUS

The PROFIBUS protocol is designed for high-speed communications with distributed I/O devices (remote I/O). A PROFIBUS system uses a bus master to poll DP I/O devices distributed in a multi-drop fashion on an RS485 serial bus.

There are many PROFIBUS devices available from a variety of manufacturers. These devices range from simple input or output modules to motor controllers and PLCs. A PROFIBUS DP device is any peripheral device which processes information and sends its output to the master. The DP device forms a passive station on the network (since it does not have bus access rights) and can only acknowledge received messages or send response messages to the master upon request. All PROFIBUS DP devices have the same priority, and all network communication originates from the master.

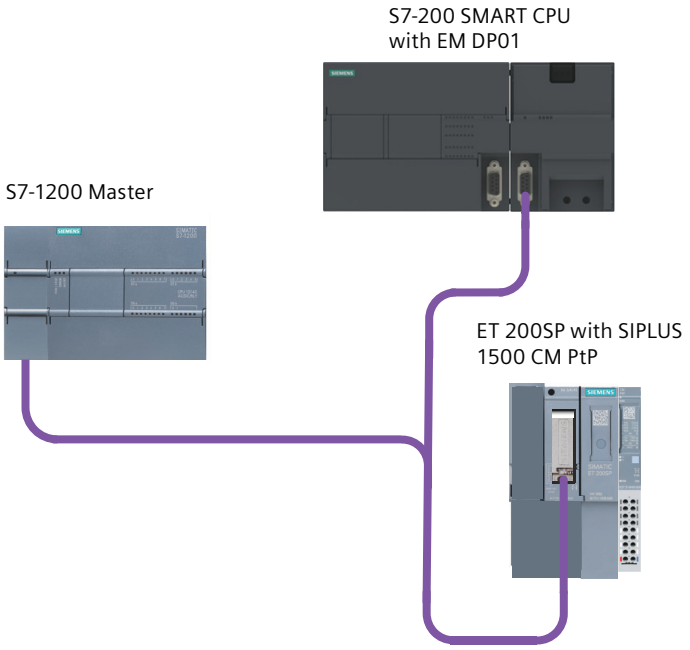
A PROFIBUS master forms an "active station" on the network. PROFIBUS DP defines two classes of masters. A class 1 master (normally a central programmable controller (PLC) or a PC running special software) handles the normal communication or exchange of data with the DP devices assigned to it. A class 2 master (usually a configuration device, such as a laptop or programming console used for commissioning, maintenance, or diagnostics purposes) is a special device primarily used for commissioning DP devices and for diagnostic purposes.

PROFIBUS networks typically have one master and several DP I/O devices. (Refer to the figure below.) You configure the master device to know what types of DP devices are connected and at what addresses. The master initializes the network and verifies that the DP devices on the network match the configuration. The master continuously writes output data to the DP devices and reads input data from them.

When a PROFIBUS DP master configures a DP device successfully, it then owns that DP device. If there is a second master device on the network, it has very limited access to the DP devices owned by the first master.

The EM DP01 PROFIBUS DP module connects the S7-200 SMART CPU to a PROFIBUS network as a DP device. The EM DP01 can be the communications partner of DP V0/V1 masters. You can access the EM DP01 GSD file at Siemens Customer Support.

In the figure below, the S7-200 SMART CPU is a DP device for an S7-1200 controller:



You can configure two PROFIBUS EMs per S7-200 SMART V3 CPU.

The local CPU stores the PROFIBUS EM configuration data, and you set the PROFIBUS addresses with switches on each module. This allows simple replacement of these communications modules when necessary.

11.5.1 EM DP01 PROFIBUS DP module

11.5.1.1 Distributed Peripheral (DP) standard communications

PROFIBUS DP (or DP Standard) is a remote I/O communications protocol defined by the European Standard EN 50170. Devices that adhere to this standard are compatible even though they are manufactured by different companies. DP stands for distributed peripherals, that is, remote I/O. PROFIBUS stands for Process Field Bus.

The EM DP01 PROFIBUS DP module has implemented the DP Standard protocol as defined for DP devices in the following communications protocol standards:

- EN 50 170 (PROFIBUS) describes the bus access and transfer protocol and specifies the properties of the data transfer medium.
- EN 50 170 (DP Standard) describes the high-speed cyclic exchange of data between DP masters and DP devices. This standard defines the procedures for configuration and parameter assignment, explains how cyclic data exchange with distributed I/O functions, and lists the diagnostic options which are supported.

A DP master is configured to know the addresses, DP device types, and any parameter assignment information that the DP devices require. The DP master is also told where to place data that is read from the DP devices (inputs) and where to get the data to write to the DP devices (outputs). The DP master establishes the network and then initializes its DP devices. The DP master writes the parameter assignment information and I/O configuration to the DP device. The DP master then reads the diagnostics from the DP device to verify that the DP device accepted the parameters and the I/O configuration. The DP master then begins to exchange I/O data with the DP device. Each transaction with the DP device writes outputs and reads inputs. The data exchange mode continues indefinitely. The DP devices can notify the DP master if there is an exception condition, and the DP master then reads the diagnostic information from the DP device.

Once a DP master has written the parameters and I/O configuration to a DP device, and the DP device has accepted the parameters and configuration from the DP master, the DP master owns that DP device. The DP device only accepts write requests from the DP master that owns it. Other DP masters on the network can read the DP device's inputs and outputs, but they cannot write anything to the DP device.

11.5.1.2 Using the EM DP01 to connect an S7-200 SMART as a DP device

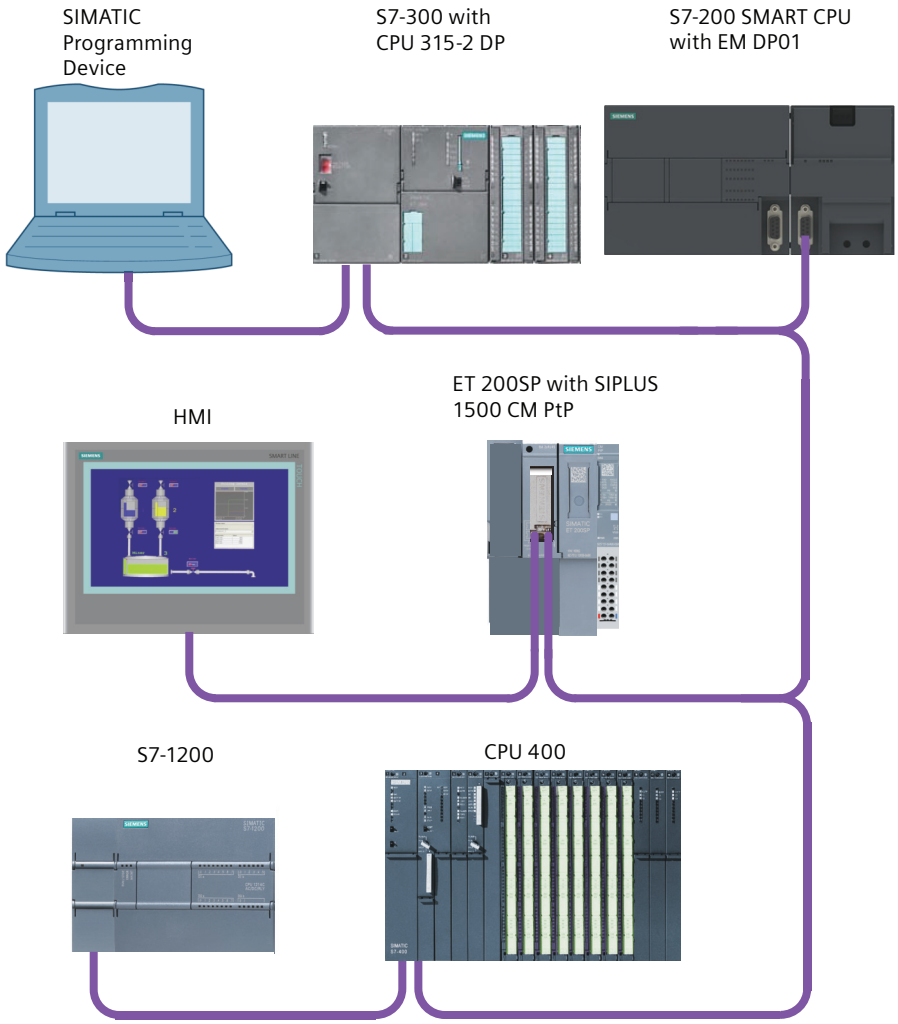
You can connect the S7-200 SMART CPU to a PROFIBUS DP network through the EM DP01 PROFIBUS DP module. You connect the EM DP01 to the S7-200 SMART CPU as an expansion module, and you connect the PROFIBUS network to the EM DP01 PROFIBUS DP module through its DP communications port. This port operates at any PROFIBUS baud rate between 9.6 Kbps and 12 Mbps. Refer to the EM DP01 PROFIBUS DP module Technical Specifications for the baud rates supported.

As a PROFIBUS DP device, the EM DP01 accepts several different I/O configurations from the DP master, allowing you to tailor the amount of data transferred to meet the requirements of the application. Unlike many DP devices, the EM DP01 does not transfer only I/O data. The EM DP01 also transfers inputs, counter values, timer values, or any other values that you move to the variable memory in the S7-200 SMART CPU. Likewise, the EM DP01 transfers data from the DP master to the variable memory in the S7-200 SMART CPU. You can then move this data from variable memory to other data areas.

You can attach the DP port of the EM DP01 PROFIBUS DP module to a DP master on the network and still communicate as an MPI device with other master devices such as SIMATIC

HMI devices or S7-300 / S7-400 CPUs on the same network. The following figure shows a PROFIBUS network with an S7-200 SMART CPU and an EM DP01 PROFIBUS DP module:

- The S7-300 with CPU 315-2 is the DP master and has been configured by a SIMATIC programming device with STEP 7 programming software. The S7-315-2 DP can read data from or write data to the EM DP01, from 1 byte up to 244 bytes.
- The S7-200 SMART CPU is a DP device owned by the CPU 315-2. The ET 200 I/O module is also a DP device owned by the CPU 315-2.
- The S7-400 CPU is attached to the PROFIBUS network and is reading data by means of X_GET instructions in the S7-400 CPU user program. (Other SIMATIC CPUs can use DB1 to access V memory in the S7-200 SMART CPU.)



11.5.1.3 Configuring the EM DP01

Procedure

1. To use the S7-200 SMART EM DP01 PROFIBUS DP module as a DP device, you must set the station address of the DP port to match the address in the configuration of the DP master. The station address is set with the rotary switches on the EM DP01.
2. You must power cycle the S7-200 SMART CPU after you have made a switch change in order for the new DP device address to take effect.

Result

The DP master device exchanges data with each of its DP devices by sending information from its output area to the DP device's output buffer. The DP device responds to the message from the DP master by returning an input buffer which the DP master stores in an input area.

Configuration steps

The S7-200 SMART EM DP01 PROFIBUS DP module can be configured by the DP master to accept output data from the DP master and return input data to the DP master. The output and input data buffers reside in the variable memory (V memory) of the S7-200 SMART CPU. When you configure the DP master, you define the byte location in V memory where the output data buffer starts as part of the parameter assignment information for the EM DP01. You also define the I/O configuration as the amount of output data to be written to the S7-200 SMART CPU and amount of input data to be returned from the S7-200 SMART CPU. The EM DP01 determines the size of the input and output buffers from the I/O configuration. The DP master writes the parameter assignment and I/O configuration information to the EM DP01. The EM DP01 then transfers the V memory address and input and output data lengths to the S7-200 SMART CPU. These values are available in the special memory (SM) of the S7-200 SMART CPU for use in the user program. Refer to the SM status information in "User program considerations" for further details.

11.5.1.4 Data consistency

PROFIBUS supports three types of data consistency:

- Byte: Ensures that bytes are transferred as whole units
- Word: Ensures that word transfers cannot be interrupted by other processes in the CPU.
- Buffer: Ensures that the entire buffer of data is transferred as a single unit, uninterrupted by any other process in the CPU.

The EM DP01 always utilizes buffer consistency in its data handling.

S7-200 SMART CPU and EM DP01 data buffer consistency

The EM DP01 and S7-200 SMART CPU offer buffer consistency for the entire transfer:

- The EM DP01 receives the outputs from the DP master in one message.
- The EM DP01 transfers all outputs to the S7-200 SMART CPU in one message that cannot be interrupted.
- The S7-200 SMART CPU transfers all outputs to the V memory area at one time. The transfer cannot be interrupted by a user interrupt.

The same consistency is true for the inputs to the DP master:

- The S7-200 SMART CPU transfers all inputs from the V memory at one time. The transfer cannot be interrupted by a user interrupt.
- The S7-200 SMART CPU transfers all the inputs to the EM DP01 in one message. This transfer cannot be interrupted.
- The EM DP01 sends the inputs to the DP master in one message.

DP master consistency

Consistency in the DP master CPU is not always buffer consistent. The DP master CPUs do not handle the entire DP message as one indivisible object unless it is very small. The DP master CPUs usually move the PROFIBUS data in smaller pieces. They can either move the data to the I/O area or the user can control the movement with DPRD_DAT (Read consistent data for DP devices) and DPWR_DAT (Write consistent data for DP devices) instructions. Using the DPRD_DAT and DPWR_DAT instructions, you obtain the information for one configuration "slot" at a time. Since we allow two configuration slots, it can take two DPRD_DAT instructions to obtain all of the data. Consistency is only guaranteed for each DPRD_DAT instruction.

11.5.1.5 Supported configurations

The following table lists the configurations that are supported by the S7-200 SMART EM DP01 PROFIBUS DP module:

Table 11-2 EM DP01 PROFIBUS DP configuration options

Configuration	Inputs to master	Outputs from master	Data consistency
1	Universal module		Buffer consistency [1]
2	4 bytes	4 bytes	
3	8 bytes	8 bytes	
4	16 bytes	16 bytes	
5	32 bytes	32 bytes	
6	64 bytes	64 bytes	
7	122 bytes	122 bytes	
8	128 bytes	128 bytes	

[1] All EM DP01 configurations are buffer consistent.

We can mix and match any two of these configurations in an EM DP01 configuration. Here are two examples:

- A configuration of 32 bytes input and output plus a configuration of 8 bytes input and output yields a total of 40 input bytes and 40 output bytes.
- A configuration of 122 bytes input and output plus a configuration of 122 bytes input and output yields a total of 244 input bytes and 244 output bytes.

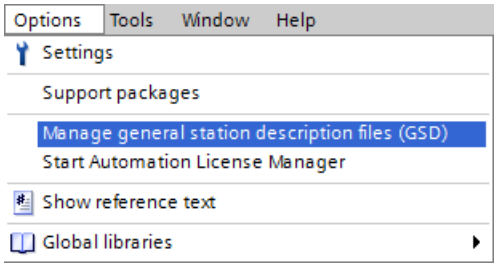
The EM DP01 allows a maximum of 244 input bytes and 244 output bytes. If you use two configurations for the EM DP01, all of the input data is contiguous, and all of the output data is contiguous. Refer to "Example of V memory and I/O address area" (Page 540) for further information.

11.5.1.6 Installing the EM DP01 GSD file

A PROFIBUS GSD file describes the DP device and its capabilities. The programmer uses the GSD file to configure the DP master.

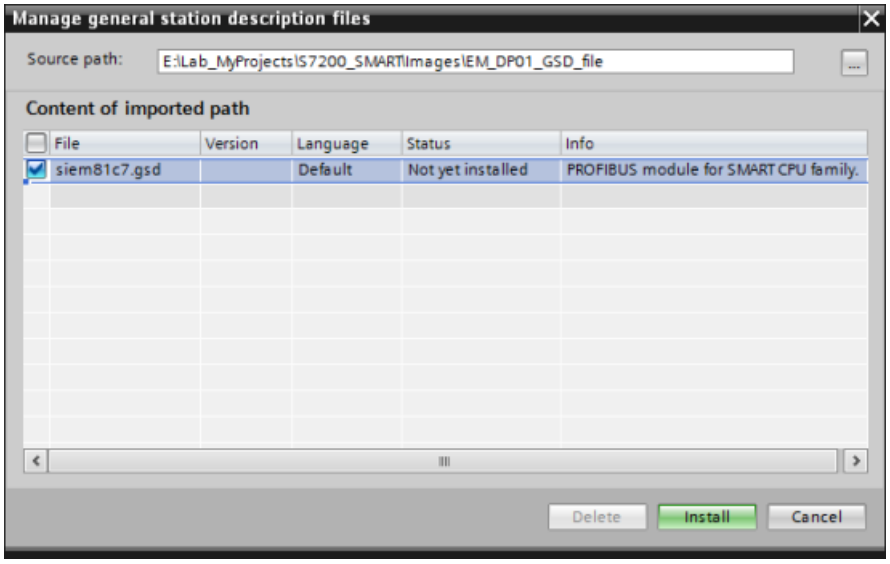
To install the EM DP01 GSD file, follow these steps:

1. Start the TIA Portal software.
2. Create a new project.
3. In the project view, locate the menu bar and select: Options > Manage general station description files (GSD)

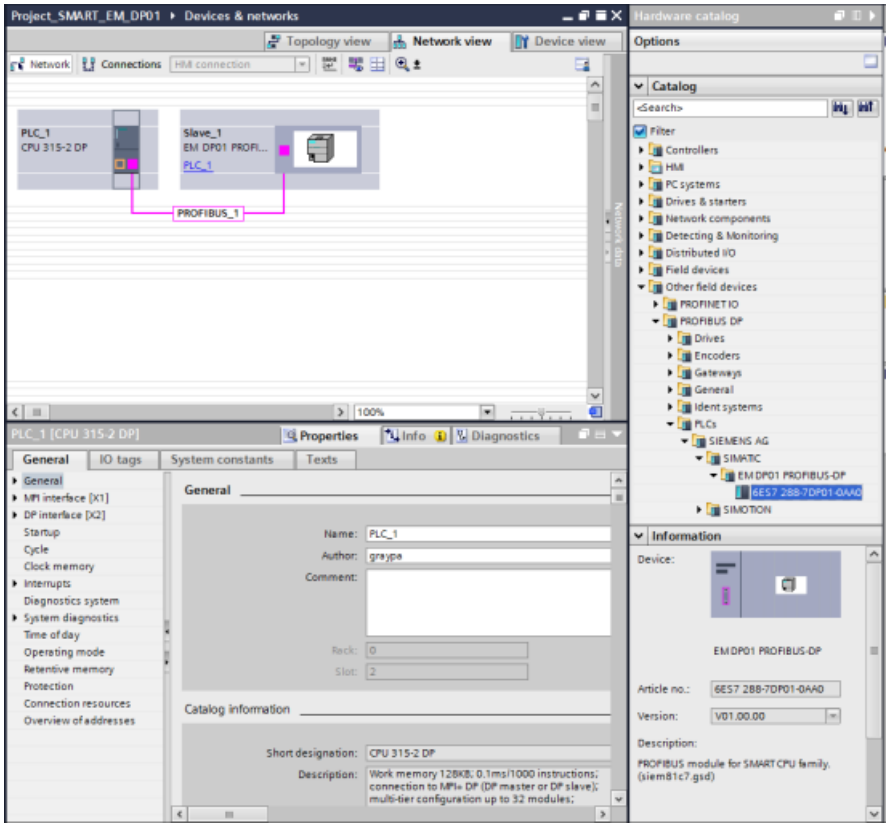


4. In the Source path, using the dropdown button, locate the EM DP01 GSD file that you have previously loaded on your computer.
5. Select the check box for the GSD file line.

6. Click the Install button:



7. This action installs the EM DP01 GSD file in the Hardware catalog as shown in the following figure:



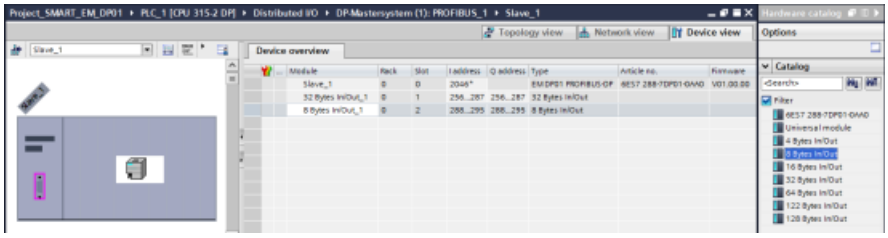
8. Insert a CPU 315-2 DP, for example, as the DP master.
9. Insert the EM DP01 PROFIBUS DP module.
10. Create a PROFIBUS network between the DP master and device as shown in the figure above.

11.5.1.7 Configuring the EM DP01 I/O

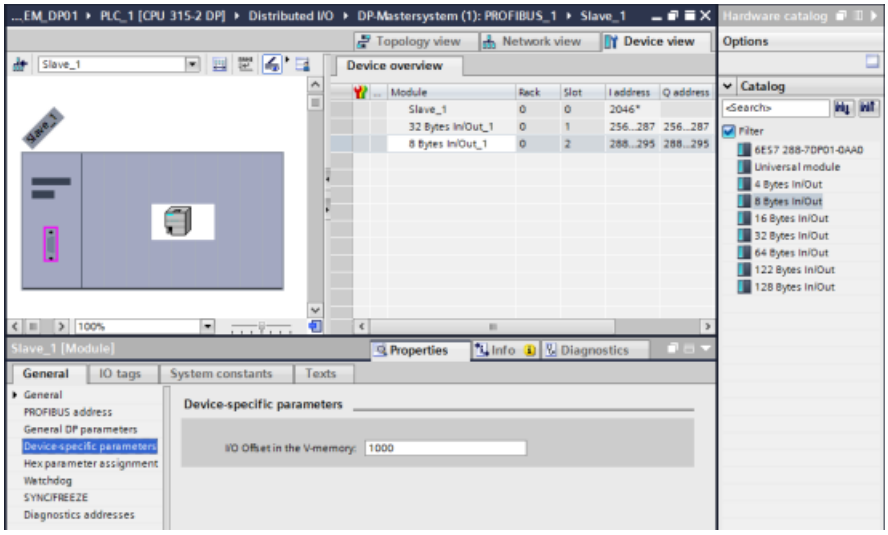
You can configure the EM DP01 I/O by using pre-configured or universal module I/O configuration selections. The EM DP01 configuration allows for two slots so that you can have more than 128 bytes of data transferred between the DP master and the S7-200 SMART CPU. This makes it possible for you to configure the maximum of 244 bytes that PROFIBUS allows. Two possible I/O configuration combinations are shown in the following examples.

32 Bytes In/Out and 8 Bytes In/Out configuration

In this example, slot one contains the "32 Bytes In/Out" pre-configured I/O selection, and slot two contains the "8 Bytes In/Out" pre-configured I/O selection.

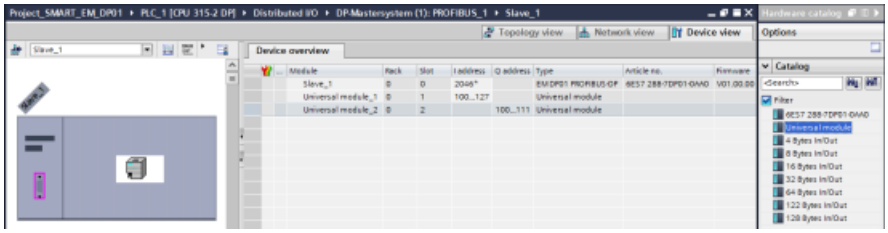


In the "Properties"", "General" tab area navigation, click on "Device-specific parameters" to display the "I/O Offset in the V memory" field. Here, you can assign the starting address of the section of V memory reserved for this operation.



Universal module configuration

In this example, slots one and two contain the "Universal module" I/O selection, and you can configure these two slots with the number of inputs and outputs your application requires (up to a maximum of 244 input bytes and 244 output bytes).



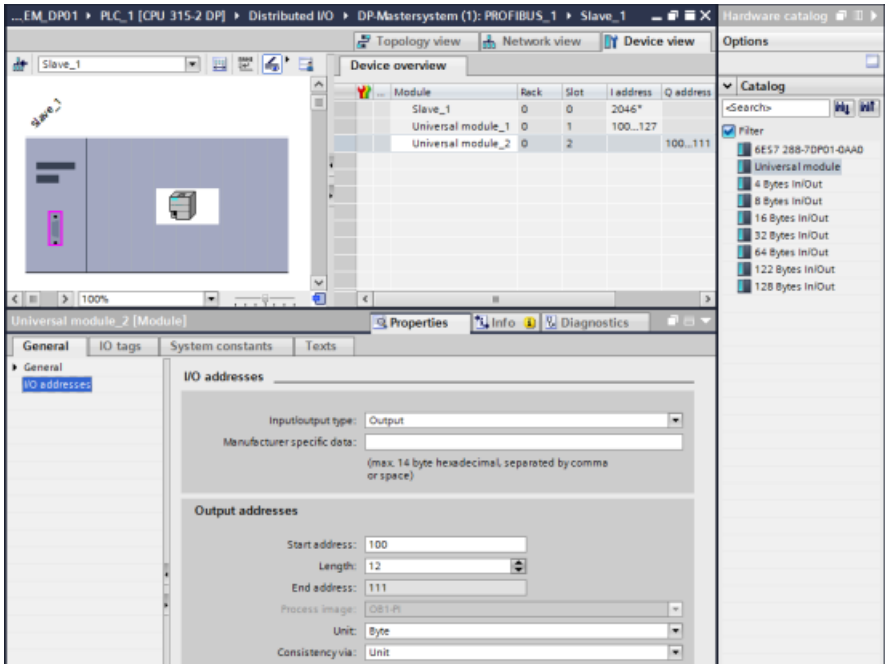
In the "Properties", "General" tab area navigation, click on "I/O addresses" to display the input/output address configuration fields. In the "Input/output type:" field, you must make one of the following selections for the universal module in this slot:

- Input
- Output
- Input/output

Then, you can configure the Input and/or output address ranges for your application.

Note

"Empty slot" is the default selection for the "Input/output type:" field. You must change "Empty slot" to 'Input", "Output", or "Input/output" in order to configure your I/O addresses.

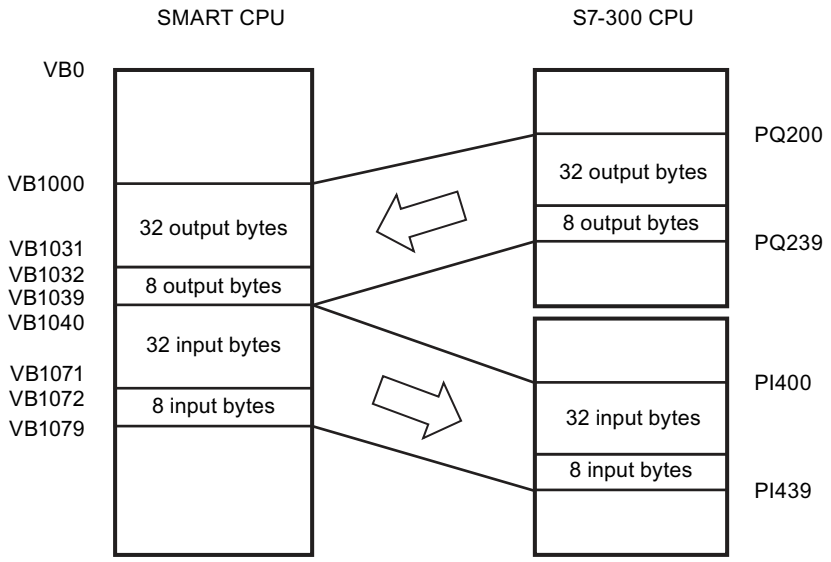


Note

In the examples above, the CPU 315-2 DP is the configured DP master. Depending on the master CPU type, the EM DP01 "Properties" can appear slightly different than those shown here.

11.5.1.8 Example of V memory and I/O address area

The following figure shows an example of the V memory in the S7-200 SMART CPU and the I/O address area of an S7-300 PROFIBUS DP master:



In this example, the DP master has defined an I/O configuration consisting of two slots and a V memory offset of 1000. The example configures the first slot as 32 bytes in and out and the second slot as 8 bytes in and out. The output and input buffers in the S7-200 SMART CPU are both 40 bytes (32 + 8). The output data (from the DP master) buffer starts at V1000; the input data (to the DP master) buffer immediately follows the output buffer and begins at V1040.

All of the output data (all 40 bytes) is treated as one buffer consistent block of data in the EM DP01 and SMART CPU. The output data in the S7-300 is treated with different consistencies depending on whether the user utilizes the I and Q areas or whether they use the DPRD_DAT (Read consistent data for DP devices) and DPWR_DAT (Write consistent data for DP devices) instructions. Even using the DPRD_DAT and DPWR_DAT instructions, the data is only consistent within the 32 byte and 8 bytes blocks. The entire 40 bytes is consistent only if the user manages this by not reading or writing the data in user interrupt blocks.

Note

If you are working with a data unit (consistent data) greater than four bytes, you can use the DPRD_DAT instruction to read the inputs of the DP device and the DPWR_DAT instruction to address the outputs of the DP device. For further information, refer to "Data consistency" and the *System Software for S7-300 and S7-400 System and Standard Functions Reference Manual*.

You can configure the location of the input and output buffers to be anywhere in the V memory of the S7-200 SMART CPU. The default address for the input and output buffers is VB0. The location of the input and output buffers is part of the parameter assignment information that the DP master writes to the S7-200 SMART CPU. You configure the DP master to recognize its DP devices and to write the required parameters and I/O configuration to each of its DP devices.

You configure SIMATIC S7 DP masters using STEP 7 programming software. For detailed information about using this configuration and programming software package, refer to the manuals for these devices. For detailed information about the PROFIBUS network and its components, refer to the *ET 200 Distributed I/O System Manual*.

11.5.1.9 User program considerations

After a DP master successfully configures the EM DP01 PROFIBUS DP module, the EM DP01 and the DP master enter data exchange mode. In data exchange mode, the DP master writes output data to the EM DP01, and the EM DP01 then responds with the most recent S7-200 SMART CPU input data. The EM DP01 continuously updates its inputs from the S7-200 SMART CPU in order to provide the most recent input data to the DP master. The EM DP01 then transfers the output data to the S7-200 SMART CPU. The S7-200 SMART CPU places the output data from the DP master into V memory (the output buffer) starting at the address that the DP master supplied during initialization. The S7-200 SMART CPU takes the input data to the DP master from the V memory locations (the input buffer) immediately following the output data.

The user program in the S7-200 SMART CPU must move the output data from the DP master from the output buffer to the data areas where the program uses it. Likewise, the user program must move the input data to the DP master from the various data areas to the input buffer for transfer to the master.

The S7-200 SMART CPU places the output data from the DP master into V memory immediately prior to the user program portion of the scan. The S7-200 SMART CPU copies the input data (to the DP master) from V memory to the EM DP01 for transfer to the DP master after the user program portion of the scan.

The S7-200 SMART CPU transmits the input data to the DP master on the EM DP01's next data exchange with the DP master.

Status information

There are 50 bytes of special memory (SM) allocated to each expansion module based upon its physical position. The module updates the SM locations corresponding to the modules' relative position to the CPU (with respect to other modules). If it is the first module, it updates SMB1400 through SMB1449. If it is the second module, it updates SMB1450 through SMB1499, and so on. Refer to the table below:

Table 11-3 Special memory bytes SMB1400 to SMB1699

Special memory bytes SMB1400 to SMB1699							
Intelligent module in slot 0	**Intelligent module in slot 1**	**Intelligent module in slot 2**	**Intelligent module in slot 3**	**Intelligent module in slot 4**	**Intelligent module in slot 5**	**Intelligent module in slot 6**	**Intelligent module in slot 7**
SMB1400 to SMB1449	SMB1450 to SMB1499	SMB1500 to SMB1549	SMB1550 to SMB1599	SMB1600 to SMB1649	SMB1650 to SMB1699	SMB1700 to SMB1749	SMB1750 to SMB1799

These SM locations show default values if DP communications have not been established with a DP master. After a DP master has written parameters and I/O configuration to the EM DP01 PROFIBUS DP module, these SM locations show the configuration set by the DP master. You should check the protocol status byte (for example SMB1424 for slot 0) to be sure that the EM DP01 is currently in data exchange mode with the DP master before using the information in the SM locations shown in the following table or data in the V memory buffer.

Note

You cannot configure the EM DP01 PROFIBUS DP I/O buffer sizes or buffer location by writing to SM memory locations. Only the DP master can configure the EM DP01 PROFIBUS DP module for DP operation.

Table 11-4 Special memory bytes for the EM DP01 PROFIBUS DP

Intelligent module in slot 0	**...**	**Intelligent module in slot 7**	**Description**	
SMB1400	...	SMB1750	DP device's station address as set by address switches (0 - 99 decimal)	
SMB1401	...	SMB1751	Address of the DP device's master (0 to 126) (displays 255 if no DP master is attached)	
SMW1402	...	SMW1752	V memory address of the output buffer as an offset from VB (for example, 1000 means VB1000).	
SMB1404	...	SMB1754	Number of bytes of output data	
SMB1405	...	SMB1755	Number of bytes of input data	
SMB1406	...	SMB1756	DP standard protocol status byte	
			Number	Description
			0	DP communications not initiated since power on
			1	Configuration/parameterization error detected

Intelligent module in slot 0	...	Intelligent module in slot 7	Description	
			2	Currently in data exchange mode
			3	Dropped out of data exchange mode
SMB1407 to SMB1449	...	SMB1757 to SMB1799	Reserved - cleared on power up	
Note: SM locations are updated each time the DP device accepts configuration / parameterization information. These locations are updated even if a configuration/parameterization error is detected. The locations are cleared on each power up. Note: This information is also available in the STEP 7-Micro/WIN SMART "PLC information" for the EM DP01. Note: The user program can access this information and use it to process the EM DP01 data.				

11.5.1.10 LED status indicators for the EM DP01 PROFIBUS DP

The EM DP01 PROFIBUS DP module has four status LEDs on the front panel to indicate the operational state of the DP port:

- DIAG LED:
  - Dual color (red / green) LED indicates the operating state and fault status of the EM DP01
  - Flashing red: Upon startup, until the EM DP01 is logged in by the CPU, or if there is a fault in the EM DP01
  - Flashing green: While the EM DP01 is waiting on configuration and parameterization from the S7-200 SMART CPU (immediately after login), or during a firmware update
  - ON green: No fault is present and the EM DP01 is configured
- POWER LED:
  - ON green: If user 24 V DC is applied
  - OFF: No user 24 V DC
- DP ERROR LED:
  - Flashing red: If there is an error in the I/O configuration or parameter information that the DP master writes to the EM DP01
  - ON red: If DP communications are interrupted
  - OFF: No error or data exchange has never been established
- DX MODE LED:
  - OFF: After the S7-200 SMART CPU is turned ON as long as DP communications are not attempted, or if DP communications are interrupted
  - ON green: Once DP communications have been successfully initiated (the EM DP01 has entered Data Exchange Mode with the DP master); remains on until the EM DP01 exits Data Exchange Mode

Note

If DP communications are lost, which forces the EM DP01 to exit Data Exchange Mode, the DX MODE LED turns OFF and the DP ERROR LED turns red. This condition persists until the S7-200 SMART CPU is powered off or Data Exchange Mode is resumed.

The following table summarizes the status indications signified by the EM DP01 status LEDs:

Table 11-5 EM DP01 PROFIBUS DP module status LEDs

Description	POWER LED (Green)	DIAG LED (Dual Red / Green)	DP ERROR LED (Red)	DX MODE LED (Green)
24 V DC user power good	Green			
No 24 V DC user power	Off			
Internal module failure		Red		
Upon startup, until the EM DP01 is logged in by the CPU, or if there is a fault in the EM DP01		Flashing red		
While the EM DP01 is waiting on configuration and parameterization from the S7-200 SMART CPU, or during a firmware update		Flashing green		
No fault is present; EM DP01 is configured		Green		
No DP error			Off	
DP communications interrupted; Data Exchange Mode stopped			Red	
Parameterization / configuration error (from the DP Master)			Flashing red	
Data Exchange Mode inactive, or DP communications interrupted				Off
Data Exchange Mode active				Green

11.5.1.11 Using HMIs and S7-CPUs with the EM DP01

The EM DP01 PROFIBUS DP module can be used as a communications interface to MPI masters, whether or not it is being used as a PROFIBUS DP device. The EM DP01 can provide a connection from an S7-300/400 to the S7-200 SMART using the X_GET/X_PUT functions of the S7-300/400. HMI devices such as the SMART HMI or the TD 400 can be used to communicate with the S7-200 SMART through the EM DP01.

Some devices allow you to select V memory as the memory area in the S7-200 SMART CPU. If V memory is not an option, you should configure the client (CPU or HMI device) to read and write to DB1 to access the V memory in the S7-200 SMART CPU. For example, a X_GET would

need the remote address set to P#DB1.DBX100.0 BYTE 20 to read 20 bytes of V memory starting at VB100.

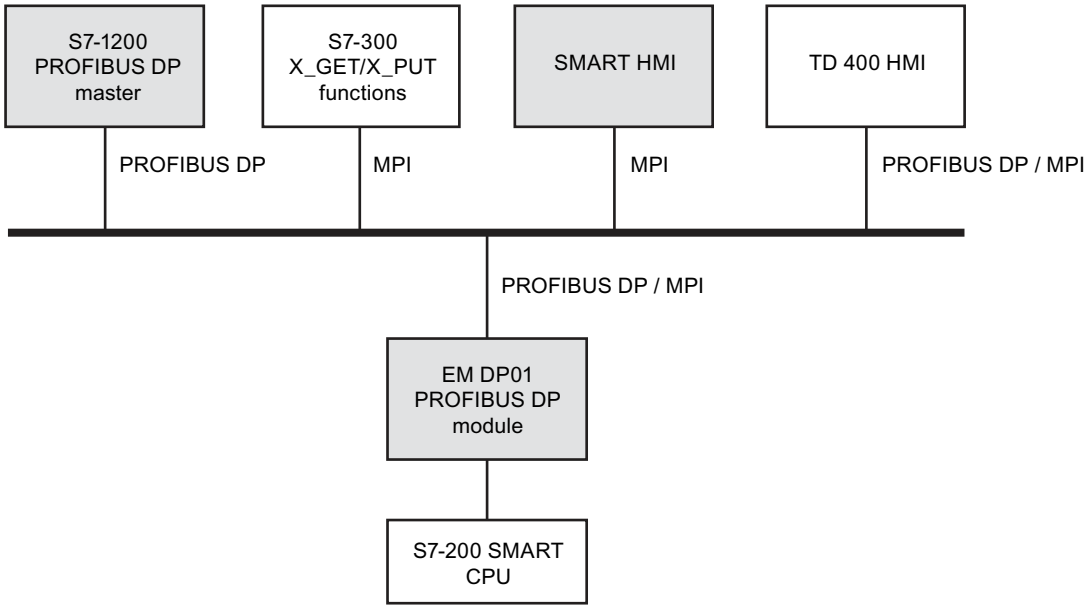
Note

An S7-1200 PROFIBUS DP master cannot access an S7-200 SMART CPU using GET/PUT functions. The S7-1200 DP master can still access the S7-200 SMART CPU using PROFIBUS Data Exchange Mode.

When the EM DP01 PROFIBUS DP module is used for MPI communications, the address parameter of the XGET/XPUT functions must be set to the address of the EM DP01 (address switches). MPI messages sent to the EM DP01 are passed on to the S7-200 SMART CPU.

Messages from MPI and HMI devices are (serviced during)/(subject to) the communication background time in the S7-200 SMART CPU. The communication background time can be increased to provide faster responses to the MPI and HMI requests.

A maximum of six connections (six devices) in addition to the DP master can be connected to the EM DP01. In order for the EM DP01 to communicate with multiple masters, all masters must be operating at the same baud rate. Refer to the figure below for one possible network configuration:



11.5.1.12 Device database file: GSD

Different PROFIBUS devices have different performance characteristics. These characteristics differ with respect to functionality (for example, the number of I/O signals and diagnostic messages) or bus parameters, such as transmission speed and time monitoring. These parameters vary for each device type and vendor and are usually documented in a technical manual. To help you achieve a simple configuration of PROFIBUS, the performance characteristics of a particular device are specified in an electronic data sheet called a device database file, or GSD file. Configuration tools based upon GSD files allow simple integration of devices from different vendors in a single network.

The GSD device database file provides a comprehensive description of the characteristics of a device in a precisely defined format. These GSD files are prepared by the vendor for each type of device and made available to the PROFIBUS user. The GSD file allows the configuration system to read in the characteristics of a PROFIBUS device and use this information when configuring the network.

If your version of software does not include a configuration file for the EM DP01, you can access the latest GSD file (SIEM81C7.GSD) from Siemens Customer Support.

If you are using a non-Siemens master device, refer to the documentation provided by the manufacturer on how to configure the master device by using the GSD file.

GSD file for the EM DP01 PROFIBUS-DP 6ES7288-7DP01-0AA0

Table 11-6 General parameters

Parameters	Values
#Profibus_DP	
GSD_Revision	= 5
Vendor_Name	= "Siemens"
Model_Name	= "EM DP01 PROFIBUS-DP"
Revision	= "V01.00.00"
Ident_Number	= 0x81C7
Protocol_Ident	= 0
Station_Type	= 0
FMS_supp	= 0
Hardware_Release	= 1
Software_Release	= "V01.00.00"
;	
9.6_supp	= 1
19.2_supp	= 1
45.45_supp	= 1
93.75_supp	= 1
187.5_supp	= 1
500_supp	= 1
1.5M_supp	= 1
3M_supp	= 1
6M_supp	= 1
12M_supp	= 1
;	
MaxTsdr_9.6	= 40
MaxTsdr_19.2	= 40
MaxTsdr_45.45	= 40
MaxTsdr_93.75	= 40
MaxTsdr_187.5	= 40
MaxTsdr_500	= 40

Parameters	Values
MaxTsdr_1.5M	= 40
MaxTsdr_3M	= 50
MaxTsdr_6M	= 100
MaxTsdr_12M	= 200
;	
Redundancy	= 0
Repeater_Ctrl_Sig	= 2
24V_Pins	= 2
Implementation_Type	= "DPC31"
Bitmap_Device	= "EM_DP01N"

Table 11-7 Slave-Specification

Parameters	Values
OrderNumber	= "6ES7 288-7DP01-0AA0"
Periphery	= "SIMATIC S7"
Info_Text	= "PROFIBUS module for SMART CPU family."
Slave_Family	= 10@TdF@SIMATIC
Freeze_Mode_supp	= 1
Sync_Mode_supp	= 1
Set_Slave_Add_Supp	= 0
Auto_Baud_supp	= 1
Min_Slave_Intervall	= 1
Fail_Safe	= 0
;	
Modular_Station	= 1
Max_Module	= 2
Modul_Offset	= 0
;	
Max_Input_len	= 244
Max_Output_len	= 244
Max_Data_len	= 488
Max_Diag_Data_Len	= 6

Table 11-8 DPV1 support

Parameters	Values
DPV1_Slave	= 1
C1_Read_Write_supp	= 1
C2_Read_Write_supp	= 1
C1_Max_Data_Len	= 240

Parameters	Values
C2_Max_Data_Len	= 240
C1_Response_Timeout	= 100
C2_Response_Timeout	= 100
C1_Read_Write_required	= 0
C2_Read_Write_required	= 0
C2_Max_Count_Channels	= 6
Max_Initiate_PDU_Length	= 64
Ident_Maintenance_supp	= 1
DPV1_Data_Types	= 0
WD_Base_1ms_supp	= 0
Check_Cfg_Mode	= 0
Publisher_supp	= 0

Table 11-9 UserPrmData-Definition

Parameters	Values
ExtUserPrmData	= 1 "I/O Offset in the V-memory"
Unsigned16 0 0-20479	
EndExtUserPrmData	

Table 11-10 UserPrmData: Length and Preset

Parameters	Values
Max_User_Prm_Data_Len	= 5
Ext_User_Prm_Data_Const (0)	= 0x00,0x00,0x00,0x00,0x00
Ext_User_Prm_Data_Ref (3)	= 1

Table 11-11 Module Definition List

Parameters	Values
Module 20 EndModule	= " 4 Bytes In/Out" 0xF1
Module 21 EndModule	= " 8 Bytes In/Out" 0xF3
Module 22 EndModule	= " 16 Bytes In/Out" 0xF7
Module 23 EndModule	= " 32 Bytes In/Out" 0xFF

Parameters	Values
Module 24 EndModule	= " 64 Bytes In/Out" 0xC0, 0xDF, 0xDF
Module 25 EndModule	= "122 Bytes In/Out" 0xC0, 0xFC, 0xFC
Module 26 EndModule	= "128 Bytes In/Out" 0xC0, 0xFF, 0xFF

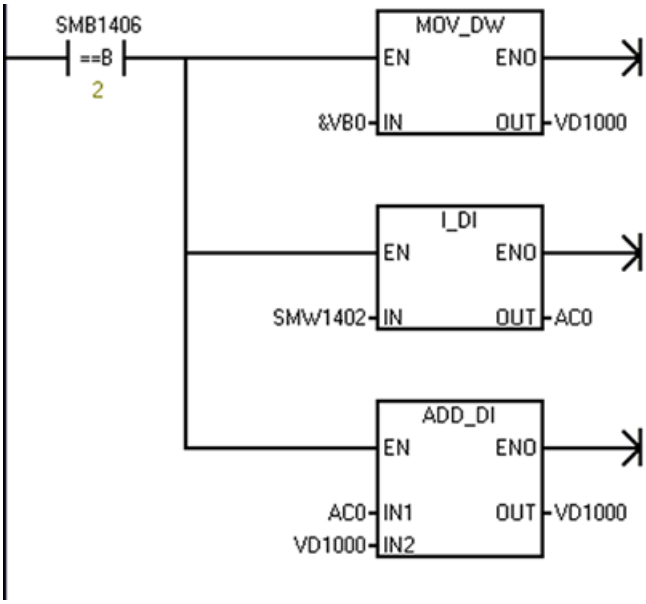
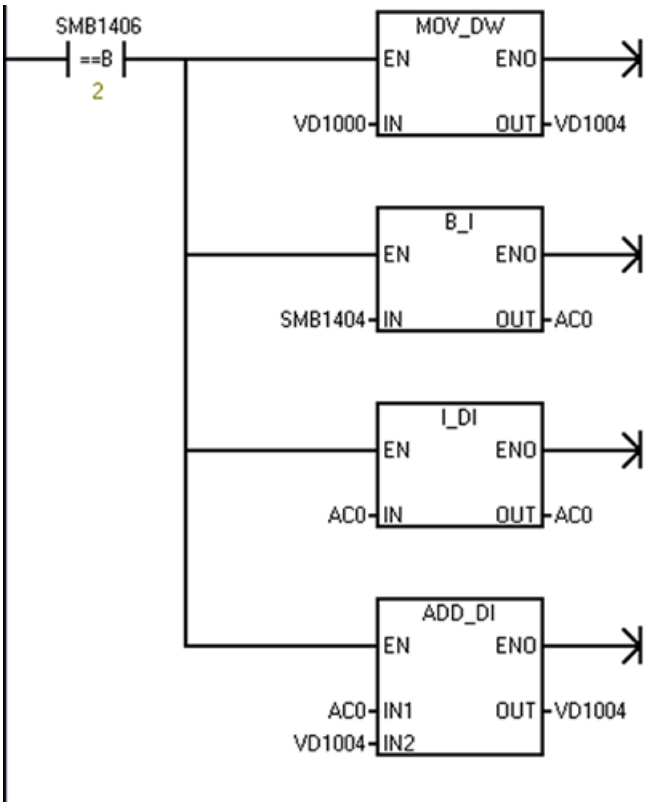
11.5.1.13 PROFIBUS DP communications to a CPU example program

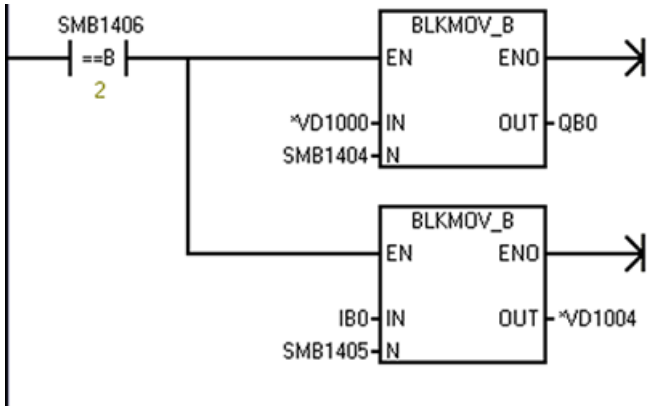
An example program for the PROFIBUS DP module in slot 0 for a CPU that uses the DP port information in SM memory is shown below. The program determines the location of the DP buffers from SMW1402 and the sizes of the buffers from SMB1404 and SMB1405. This information is used to copy the data in the DP output buffer to the process image output register of the CPU. Similarly, the data in the process image input register of the CPU are copied into the V memory input buffer.

In the following example program for a DP module in position 0, the DP configuration data in the SM memory area provides the configuration of the DP device. The program uses the following data:

SMB1406	DP Status
SMB1401	Master Address
SMW1402	V memory offset of outputs
SMB1404	Number of bytes of output data
SMB1405	Number of bytes of input data
VD1000	Output Data Pointer
VD1004	Input Data Pointer

Table 11-12 Example: Configuring DP communications to an S7-200 SMART CPU

LAD/FBD	Description	STL
Network 1:  	Calculate the Output data pointer. If in data exchange mode: 1. Output buffer is an offset from VB0. 2. Convert V memory offset to double integer. 3. Add to VB0 address to get output data pointer.	**LDB= SMB224, 2** **MOVD &VB0, VD1000** **ITD SMW226, AC0** **+D AC0, VD1000**
Network 2:  	Calculate the Input data pointer. If in data exchange mode: 1. Copy the output data pointer. 2. Get the number of output bytes. 3. Add to output data pointer to get starting input data pointer.	**LDB= SMB224, 2** **MOVD VD1000, VD1004** **BTI SMB228, AC0** **ITD AC0, AC0** **+D AC0, VD1004**

LAD/FBD	Description	STL
Network 3:  	Transfer Master outputs to CPU outputs. Copy CPU inputs to the Master inputs. If in data exchange mode: 1. Copy Master outputs to CPU outputs. 2. Copy CPU inputs to Master inputs.	**LDB= SMB224, 2** **BMB *VD1000, QB0, VB1008** **BMB IB0, *VD1004, VB1009**

11.5.1.14 Reference to the EM DP01 PROFIBUS DP module technical specifications

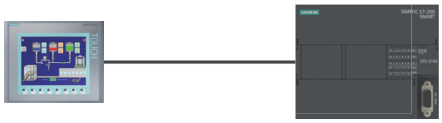
For further information on the EM DP01 PROFIBUS DP module, refer to the "EM DP01 PROFIBUS DP module" (Page 1048) technical specifications.

11.6 RS485

11.6.1 PPI protocol

Definition

PPI is a master-slave protocol: the master devices send requests to the slave devices, and the slave devices respond. See the following figure. Slave devices do not initiate messages, but wait until a master sends them a request or polls them for a response.



HMI: Master S7-200 SMART V3: Slave

Masters communicate to slaves by means of a shared connection which is managed by the PPI protocol. PPI does not limit the number of masters that can communicate with any one slave; however, you cannot install more than 32 masters on the network.

Note

After downloading PPI Fix IP, the changes are applied immediately without requiring a system power cycle (power off and power on).

PPI protocol and S7-200 SMART V3 CPUs

PPI Advanced allows network devices to establish a logical connection between the devices. With PPI Advanced, there are a limited number of connections supplied by each device. See the following table for the number of connections supported by the S7-200 SMART V3 CPU.

All S7-200 SMART V3 CPUs support both PPI and PPI Advanced protocols.

Table 11-13 Number of connections for the S7-200 SMART V3 CPU

Module	Baud rate	Connections
RS485 port	9.6 Kbps, 19.2 Kbps, or 187.5 Kbps	5
RS485/RS232 signal board	9.6 Kbps, 19.2 Kbps, or 187.5 Kbps	4

11.6.2 Baud rate and network address

11.6.2.1 Definition of baud rate and network address

Baud rate

The speed that data is transmitted across the network is the baud rate, which is typically measured in units of kilobits (Kbps) or megabits (Mbps). The baud rate measures how much data can be transmitted within a given amount of time. For example, a baud rate of 19.2 Kbps describes a transmission rate of 19,200 bits per second.

Every device that communicates over a given network must be configured to transmit data at the same baud rate. Therefore, the fastest baud rate for the network is determined by the slowest device connected to the network.

The following table lists the baud rates supported by the S7-200 SMART CPU.

Table 11-14 Baud rate supported by the S7-200 SMART CPU

Network	Baud rate
PPI protocol	9.6 Kbps, 19.2 Kbps, and 187.5 Kbps only
Freeport Mode	1.2 Kbps to 115.2 Kbps

Network address

The network address is a unique number that you assign to each device on the network. The unique network address ensures that the data is transferred to or retrieved from the correct device. The S7-200 SMART CPU supports network addresses from 0 to 126. The following table lists the default (factory) settings for the S7-200 SMART devices.

Table 11-15 Default addresses for S7-200 SMART devices

S7-200 SMART device	Default address
STEP 7-Micro/WIN SMART	0
HMI	1
S7-200 SMART CPU	2

11.6.2.2 Setting the baud rate and network address for the S7-200 SMART CPU

Introduction

To communicate over the RS485 network with STEP 7-Micro/WIN SMART or SIMATIC HMIs (Page 477), you must configure the RS485 network address and baud rate for the S7-200 SMART CPU.

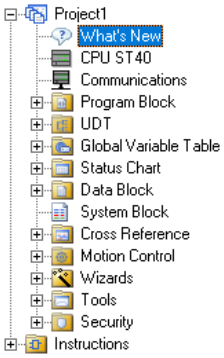
The RS485 port network address must be unique for all devices on the RS485 network, and the RS485 port baud rate must be the same as the other devices on the RS485 network. The default RS485 port network address is 2, and the default RS485 port baud rate for each CPU port is 9.6 Kbps.

The system block of the CPU stores the RS485 port network address and baud rate. After you select the parameters for the CPU, you must download the project to the S7-200 SMART CPU.

Procedure

To access the "System Block" dialog, click one of the following:

System block button in the navigation bar

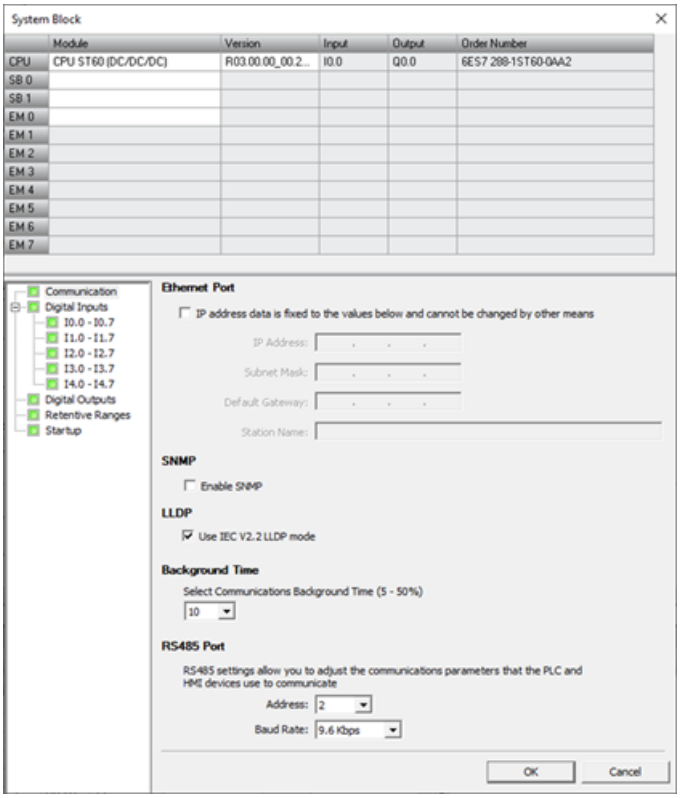


System block in the project tree

System block in the Component drop-down list in the Windows area of the View menu ribbon strip

After you select the "System Block" dialog, you perform the following steps:

1. Select the network address and baud rate for the RS485 port.
2. Download the system block to the CPU.



Note

Freeport protocol baud rates are set using SM memory.

11.6.3 Sample RS485 network configurations

11.6.3.1 Single-master PPI networks

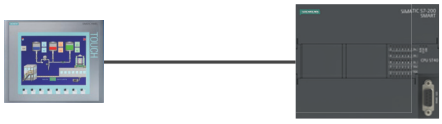
Introduction

The following network configurations are possible using only S7-200 SMART devices:

- Single-master PPI networks
- Multi-master and multi-slave PPI networks
- Complex PPI networks

Single-master PPI networks

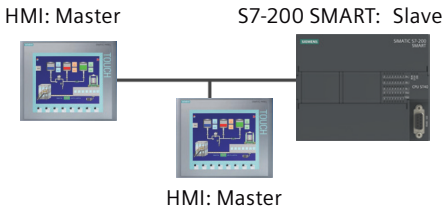
In the sample network in the figure below, a human-machine interface (HMI) device (for example, a TD400C, TP, or KP) is the network master:



In the sample network, the CPU is a slave that responds to requests from the master.

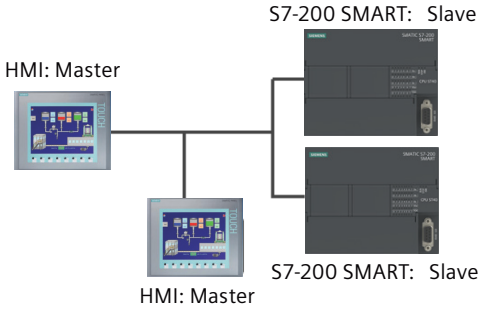
11.6.3.2 Multi-master and multi-slave PPI networks

The following figure shows a sample network of multiple masters with one slave. The HMI devices share the network.



The HMI devices are masters and must have separate network addresses. The S7-200 SMART CPU is a slave.

The following figure shows a PPI network with multiple masters communicating with multiple slaves. In this example, the HMI can request data from any CPU slave.



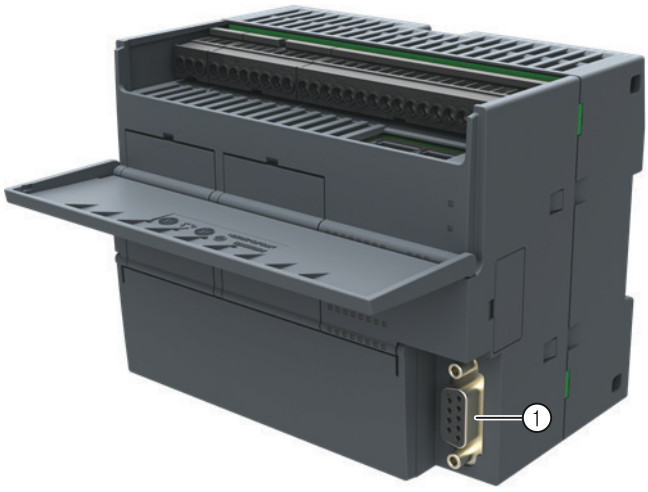
All devices (masters and slaves) have different network addresses. The S7-200 SMART CPUs are slaves.

11.6.4 Assigning RS485 addresses

11.6.4.1 Configuring or changing an RS485 address for a CPU or device in your project

You must enter the following information for each S7-200 SMART CPU that is attached to your RS485 network:

- RS485 address: Each CPU or device must have an RS485 address. The CPU or device uses this address to deliver data over the network.
- Baud rate: The speed that data is transmitted across the network is the baud rate, which is typically measured in units of Kbps or Mbps. The baud rate measures how much data can be transmitted within a given amount of time (for example, a transmission rate of 19.2 Kbps).



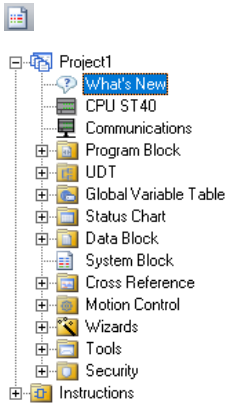
① RS485 port

You must configure or change RS485 network information for the onboard RS485 port of a CPU or device in the "System Block" dialog and download the configuration to the CPU.

Configuring RS485 network information in the System Block dialog

RS485 network information configuration or changes done in the system block are part of the project and do not become active until you download your project to the CPU.

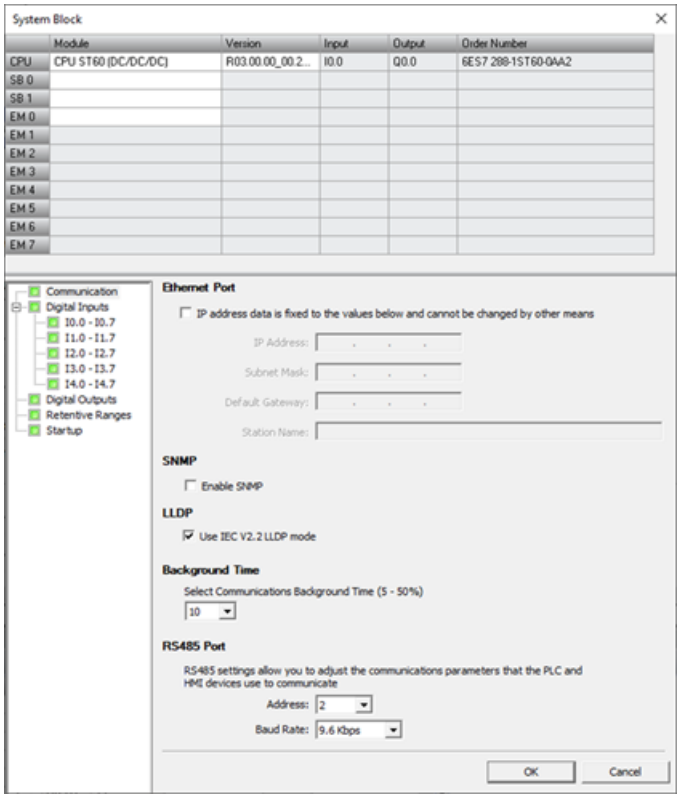
To access this dialog, perform one of the following:



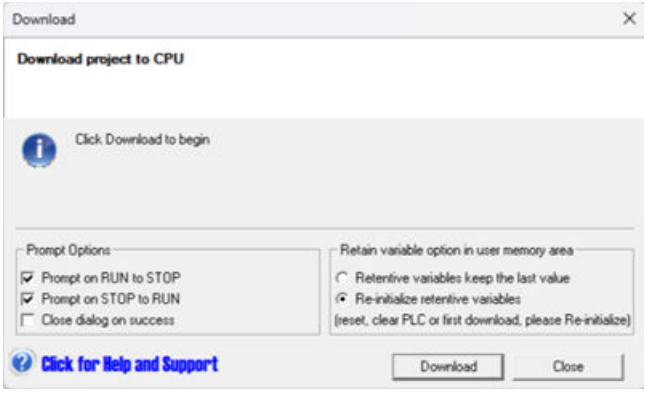
- In the navigation bar, click the "System Block" button.
- In the Project tree, select the "System Block" node, then press Enter; or double-click the "System Block" node.

Enter or change the following access information:

- RS485 port address
- RS485 port baud rate



After completing the RS485 network configuration, download the project to the CPU.



All CPUs and devices that have valid RS485 port addresses are displayed in the "Communications" dialog.

You can access CPUs in one of two ways:

- "Found CPUs": CPUs located on the RS485 network
- "Added CPUs": CPUs on the RS485 network (for example, enter the RS485 network address of a CPU directly that is on the RS485 network)

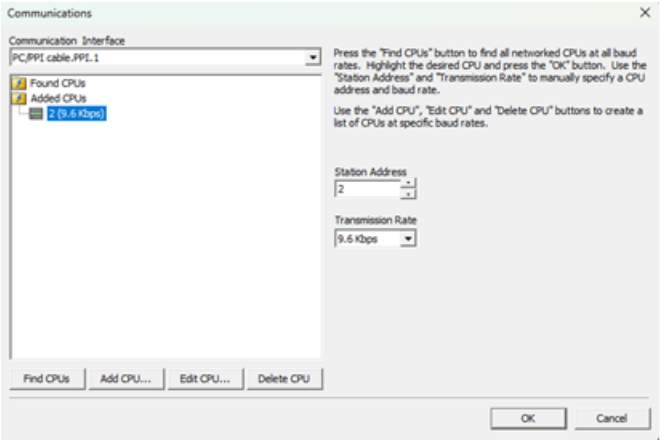
For "Found CPUs" (CPUs located on your local network), use the "Communications" dialog to connect with your CPU:

- Click the "Communication Interface" dropdown list, and select "PC/PPI cable.PPI.1" for your RS485 network.
- Click the "Find CPUs" button to display all operational CPUs ("Found CPUs") on the RS485 network. All CPUs default their RS485 network settings to address 2 and 9.6 Kbps.
- Highlight a CPU, and then click "OK".

Note

You can open multiple copies of STEP 7-Micro/WIN SMART on a computer. Be aware that when you open a second copy of STEP 7-Micro/WIN SMART or use the "Find CPUs" button in either copy, the communication connection to the CPU in your first/other copy of STEP 7-Micro/WIN SMART might be disconnected.

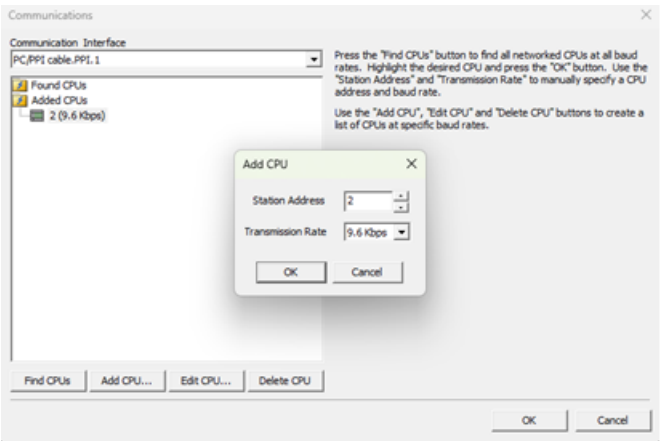
Once the connection is established from the first instance (even if password protected), the subsequent instances from that device can also connect to the CPU.



For "Added CPUs" (CPUs on the RS485 network), use the "Communications" dialog to connect with your CPU:

- Click the "Communication Interface" dropdown list, and select "PC/PPI cable.PPI.1" for your RS485 network.
- Click the "Add CPU" button, and enter the following access information for a CPU that you wish to access directly on the RS485 network:
  - RS485 network address
  - RS485 network baud rate

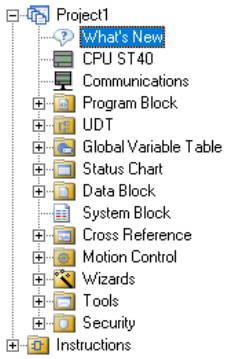
  You can add multiple CPUs on the RS485 network. As always, STEP 7-Micro/WIN SMART communicates with one CPU at a time. All CPUs default their RS485 network settings to address 2 and 9.6 Kbps.
- Highlight a CPU, and then click "OK".



11.6.4.2 Searching for CPUs and devices on your RS485 network

You can search for and identify the S7-200 SMART CPUs that are attached to your RS485 network in the "Communications" dialog. To access this dialog, click one of the following:

"Communications" button in the navigation bar



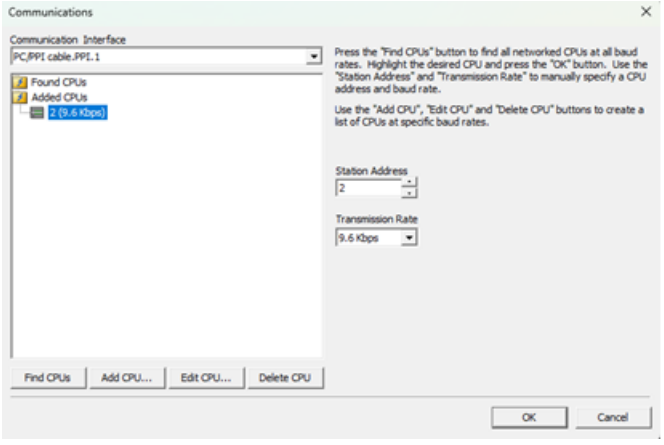
"Communications" in the project tree

"Communications" from the "Component" drop-down list in the Windows area of the "View" menu ribbon strip

The "Communications" dialog autodetects all connected and available S7-200 SMART CPUs on a given RS485 network by creating a lifelist. (See the figure below.) After selecting a CPU, the dialog lists the following detailed information about the CPU:

- RS485 port address
- RS485 network baud rate

The STEP 7-Micro/WIN SMART project includes all added CPUs. However, opening a STEP 7-Micro/WIN SMART project does not automatically select an RS485 network address or establish a connection to a CPU. Every time you create a new or open an existing STEP 7-Micro/WIN SMART project, you must go to the Communications dialog to establish a connection to a CPU. The Communications dialog will show the last selected CPU.



Note

When you press the "Find CPUs" button, the EM DP01 PROFIBUS DP module autobauds and displays in the "Found CPUs " folder at 9.6 Kbps. If you want to communicate through a DP01 modules at a higher baud rate, you must press the "Add CPU" button and add the EM DP01 module using the module's network address and specify a baud rate such as 187.5 Kbps.

You must power cycle the DP01 module to connect at the new baud rate.

11.6.5 Building your network

11.6.5.1 General guidelines

Always install appropriate surge suppression devices for any wiring that could be subject to lightning surges.

Avoid placing low-voltage signal wires and communication cables in the same wire tray with AC wires and high-energy, rapidly switched DC wires. Always route wires in pairs, with the neutral or common wire paired with the hot or signal-carrying wire.

The communication port of the S7-200 SMART CPU is not isolated. Consider using an RS485 repeater to provide isolation for your network.

NOTICE
Avoiding unwanted current flow Interconnecting equipment with different reference potentials can cause unwanted currents to flow through the interconnecting cable. These unwanted currents can cause communications errors or can damage equipment. Be sure all equipment that you are about to connect with a communications cable either shares a common circuit reference or is isolated to prevent unwanted current flows.

11.6.5.2 Determining the distances, transmission rates, and cable lengths for your network

As shown in the following table, the maximum length of a network segment is determined by two factors: isolation (using an RS485 repeater) and baud rate.

Isolation is required when you connect devices at different ground potentials. Different ground potentials can exist when grounds are physically separated by a long distance. Even

over short distances, load currents of heavy machinery can cause a difference in ground potential.

Table 11-16 Maximum length for a network cable

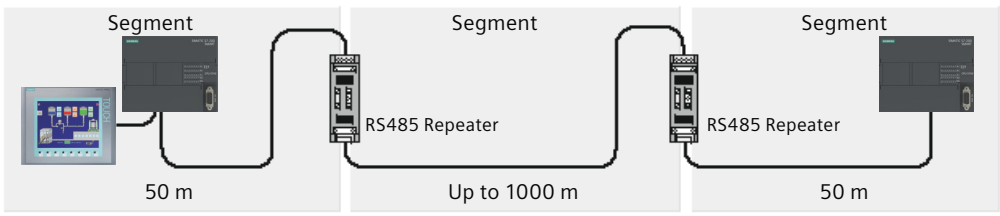
Baud rate	Non-isolated CPU port[1]	CPU port with repeater
9.6 Kbps to 187.5 Kbps	50 m	1,000 m
500 Kbps	Not supported	400 m
1 Mbps to 1.5 Mbps	Not supported	200 m
3 Mbps to 12 Mbps	Not supported	100 m
[1] The maximum distance allowed without using an isolator or repeater is 50 m. You measure this distance from the first node to the last node in the segment.		

11.6.5.3 Repeaters on the network

An RS485 repeater provides bias and termination for the network segment. You can use a repeater for the following purposes:

- **To increase the length of a network**
  Adding a repeater to your network allows you to extend the network another 50 m. If you connect two repeaters with no other nodes in between, (as shown in the figure below), you can extend the network to the maximum cable length for the baud rate. You can use up to 9 repeaters in series on a network, but the total length of the network must not exceed 9600 m.
- **To add devices to a network**
  Each segment can have a maximum of 32 devices connected up to 50 m at 9.6 Kbps. Using a repeater allows you to add another segment (32 devices) to the network.
- **To electrically isolate different network segments**
  Isolating the network improves the quality of the transmission by separating the network segments which may be at different ground potentials.

A repeater on your network counts as one of the nodes on a segment, even though it is not assigned a network address. The following is a sample network with repeaters.



11.6.5.4 Specifications for RS485 cable

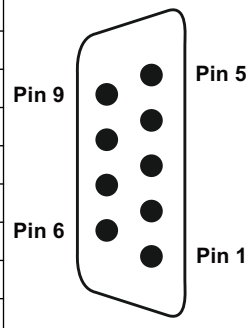
S7-200 SMART CPU networks use the RS485 standard on twisted pair cables. The following table lists the specifications for the network cable. You can connect up to 32 devices on a network segment.

Specifications	Description
Cable type	Shielded, twisted pair
Loop resistance	≤115 Ω/km
Effective capacitance	30 pF/m
Nominal impedance	Approximately 135 Ω to 160 Ω (frequency = 3MHz to 20 MHz)
Attenuation	0.9 dB/100 m (frequency=200 kHz)
Cross-sectional core area	0.3 mm^2 to 0.5 mm^2
Cable diameter	8 mm +0.5 mm

11.6.5.5 Connector pin assignments

The RS485 communication port on the S7-200 SMART CPUs is RS485-compatible on a nine-pin subminiature D connector, in accordance with the PROFIBUS standard as defined in the European Standard EN 50170. The following table shows the connector that provides the physical connection for the communication port and describes the communication port pin assignments.

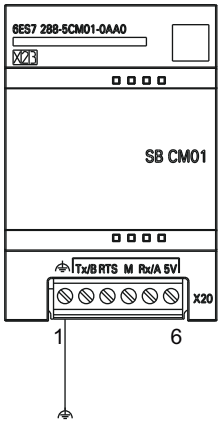
Table 11-17 Pin assignments for the S7-200 SMART CPU integrated RS485 port (Port 0)

Pin number	Connector	Signal	Integrated RS485 port (Port 0)
1		Shield	Chassis ground
2		24 V Return	Logic common
3		RS485 Signal B	RS485 Signal B
4		Request-to-Send	RTS (TTL)
5		5 V Return	Logic common
6		+5 V	+5 V output, 100 Ω series resistor
7		+24 V	+24 V output
8		RS485 Signal A	RS485 Signal A
9		Not applicable	Programmer detection (input) [1]
Connector shell		Shield	Chassis ground

[1] The CPU utilizes pin 9 of the RS485 connector to detect when a USB-PPI cable is connected. The check for USB-PPI cables is only done on the CRs models. The ST and SR models ignore the state of pin 9. Ensure that any cables used for Freeport do not connect to pin 9 for the CRs models.

The CM01 signal board is RS485-compatible. The following table shows the connector that provides the physical connection for the signal board and describes the pin assignments.

Table 11-18 Pin assignments for the S7-200 SMART CM01 Signal Board (SB) port (Port 1)

Pin number	Connector	Signal	CM01 Signal Board (SB) port (Port 1)
1		Ground	Chassis ground
2		Tx/B	RS232-Tx/RS485-B
3		Request-to-Send	RTS (TTL)
4		M-ground	Logic common
5		Rx/A	RS232-Rx/RS485-A
6		+5 V DC	+5 V, 100 Ω series resistor

11.6.5.6 Biasing and terminating the network cable

Siemens provides two types of network connectors that you can use to easily connect multiple devices to a network:

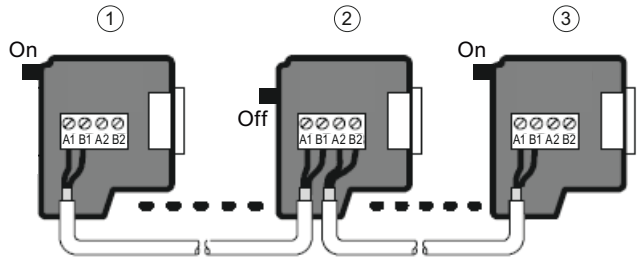
- Standard network connector
- Connector that includes a port which allows you to connect an HMI device to the network without disturbing any existing network connections

The programming port connector passes all signals (including the power pins) from the S7-200 SMART CPU through to the programming port, which is especially useful for connecting devices that draw power from the S7-200 SMART CPU (such as a TD 400C).

Both connectors have two sets of terminal screws to allow you to attach the incoming and outgoing network cables. Both connectors also have switches to bias and terminate

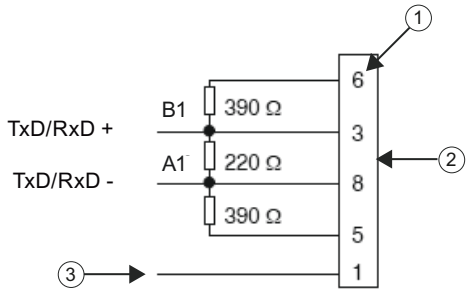
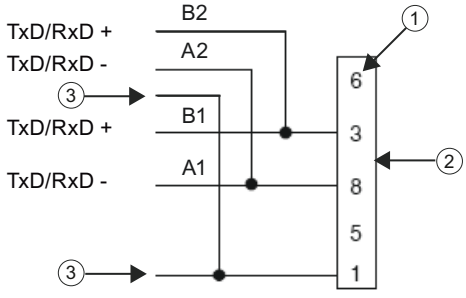
the network selectively. The following shows typical biasing and termination for the cable connectors.

Table 11-19 Biasing and termination for cable connectors

Cable must be terminated and biased at both ends. Bare shielding: Approximately 12 mm (1/2 in) must contact the metal guides of all locations.


① Switch position = On: Terminated and biased
② Switch position = Off: No termination or bias
③ Switch position = On: Terminated and biased

Table 11-20 Termination and bias switch positions

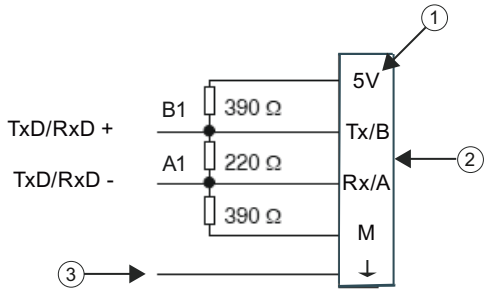
Switch position = On: Termination and biased	Switch position = Off: No termination or bias
	

① Pin number
② Network connector
③ Cable shield

11.6.5.7 Biasing and terminating the CM01 signal board

You can use the CM01 signal board to easily connect multiple devices to a network.

The signal board passes all signals from the S7-200 SMART CPU through to the programming port.



① Terminal name
② Terminal block
③ Cable shield

11.6.5.8 Using HMI devices on your RS485 network

Introduction

The S7-200 SMART CPU supports many types of RS485 HMI devices from Siemens and also from other manufacturers. While some of these HMI devices (such as the TD400C) do not allow you to select the communication protocol used by the device, other devices (such as the KP and TP product lines) allow you to select the communication protocol for that device.

Guidelines

If your HMI device allows you to select the communication protocol, consider the following guideline. For an HMI device connected to the communication port of the CPU, with no other devices on the network, select the PPI protocol for the HMI device.

For more information about how to configure the HMI device, refer to the specific manual for your device (see the following table). These manuals are included in the STEP 7-Micro/WIN SMART V3 documentation CD.

Table 11-21 RS485 HMI devices supported by the S7-200 SMART CPU

HMI	Configuration software
TD400C	Text Display wizard (part of STEP 7-Micro/WIN SMART V3)
KTP600 DP	WinCC flexible
KTP1000 DP	WinCC flexible

11.6.6 Freeport mode

11.6.6.1 Creating user-defined protocols with Freeport mode

Introduction

Freeport mode allows your program to control the communication port of the S7-200 SMART CPU. You can use Freeport mode to implement user-defined communication protocols to communicate with many types of intelligent devices. Freeport mode supports both ASCII and binary protocols.

Using Freeport mode

To enable Freeport mode, you use special memory bytes SMB30 (for the Integrated RS485 port (Port 0)) and, if your CPU model supports it, SMB130 (for the CM01 Signal Board (SB) port (Port 1), SMB31 (for the CM01 Signal Board (SB) port (Port 2)). Your program uses the following to control the operation of the communication port:

- **Transmit instruction (XMT) and the transmit interrupt**:
  The Transmit instruction allows the S7-200 SMART CPU to transmit up to 255 characters from the COM port. The transmit interrupt notifies your program in the CPU when the transmission has been completed.
- **Receive character interrupt**:
  The receive character interrupt notifies the user program that a character has been received on the COM port. Your program can then act on that character, based on the protocol being implemented.
- **Receive instruction (RCV)**:
  The Receive instruction receives the entire message from the COM port and then generates an interrupt for your program when the message has been completely received. You use the SM memory of the CPU to configure the Receive instruction for starting and stopping the receiving of messages, based on defined conditions. The Receive instruction allows your program to start or stop a message based on specific characters or time intervals. Most protocols can be implemented with the Receive instruction, instead of using the more cumbersome receive-character interrupt method.

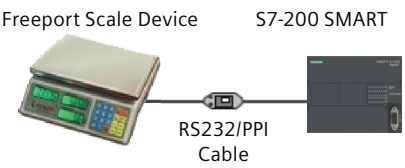
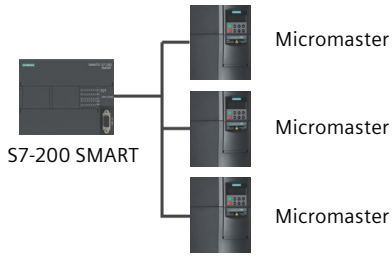
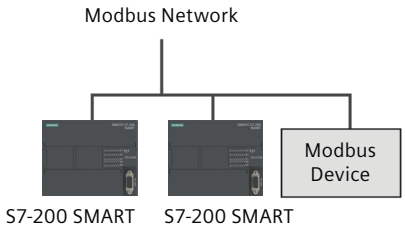
Freeport mode is active only when the CPU is in RUN mode. Setting the CPU to STOP mode halts all Freeport communication, and the communication port then reverts to the PPI protocol with the settings which were configured in the system block of the CPU.

Note

Attaching a USB-PPI cable to the CPU's RS485 port forces the CPU to exit Freeport mode and enable PPI mode. This allows STEP 7-Micro/WIN SMART to regain control of the the CPU.

If you have attached a USB-PPI cable to the CPU's RS485 port, the CPU cannot enable Freeport. The CPU will not automatically restart Freeport when you remove the USB-PPI cable.

Table 11-22 Using Freeport mode

Network configuration		Description
Using Freeport over an RS232 connection		Example: Using an S7-200 SMART CPU with an electronic scale that has an RS232 port. • Connect the two devices using one of the following methods: – RS232/PPI Multi-Master cable connects the RS232 port on the scale to the RS485 port on the CPU. (Set the cable to PPI/Freeport mode, switch 5 = 0.) – Using the CM01 signal board (SB) which supports RS232 and RS485, you can connect the RS232 device directly to the CPU SB RS232 without using the RS232/PPI cable. • CPU uses Freeport to communicate with the scale. • Baud rate can be from 1.200 Kbps to 115.2 Kbps. • User program defines the protocol.
Using USS protocol		Example: Using an S7-200 SMART CPU with SIMODRIVE MicroMaster drives. • STEP 7-Micro/WIN SMART provides a USS library. • The CPU is a master, and the drives are slaves.
Creating a user program that emulates a slave device on another network		Example: Connecting S7-200 SMART CPUs to a Modbus network. • User program in the CPU emulates a Modbus slave. • STEP 7-Micro/WIN SMART provides a Modbus library.

11.6.6.2 Using the RS232/PPI Multi-Master cable and Freeport mode with RS232 devices

Purpose

You can use the RS232/PPI Multi-Master cable and the Freeport communication functions to connect the S7-200 SMART CPU to many devices that are compatible with the RS232 standard. The cable must be set to PPI/Freeport mode (switch 5 = 0) for Freeport operation. Switch 6 selects either Local mode (DCE) (switch 6 = 0), or Remote mode (DTE) (switch 6 = 1).

The RS232/PPI Multi-Master cable is in Transmit mode when data is transmitted from the RS232 port to the RS485 port. The cable is in Receive mode when it is idle or is transmitting data from the RS485 port to the RS232 port. The cable changes from Receive to Transmit mode immediately when it detects characters on the RS232 transmit line.

The CM01 signal board (SB) supports both RS232 half-duplex and RS485. With the CM01 signal board, you can connect an RS232 device directly to the CPU SB RS232 port without using a RS232/PPI cable.

Baud Rates and turnaround time

The RS232/PPI Multi-Master cable supports baud rates between 1.2 Kbps and 115.2 Kbps. Use the DIP switches on the housing of the RS232/PPI Multi-Master cable to configure the cable for the correct baud rate. The following table shows the baud rates (bits per second) and switch positions.

Table 11-23 Turnaround time and settings

Baud rate	Turnaround time	Settings (1 = Up)
115200	0.15 ms	110
57600	0.3 ms	111
38400	0.5 ms	000
19200	1.0 ms	001
9600	2.0 ms	010
4800	4.0 ms	011
2400	7.0 ms	100
1200	14.0 ms	101

The cable switches back to Receive mode when the RS232 transmit line is in the idle state for a period of time defined as the turnaround time of the cable. The baud rate selection of the cable determines the turnaround time, as shown in the table.

If you use the RS232/PPI Multi-Master cable in a system which uses Freeport communications, the program in the S7-200 SMART CPU must comprehend the turnaround time for the following situations:

- The CPU responds to messages transmitted by the RS232 device.
  After the CPU receives a request message from the RS232 device, the CPU must delay the transmission of a response message for a period of time greater than or equal to the turnaround time of the cable.
- The RS232 device responds to messages transmitted from the CPU.
  After the CPU receives a response message from the RS232 device, the CPU must delay the transmission of the next request message for a period of time greater than or equal to the turnaround time of the cable.

In both situations, the delay allows the RS232/PPI Multi-Master cable sufficient time to switch from Transmit mode to Receive mode so that data can be transmitted from the RS485 port to the RS232 port.

11.7 RS232

An RS232 network is a point-to-point connection between two devices. RS232 allows for data transfer at relatively slow speeds (up to 115.2 Kbps) and short distances (up to 50 feet).

Possible RS232 connections include the following:

- Freeport
- Modems
- RS232-compatible devices (for example, a bar code scanner)
- Devices that have RS232 interfaces (for example, a control system)
- RS232 displays

Libraries 12

12.1 Library types (Siemens and user-defined)

Library types

Siemens provides two types of libraries with the installation of STEP 7-Micro/WIN SMART:

- Siemens-supplied (Modbus RTU (Page 579), Modbus TCP (Page 599), Open User Communication (Page 616),PN Read Write Record library (Page 659), SINAMICS Library (Page 678) and USS protocol (Page 662))
- User-defined (Page 575) (libraries that you create from project POUs or obtain from other sources)

Note

You must start STEP 7-Micro/WIN SMART V3 with a "Run as administrator" command to create a user-defined library.

Note

You cannot give a user-defined library the same name as a Siemens-supplied library.

Note

Only call the library functions from either the main program or from interrupt routines, but not both.

Note

The program space varies according to the start address of V memory area. The program space of each library is calculated based on the start address VB0.

Modbus RTU

STEP 7-Micro/WIN SMART makes communicating to Modbus devices easier by including pre-configured subroutines and interrupt routines for Modbus communication through the serial ports of the CPUs. With the Modbus RTU instructions, you can configure the S7-200 SMART to act as a Modbus RTU master or slave device.

You can find these instructions in the Libraries folder of the Instructions folder in the project tree (Page 149). When you put a Modbus RTU library instruction in your program, STEP 7-Micro/WIN SMART automatically puts one or more associated subroutines and interrupt routines in your project.

Modbus TCP

STEP 7-Micro/WIN SMART makes communicating to Modbus devices easier by including preconfigured subroutines that are specifically designed for Modbus communication over Industrial Ethernet. With the Modbus TCP protocol instructions, you can configure the S7-200 SMART to act as a Modbus TCP client or server device.

You can find these instructions in the Libraries folder of the Instructions folder in the project tree (Page 149). When you put a Modbus TCP library instruction in your program, STEP 7-Micro/WIN SMART automatically puts one or more associated subroutines in your project.

Open user communication

Open User Communication (OUC) provides a mechanism for your program to transmit and receive messages over an Ethernet network. You can select the Ethernet protocol used as the transport mechanism: UDP, TCP, or ISO-on-TCP

You can find these instructions in the Libraries folder of the Instructions folder in the project tree (Page 149). When you put an OUC library instruction in your program, STEP 7-Micro/WIN SMART automatically puts one or more associated subroutines in your project.

PN Read Write Record library

STEP 7-Micro/WIN SMART makes reading a data record from any connected PROFINET device or writing a data record to any connected PROFINET device easier by including pre-configured subroutines that are specifically designed for the PN Read Write Record library.

SINAMICS library

STEP 7-Micro/WIN SMART makes controlling the drive easier by including pre-configured subroutines that are specifically designed for the SINAMICS library. With the SINAMICS instructions, you can control the position and speed of a physical drive, and read or modify the drive parameters.

You can find these instructions in the Libraries folder of the Instructions folder in the project tree (Page 149). When you put a SINAMICS library instruction in your program, STEP 7-Micro/WIN SMART automatically puts one or more associated subroutines in your project.

USS protocol

The USS protocol library supports Siemens drives. The STEP 7-Micro/WIN SMART USS instruction libraries make controlling drives easier by including preconfigured subroutines and interrupt routines that are specifically designed for using the USS protocol to communicate with the drive. With the USS instructions, you can control the physical drive and the read/write drive parameters.

You can find these instructions in the Libraries folder of the Instructions folder in the project tree (Page 149). When you put a USS library instruction in your program,

STEP 7-Micro/WIN SMART automatically puts one or more associated subroutines in your project

WARNING

Security: Protecting networks against physical access and possible reads and writes of PLC data

Communication through Modbus RTU, Open User Communication, and USS Protocol library instructions have no security features. If an attacker can physically access your networks through one of these forms of communication, the attacker can possibly read and write PLC data. Unauthorized access to PLC data can result in death or severe personal injury.

You must protect these forms of communication by limiting physical access. For security information and recommendations, refer to the following document: Operational Guidelines for Industrial Security (http://www.industry.siemens.com/topics/global/en/industrial-security/Documents/operational_guidelines_industrial_security_en.pdf)

12.2 Creating a user-defined library of instructions

STEP 7-Micro/WIN SMART V3 allows you either to create a custom library of instructions, or to use a library created by someone else.

Creating a library

To create a user-defined library of instructions, you create standard STEP 7-Micro/WIN SMART V3 subroutines and group them together. By grouping these subroutines into a library, you can hide the code. Putting code into a library helps prevent unwanted changes and provides a higher degree of protection for the technology (know-how) of the author.

To create a user-defined library, perform the following tasks:

1. Write the program as a standard project and put the function to be included in the library into subroutines.
2. Ensure that you have assigned a symbolic name to all V memory addresses in the subroutines or interrupt routines. To minimize the amount of V memory that the library requires, use sequential V memory addresses.
3. Rename the subroutines to the names that you want to appear in the instruction library.
4. Select the subroutines that you want to include in the library.
5. Click the Create button Create from the Libraries area of the File menu ribbon strip to compile and create the new library. If the subroutine references interrupts, STEP 7-Micro/WIN SMART V3 also includes the interrupt routines in the library.

The libraries that you create are available for new or existing projects. They are not available for the project from which you created them.

Further information

See the STEP 7-Micro/WIN SMART V3 online help library topics for library programming tips and a user-defined library example.

Note

Lib 3.0 does not support complex data and function blocks (FBs); the file extension for Lib has been changed to smartlibV3.

Note

Using custom libraries

You must be responsible for testing any libraries that you add to your project. Siemens bears no responsibility for custom libraries.

If the CPU in your project does not support the instructions in your library, you will get errors when you compile.

12.3 Overview of Modbus communication

STEP 7-Micro/WIN SMART V3 and the S7-200 SMART CPUs make communicating to Modbus devices easier by including pre-configured subroutines and interrupt routines for the following types of communication:

- Modbus RTU communication through the serial ports of the CPUs
- Modbus TCP communication over Industrial Ethernet

Some characteristics of Modbus communication are common to both Modbus RTU and Modbus TCP.

12.3.1 Modbus addressing

Modbus addresses are five-to-six digit numbers that indicate the data type as well as the address value.

Modbus RTU master/ Modbus TCP client addressing

Modbus RTU master and Modbus TCP client instructions map the address to the correct functions to send to the slave device or client device. The Modbus address definitions are as follows:

- 00001 to 09999 are discrete outputs (coils)
- 10001 to 19999 are discrete inputs (contacts)
- 30001 to 39999 are input registers (generally analog inputs)
- 40001 to 49999 and 400001 to 465535 are holding registers

All Modbus addresses are one-based, meaning that the first data value starts at address one. The actual range of valid addresses depends on the slave device. Different devices support different data types and address ranges.

Modbus RTU slave/ Modbus TCP server addressing

The Modbus RTU slave instructions and Modbus TCP server instructions support the following addresses:

- 00001 to 09216 are discrete outputs mapped to Q0.0 - Q1151.7.
- 10001 to 19216 are discrete inputs mapped to I0.0 - I1151.7 .
- 30001 to 30266 are analog input registers mapped to AIW0 - AIW530.
- 40001 to 49999 and 400001 to 465535 are holding registers mapped to V memory.

Mapping Modbus addresses to CPU addresses

All Modbus addresses are one-based.

Table 12-1 Mapping Modbus addresses to CPU addresses

Modbus address	CPU address
00001	Q0.0
00002	Q0.1
00003	Q0.2
...	...
01025	Q128.0 [1]
01026	Q128.1[1]
01027	Q128.2[1]
...	...
09215	Q1151.6[1]
09216	Q1151.7[1]
10001	I0.0
10002	I0.1
10003	I0.2
...	...
11025	I128.0[1]
11026	I128.1[1]
11027	I128.2[1]
...	...
19215	I1151.6[1]
19216	I1151.7[1]
30001	AIW0
30002	AIW2
30003	AIW4
...	...

Modbus address		CPU address
30266		AIW530
40001	400001	Vx (Holding reg. start)
40002	400002	Vx+2 =(Hold reg. start+2)
40003	400003	Vx+4 =(Hold reg. start+4)
...		...
4yyyy	4zzzzz	Vx+2(yyyy-1) or Vx+2(zzzzz-1)

Note

[1] CPU V2.4 and later versions support the updated memory address: Q128.0 - Q1151.7, and I128.0 - I1151.7. For the detailed information, refer to Memory ranges and features (Page 1117).

MBUS parameters that limit slave accessibility

The Modbus slave/protocol allows you to limit the number of inputs, outputs, analog inputs, and holding registers (V memory) that are accessible to a Modbus master.

- **MaxIQ** assigns the maximum number of discrete inputs or outputs (I or Q addresses) a Modbus master is allowed to access.
- **MaxAI** assigns the maximum number of input registers (AIW addresses) a Modbus master is allowed to access.
- **MaxHold** assigns the maximum number of holding registers (V memory words) a Modbus master is allowed to access.

See the description of the MBUS_INIT (Page 592) instruction for more information on setting up the memory restrictions for the Modbus RTU slave.

See the description of the MBUS SERVER (Page 605) instruction for more information on setting up the memory restrictions for the Modbus TCP server.

12.3.2 Modbus read and write functions

The Modbus RTU master instructions utilize the Modbus functions shown below to read or write a specific Modbus address. The Modbus RTU slave device must support the appropriate Modbus function to read or write a particular Modbus address.

Table 12-2 Required Modbus slave function support

Modbus address	Read or write	Modbus slave function required
00001 – 09999 discrete outputs	Read	Function 1
	Write	Function 5 for a single output point Function 15 for multiple output points
10001 – 19999 discrete inputs	Read	Function 2
	Write	not possible

Modbus address	Read or write	Modbus slave function required
30001 – 39999 input registers	Read	Function 4
	Write	not possible
40001 – 49999 holding registers 400001 - 465535	Read	Function 3
	Write	Function 6 for a single register Function 16 for multiple registers

Modbus message length

The S7-200 SMART CPU supports Modbus messages with up to 240 bytes (1920 bits or 120 registers) of data per message. Some slave devices might support fewer than 240 bytes of data.

12.4 Modbus RTU library instructions

12.4.1 Modbus RTU communication overview

12.4.1.1 Modbus RTU library features

STEP 7-Micro/WIN SMART V3 includes Siemens Modbus RTU libraries. The Modbus RTU libraries includes pre-configured subroutines and interrupt routines that make communicating to Modbus RTU master and slave devices easier.

STEP 7-Micro/WIN SMART V3 supports Modbus communication over RS-485 (integrated port 0 and optional signal board port 1 and port 2) and RS-232 (optional signal board port 1 and port 2) for both master and slave devices.

Modbus RTU master instructions can configure the S7-200 SMART to act as a Modbus RTU master device and communicate to one or more Modbus RTU slave devices. You can configure up to three Modbus RTU masters.

Modbus RTU slave instructions can configure the S7-200 SMART to act as a Modbus RTU slave device and communicate with Modbus RTU master devices.

Open the Libraries folder in the Instruction folder of the project tree for access to the Modbus instructions. When you place a Modbus instruction in your program, STEP 7-Micro/WIN SMART V3 places one or more associated POUs in your project.

Note

Only call the library functions from either the main program or from interrupt routines, but not both.

12.4.1.2 Requirements for using Modbus instructions

Modbus RTU master protocol

Modbus master instructions use the following resources from the CPU:

- MBUS_CTRL / MB_CTRL2 / MB_CTRL3 (Page 584) execution initializes the Modbus master protocol and dedicates the assigned CPU port (0, 1 or 2) for Modbus master communication. When you use a CPU port for Modbus communications, you cannot use it for any other purpose, including communication with an HMI.
- Modbus master instructions affect all of the SM locations associated with Freeport communications on the port assigned by the MBUS_CTRL / MB_CTRL2 / MB_CTRL3 instruction.
- Modbus master instructions use interrupts for some functions. These interrupts must not be disabled by the user program.
- Modbus master instructions program size
  - 3 subroutines and 1 interrupt routine
  - Variables for Modbus master instructions require a 286 byte block of V memory. You must assign the starting address for this block using the Library Memory command in STEP 7-Micro/WIN SMART V3. This command is available from the shortcut memory of the Library node under the Program Block node in the project tree, or from the Libraries section of the File menu ribbon strip.

Note

To change the CPU communication port from Modbus back to PPI so that you can communicate with an HMI device, set the mode parameter of the MBUS_CTRL/MB_CTRL2/MB_CTRL3 instruction to a zero (0).

Modbus RTU slave protocol

Modbus slave protocol instructions use the following resources from the CPU:

- The MBUS_INIT instruction (Page 592) initializes the Modbus slave protocol and dedicates the assigned CPU port (0, 1 or 2) for Modbus slave communication.
  When you use a CPU port for Modbus communication, you cannot use it for any other purpose, including communications with an HMI.
- Modbus slave instructions affect all of the SM locations associated with a Freeport communications on the port assigned by the MBUS_INIT instruction.
- Modbus slave instructions program size:
  - 3 subroutines and 2 interrupts.
  - 2113 bytes of program space for the slave instructions and support routines.
  - The variables for the Modbus slave instructions require a 781 byte block of V memory. You must assign the starting address for this block using the Library Memory command in STEP 7-Micro/WIN SMART V3. This command is available from the shortcut memory of the Library node under the Program Block node in the project tree, or from the Libraries section of the File menu ribbon strip.

Note

To change the CPU communication port from Modbus back to PPI so that you can communicate with an HMI device, set the mode parameter of the MBUS_INIT instruction to a zero (0).

12.4.1.3 Initialization and execution time for Modbus protocol

- **Modbus RTU master protocol:** The master protocol requires a small amount of time every scan to execute the MBUS_CTRL, MB_CTRL2 and MB_CTRL3 instruction, if present. The time is about 0.2 milliseconds when MBUS_CTRL / MB_CTRL2 / MB_CTRL3 is initializing the Modbus master (first scan), and about 0.1 milliseconds on subsequent scans.
  Execution of the MBUS_MSG / MB_MSG2 / MB_MSG3 instruction extends the scan time, especially in calculating the Modbus CRC for the request and response. The CRC (Cyclic Redundancy Check) ensures the integrity of the communications message. Each word in request and in the response extends the PLC scan time by about 86 microseconds. A maximum request/response (read or write of 120 words) extends the scan time to approximately 10.3 milliseconds. A read request extends the scan mainly when the program is receiving a response from a slave, and to a lesser extent when sending the request. A write request extends the scan mainly when sending data to a slave, and to a lesser extent when receiving a response.
- **Modbus RTU slave protocol:** Modbus communication uses a CRC (cyclic redundancy check) to ensure the integrity of the communications messages. The Modbus slave protocol uses a table of pre-calculated values to decrease the time required to process a message. The initialization of this CRC table requires about 11.3 milliseconds. The MBUS_INIT instruction performs this initialization, which normally happens during the first scan after entering RUN mode. You are responsible for resetting the watchdog timer if the time required by the MBUS_INIT instruction and any other user initialization exceeds the 500 millisecond scan watchdog time. Writing to the outputs of the module resets the output module watchdog timer.
  MBUS_SLAVE extends the scan time when it services a request. Calculating the Modbus CRC extends the scan time by about 40 microseconds for every byte in the request and in the response. A maximum request/response (read or write of 120 words) extends the scan time by approximately 4.8 milliseconds.

12.4.2 Modbus RTU master

12.4.2.1 Using the Modbus RTU master instructions

STEP 7-Micro/WIN SMART and the S7-200 SMART CPU support three Modbus RTU Masters. For a single Modbus RTU Master, use the instructions MBUS_CTRL (Page 584)and MBUS_MSG (Page 586). For a second Modbus RTU Master, use the instructions MB_CTRL2 (Page 584) and MB_MSG2 (Page 586). For a third Modbus RTU Master, use the instructions MB_CTRL3 and MB_MSG3.

If you use three Modbus masters in your project, make sure to use different port numbers for MBUS_CTRL, MB_CTRL2 and MB_CTRL3.

Procedure

To use the Modbus RTU master instructions in your S7-200 SMART program, follow these steps:

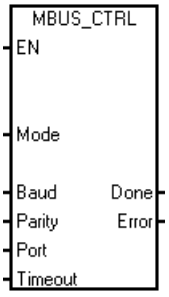
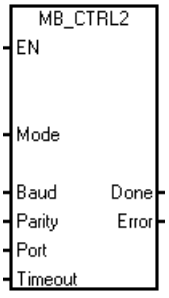
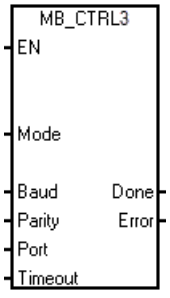
1. Insert the MBUS_CTRL / MB_CTRL2 / MB_CTRL3 instruction in your program and execute it on every scan. You can use the MBUS_CTRL / MB_CTRL2 / MB_CTRL3 instruction either to initiate or to change the Modbus communications parameters. When you insert the MBUS_CTRL / MB_CTRL2 / MB_CTRL3 instruction, STEP 7-Micro/WIN SMART adds several protected subroutines and interrupt routines to your program.
2. Click the Memory button Memory from the Libraries area of the File menu ribbon strip to assign a starting address for the V-Memory that the Modbus library requires. Alternatively, you can right-click the Program Block node in the project tree and select "Library Memory" from the context menu.
3. Place one or more MBUS_MSG / MB_MSG2 / MB_MSG3 instructions in your program. You can add as many MBUS_MSG / MB_MSG2 / MB_MSG3 instructions to your program as you require, but only one of these instructions can be active at a time.
4. Connect a communications cable between the S7-200 SMART CPU port you assigned with the MBUS_CTRL / MB_CTRL2 / MB_CTRL3 port parameter and the Modbus slave device.

NOTICE
Avoiding unwanted current flow Interconnecting equipment with different reference potentials can cause unwanted currents to flow through the interconnecting cable. These unwanted currents can cause communications errors or damage equipment. Ensure that you connect all equipment that with a communications cable that either shares a common circuit reference or is isolated to prevent unwanted current flows.

12.4.2.2 MBUS_CTRL / MB_CTRL2 / MB_CTRL3 instruction (initialize master)

MBUS_CTRL, MB_CTRL2 and MB_CTRL3 have identical behavior and parameters. MBUS_CTRL is for a single Modbus RTU master. MB_CTRL2 is for a second Modbus RTU master. MB_CTRL3 is for a third Modbus RTU master. Correspondingly, MBUS_MSG is for use with MBUS_CTRL and a single Modbus RTU master. MB_MSG2 is for use with MB_CTRL2 and a second Modbus RTU master. MB_MSG3 is for use with MB_CTRL3 and a third Modbus RTU master.

Table 12-3 MBUS_CTRL, MB_CTRL2 and MB_CTRL3 instruction

LAD / FBD	STL	Description
  	**CALL MBUS_CTRL, Mode, Baud, Parity, Port, Timeout, Done, Error** **CALL MB_CTRL2, Mode, Baud, Parity, Port, Timeout, Done, Error** **CALL MB_CTRL3, Mode, Baud, Parity, Port, Timeout, Done, Error**	The program calls the MBUS_CTRL / MB_CTRL2 / MB_CTRL3 instruction to initialize, monitor, or to disable Modbus communications. Before executing the MBUS_MSG / MB_MSG2 / MB_MSG3 instruction, the program must execute the MBUS_CTRL / MB_CTRL2 / MB_CTRL3 without errors. The instruction completes and sets the Done bit ON before continuing to the next instruction. This instruction executes on each scan when the EN input is on.

The program must call the MBUS_CTRL / MB_CTRL2 / MB_CTRL3 instruction every scan (including the first scan) to allow it to monitor the progress of any outstanding messages initiated with the MBUS_MSG / MB_MSG2 / MB_MSG3 instruction. The Modbus master

protocol does not operate correctly unless the program executes MBUS_CTRL / MB_CTRL2 / MB_CTRL3 every scan.

Table 12-4 Parameters for the MBUS_CTRL / MB_CTRL2 instruction

Parameter	Data type	Operands
Mode	BOOL	I, Q, M, S, SM, T, C, V, L
Baud	DWORD	VD, ID, QD, MD, SD, SMD, LD, AC, Constant, *VD, *AC, *LD
Parity, Port	BYTE	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *AC, *LD
Timeout	WORD	VW, IW, QW, MW, SW, SMW, LW, AC, Constant, *VD, *AC, *LD
Done	BOOL	I, Q, M, S, SM, T, C, V, L
Error	BYTE	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD

The value for the **Mode** input selects the communications protocol. An input value of 1 assigns the CPU port to Modbus protocol and enables the protocol. An input value of 0 assigns the CPU port to PPI system protocol and disables Modbus protocol.

Parameter **Baud** sets the baud rate at 1200, 2400, 4800, 9600, 19200, 38400, 57600, or 115200.

Parameter **Parity** is set to match the parity of the Modbus slave device. All settings use one start bit and one stop bit. The allowed values are: 0 (no parity). 1 (odd parity), and 2 (even parity).

Parameter **Port** sets the physical communication port (0 = RS-485 integrated in CPU, 1 = RS-485 or RS-232 located on the first optional CM01 signal board, 2 = RS-485 or RS-232 located on the second optional CM01 signal board).

Parameter **Timeout** is set to the number of milliseconds to wait for the response from the slave. The Timeout value can be set anywhere in the range of 1 millisecond to 32767 milliseconds. A typical value would be 1000 milliseconds (1 second). The Timeout parameter should be set to a value large enough so that the slave device has time to respond at the selected baud rate.

The Timeout parameter is used to determine if the Modbus slave device is responding to a request. The Timeout value determines how long the Modbus Master will wait for the first character of the response after the last character of the request has been sent. The Modbus master will receive the entire response from the Modbus slave device if at least one character of the response is received within the Timeout time.

When the MBUS_CTRL / MB_CTRL2 / MB_CTRL3 instruction completes, the instruction returns TRUE for the Done output.

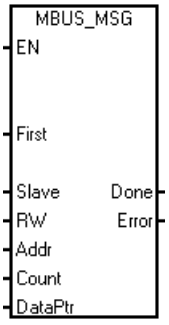
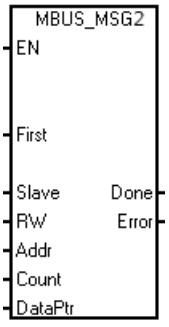
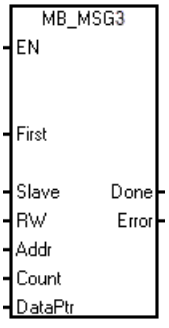
The Error output contains the result of executing the instruction.

See also Modbus RTU master execution error codes (Page 589)

12.4.2.3 MBUS_MSG / MB_MSG2 / MB_MSG3 instruction

MBUS_MSG, MB_MSG2 and MB_MSG3 have identical behavior and parameters. MBUS_MSG is for a single Modbus RTU master. MB_MSG2 is for a second Modbus RTU master. MB_MSG3 is for a three Modbus RTU master.

Table 12-5 MBUS_MSG / MB_MSG2 / MB_MSG3 instruction

LAD / FBD	STL	Description
  	**CALL MBUS_MSG, First, Slave, RW, Addr, Count, DataPtr, Done, Error** **CALL MB_MSG2, First, Slave, RW, Addr, Count, DataPtr, Done, Error** **CALL MB_MSG3, First, Slave, RW, Addr, Count, DataPtr, Done, Error**	The program calls the MBUS_MSG / MB_MSG2 / MB_MSG3 instruction to initiate a request to a Modbus slave and process the response.

The MBUS_MSG / MB_MSG2 / MB_MSG3 instruction initiates a master request to a Modbus slave when both the EN input and the First inputs are ON. Sending the request, waiting for the response, and processing the response usually requires several PLC scan times. The EN input must be ON to enable the send request, and must remain ON until the instruction returns ON (True) for the Done bit.

Only one each of MBUS_MSG, MB_MSG2 or MB_MSG3 instruction can be active at a time. If the program enables more than one MBUS_MSG instruction or more than one MB_MSG2 or MB_MSG3 instruction, then the CPU processes the first MBUS_MSG, MB_MSG2 or MB_MSG3

instruction and all subsequent MBUS_MSG, MB_MSG2 or MB_MSG3 instructions will abort with an error code 6.

Table 12-6 Parameters for the MBUS_MSG / MB_MSG2 / MB_MSG3 instruction

Parameter	Data type	Operands
First	BOOL	I, Q, M, S, SM, T, C, V, L (Power flow conditioned by a positive edge detection element)
Slave	BYTE	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *AC, *LD
RW	BYTE	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *AC, *LD
Addr	DWORD	VD, ID, QD, MD, SD, SMD, LD, AC, Constant, *VD, *AC, *LD
Count	INT	VW, IW, QW, MW, SW, SMW, LW, AC, Constant, *VD, *AC, *LD
DataPtr	DWORD	&VB
Done	BOOL	I, Q, M, S, SM, T, C, V, L
Error	BYTE	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD

Set parameter **First** ON for only one scan when there is a new request to send. Pulse the First input through an edge detection element (for example, Positive Edge), which causes the program to transmit the request one time. See the example program (Page 595) for details.

Parameter **Slave** is the address of the Modbus slave device. The allowed range is 0 through 247. Address 0 is the broadcast address. Use address 0 only for write requests. There is no response to a broadcast request to address 0. Not all slave devices support the broadcast address. The S7-200 SMART Modbus slave library does not support the broadcast address.

Use parameter **RW** to indicate whether this message is to be a read or a write: 0 (Read) and 1 (Write).

Discrete outputs (coils) and holding registers support both read and write requests. Discrete inputs (contacts) and input registers only support read requests.

Parameter **Addr** is the starting Modbus address. S7-200 SMART supports the following ranges of addresses:

- 00001 to 09999 for discrete outputs (coils)
- 10001 to 19999 for discrete inputs (contacts)
- 30001 to 39999 for input registers
- 40001 to 49999 and 400001 to 465535 for holding registers

The addresses that the Modbus slave device supports determine the actual range of values for Addr.

Parameter **Count** assigns the number of data elements to read or write in this request. The Count is the number of bits for the bit data types, and the number of words for the word data types.

- Address 0xxxx Count is the number of bits to read or write
- Address 1xxxx Count is the number of bits to read
- Address 3xxxx Count is the number of input register words to read
- Address 4xxxx or 4yyyyy Count is the number of holding register words to read or write

The MBUS_MSG / MB_MSG2 / MB_MSG3 instruction reads or writes a maximum of 120 words or 1920 bits (240 bytes of data). The actual limit on the value of Count depends on the limits in the Modbus slave device.

The parameter **DataPtr** is an indirect address pointer that points to the V memory in the CPU for the data associated with the read or write request. For a read request, set the DataPtr to the first CPU memory location used to store the data read from the Modbus slave. For a write request, set DataPtr to point to the first CPU memory location of the data to be sent to the Modbus slave.

The program passes the DataPtr value to MBUS_MSG / MB_MSG2 / MB_MSG3 as an indirect address pointer. For example, if the data to be written to a Modbus slave device starts at address VW200 in the CPU, the value for the DataPtr would be &VB200 (address of VB200). Pointers must always be a type VB even if they point to word data.

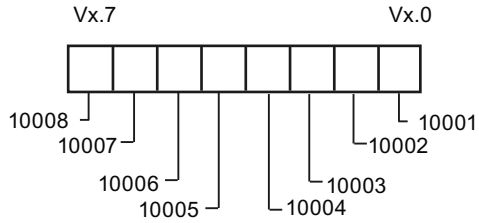
Memory layout

Holding registers (address 4xxxx or 4yyyyy) and input registers (address 3xxxx) are word values (2 bytes or 16 bits). CPU words are formatted the same as Modbus registers. The lower numbered V memory address is the most significant byte of the register. The higher numbered V memory address is the least significant byte of the register. The table below shows how the CPU byte and word addressing corresponds to the Modbus register format.

Table 12-7 Modbus Holding Register

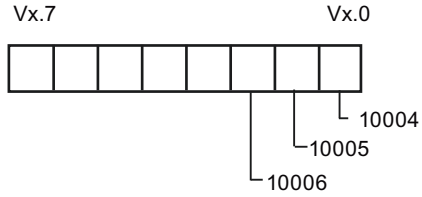
CPU memory byte address		**CPU memory word address**		**Modbus holding register address**	
Address	**Hex data**	**Address**	**Hex data**	**Address**	**Hex data**
VB200	12	VW200	12 34	40001	12 34
VB201	34				
VB202	56	VW202	56 78	40002	56 78
VB203	78				
VB204	9A	VW204	9A BC	40003	9A BC
VB205	BC				

The CPU reads and writes the bit data (addresses 0xxxx and 1xxxx) areas as packed bytes; that is, each byte consists of 8 bits of data. The least significant bit of the first data byte is the addressed bit number (the parameter Addr). If you intend to write only a single bit then you must set the bit in the least significant bit (Vx.0) of the byte pointed to by DataPtr.



Format for Packed Bytes (Discrete Input Addresses)

For bit data addresses that do not start on a byte boundary, you must set the bit corresponding to the starting address in the least significant bit of the byte. See the example of the packed byte format for 3 bits starting at Modbus address 10004.



Format for Packed Bytes (Discrete input starting at address 10004)

When writing to the discrete output data type (coils), you must place the bits in the correct bit positions within the packed byte before passing the data to the MBUS_MSG / MB_MSG2 / MB_MSG3 instruction by means of the DataPtr.

Outputs

The Done output is FALSE after the program has sent a request and before the program has received a response. The Done output is TRUE when the response is complete or when the MBUS_MSG / MB_MSG2 / MB_MSG instruction aborts because of an error.

The Error output (Page 589) is valid only when the Done output is TRUE.

12.4.2.4 Modbus RTU master execution error codes

The high numbered error codes (starting with 101) are errors that are returned by the Modbus slave device. These errors indicate that the slave does not support the requested function or that the requested address (either data type or range of addresses) is not supported by the Modbus slave device.

The low numbered error codes (1 through 12) are errors that are detected by the MBUS_MSG instruction. These error codes generally indicate a problem with the input parameters of the MBUS_MSG instruction, or a problem receiving the response from the slave. Parity and CRC errors indicate that there was a response but that the data was not received correctly. This is usually caused by an electrical problem such as a bad connection or electrical noise.

MBUS_CTRL error code	Description
0	No error
1	Invalid parity type
2	Invalid baud rate
3	Invalid timeout

MBUS_CTRL error code	Description
4	Invalid mode
9	Invalid port number
10	Signal board port 1 missing or not configured
11	Signal board port 2 missing or not configured

MBUS_MSG error code	Description
0	No error
1	Parity error in response: This is only possible if even or odd parity is used. The transmission was disturbed and possibly incorrect data was received. This error is usually caused by an electrical problem such as incorrect wiring or electrical noise affecting the communication.
2	Not used
3	Receive timeout: There was no response from the slave within the Timeout time. Some possible causes are bad electrical connections to the slave device, master and slave are set to a different baud rate / parity setting, and incorrect slave address.
4	Error in request parameter: One or more of the input parameters (Slave, RW, Addr, or Count) is set to an illegal value. Check the documentation for allowed values for the input parameters.
5	Modbus master not enabled: Call MBUS_CTRL on every scan prior to calling MBUS_MSG.
6	Modbus is busy with another request: Only one MBUS_MSG instruction can be active at a time.
7	Error in response: The response received does not correspond to the request. This indicates some problem in the slave device or that the wrong slave device answered the request.
8	CRC error in response: The transmission was disturbed and possibly incorrect data was received. This error is usually caused by an electrical problem such as incorrect wiring or electrical noise affecting the communication.
11	Invalid port number
12	Signal board port 1 missing or not configured
13	Signal board port 2 missing or not configured
101	Slave does not support the requested function at this address: See the required Modbus slave function support table in the "Using the Modbus master Instructions" help topic.
102	Slave does not support the data address: The requested address range of Addr plus Count is outside the allowed address range of the slave.
103	Slave does not support the data type: The Addr type is not supported by the slave device.
104	Slave device failure
105	Slave accepted the message but the response is delayed: This is an error for MBUS_MSG and the user program should resend the request at a later time.
106	Slave is busy and rejected the message: You can try the same request again to get a response.
107	Slave rejected the message for an unknown reason.
108	Slave memory parity error: There is an error in the slave device.

12.4.3 Modbus RTU slave

12.4.3.1 Using the Modbus RTU slave instructions

Procedure

To use the Modbus slave instructions in your S7-200 SMART program, follow these steps:

1. Insert the MBUS_INIT instruction in your program and execute the MBUS_INIT instruction for one scan only. You can use the MBUS_INIT instruction either to initiate or to change the communications parameters. When you insert the MBUS_INIT instruction, several hidden subroutines and interrupt routines are automatically added to your program.
2. Click the Memory button Memory from the Libraries area of the File menu ribbon strip to assign a starting address for the V memory that the Modbus library requires. Alternatively, you can right-click the Program Block node in the project tree and select "Library Memory" from the context menu. In addition to this V memory block, you define another V memory block with the HoldStart and MaxHold parameters of MBUS_INIT. Be careful that your program assignments in V memory do not overlap. If there is any overlap of the memory areas, the MBUS_INIT instruction returns an error.
3. Place only one MBUS_SLAVE instruction in your program. This instruction should be called every scan to service any requests that have been received.
4. Connect a communications cable between the S7-200 SMART CPU port you assigned with the MBUS_INIT port parameter and the Modbus master device.

NOTICE
Avoiding unwanted current flow Interconnecting equipment with different reference potentials can cause unwanted currents to flow through the interconnecting cable. These unwanted currents can cause communications errors or damage equipment. Ensure that all equipment that is connected with a communications cable either shares a common circuit reference or is isolated to prevent unwanted current flows.

The accumulators (AC0, AC1, AC2, AC3) are used by the Modbus slave instructions and appear in the Cross Reference listing. Prior to execution, the values in the accumulators of a Modbus slave instruction are saved and restored to the accumulators before the Modbus slave instruction is complete, ensuring that all user data in the accumulators is preserved while executing a Modbus slave instruction.

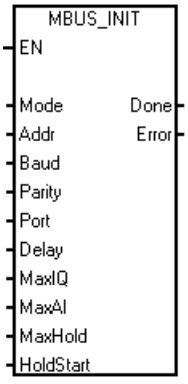
The Modbus slave instructions support the Modbus RTU protocol. These instructions use the Freeport feature of the S7-200 SMART CPU to support the most common Modbus functions. The following Modbus functions are supported:

Function	Description
1	Read single/multiple coil (discrete output) status. Function 1 returns the on/off status of any number of output points (Qs).
2	Read single/multiple contact (discrete input) status. Function 2 returns the on/off status of any number of input points (Is).

Function	Description
3	Read single/multiple holding registers. Function 3 returns the contents of V memory. Holding registers are word values under Modbus and allow you to read up to 120 words in one request.
4	Read single/multiple input registers. Function 4 returns analog Input values.
5	Write single coil (discrete output). Function 5 sets a discrete output point to the specified value. The point is not forced and the program can overwrite the value written by the Modbus request.
6	Write single holding register. Function 6 writes a single holding register value to the V memory of the S7-200 SMART.
15	Write multiple coils (discrete outputs). Function 15 writes the discrete output values to the Q image register of the S7-200 SMART. The starting output point must begin on a byte boundary (for example, Q0.0 or Q2.0) and the number of outputs written must be a multiple of eight. This is a restriction for the Modbus slave protocol instructions. The points are not forced and the program can overwrite the values written by the Modbus request.
16	Write multiple holding registers. Function 16 writes multiple holding registers to the V memory of the S7-200 SMART. There can be up to 120 words written in one request.

12.4.3.2 MBUS_INIT instruction (initialize slave)

Table 12-8 MBUS_INIT instruction

LAD/ FBD	STL	Description
	**CALL MBUS_INIT, Mode, Addr, Baud, Parity, Port, Delay, MaxIQ, MaxAI, MaxHold, HoldStart, Done, Error**	The MBUS_INIT instruction enables, initializes, or disables Modbus communications. Before an MBUS_SLAVE instruction can be used, MBUS_INIT must be executed without errors. The instruction completes and the Done bit is set immediately, before continuing to the next instruction. The instruction is executed on each scan when the EN input is ON.

The program must execute MBUS_INIT instruction exactly once for each change in communications state. Therefore, pulse the EN input through an edge detection element, or execute MBUS_INIT only on the first scan.

Table 12-9 MBUS_INIT parameters

Inputs/outputs	Data type	Operands
Mode, Addr, Parity, Port	BYTE	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *AC, *LD
Baud, HoldStart	DWORD	VD, ID, QD, MD, SD, SMD, LD, AC, Constant, *VD, *AC, *LD
Delay, MaxIQ, MaxAI, MaxHold	WORD	VW, IW, QW, MW, SW, SMW, LW, AC, Constant, *VD, *AC, *LD
Done	BOOL	I, Q, M, S, SM, T, C, V, L
Error	BYTE	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD

The value for the **Mode** input selects the communications protocol: an input value of 1 assigns Modbus protocol and enables the protocol, and an input value of 0 PPI protocol and disables Modbus protocol.

Parameter **Addr** sets the address at inclusive values between 1 and 247.

Parameter **Baud** sets the baud rate at 1200, 2400, 4800, 9600, 19200, 38400, 57600, or 115200.

Parameter **Parity** is set to match the parity of the Modbus master. All settings use one stop bit. The accepted values are: 0 (no parity), 1 (odd parity), and 2 (even parity).

Parameter **Port** sets the physical communication port (0 = RS-485 integrated in CPU, 1 = RS-485 or RS-232 located on the first optional signal board, 2 = RS-485 or RS-232 located on the second optional signal board).

Parameter **Delay** extends the standard Modbus end-of-message timeout condition by adding the assigned number of milliseconds to the standard Modbus message timeout. The typical value for this parameter should be 0 when operating on a wired network. If you are using modems with error correction, set the delay to a value of 50 to 100 milliseconds. If you are using spread spectrum radios, set the delay to a value of 10 to 100 milliseconds. The Delay value can be 0 to 32767 milliseconds.

Parameter **MaxIQ** sets the number of I and Q points available to Modbus addresses 0xxxx and 1xxxx at values of 0 to 9216. A value of 0 disables all reads and writes to the inputs and outputs. The suggested value for MaxIQ is 9216.

Parameter **MaxAI** sets the number of word input (AI) registers available to Modbus address 3xxxx at values of 0 to 266. A value of 0 disables reads of the analog inputs. The suggested value for MaxAI is 266.

Parameter **MaxHold** sets the number of word holding registers in V memory available to Modbus address 4xxxx or 4yyyyy. The suggested value for MaxHold is 32767. For example, if you want to allow Modbus master access for 2000 bytes of V memory, set MaxHold to a value of 1000 words (holding registers).

Parameter **HoldStart** is the address of the start of the holding registers in V memory. This value is generally set to VB0, so the parameter HoldStart is set to &VB0 (address of VB0). Other V memory addresses can be specified as the starting address for the holding registers to allow VB0 to be used elsewhere in the project. The Modbus master has access to MaxHold number of words of V memory starting at HoldStart.

When the MBUS_INIT instruction completes, the Done output is turned ON.

The Error output (Page 595) byte contains the result of executing the instruction. This output is only valid if Done is ON. If Done is OFF, the error parameter is not changed.

12.4.3.3 MBUS_SLAVE instruction

Table 12-10 MBUS_SLAVE instruction

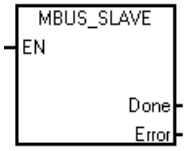
LAD / FBD	STL	Description
	`CALL MBUS_SLAVE, Done, Error`	The MBUS_SLAVE instruction is used to service a request from the Modbus master and must be executed every scan to allow it to check for and respond to Modbus requests. The instruction is executed on each scan when the EN input is ON. The MBUS_SLAVE instruction has no input parameters.

Table 12-11 Parameters for the MBUS_SLAVE Instruction

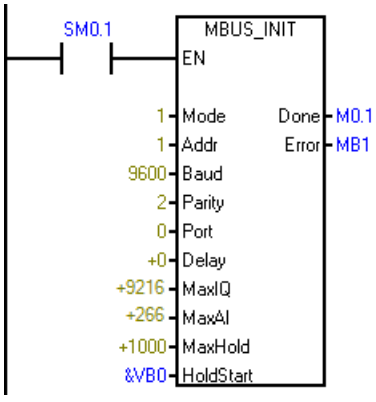
Parameter	Data Type	Operands
Done	BOOL	I, Q, M, S, SM, T, C, V, L
Error	BYTE	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD

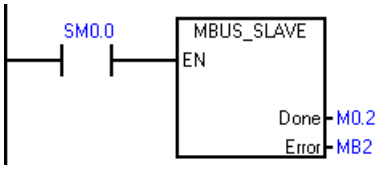
The Done output is ON when the MBUS_SLAVE instruction responds to a Modbus request. The Done output is OFF, if there was no request serviced.

The Error output contains the result of executing the instruction. This output is only valid if Done is ON. If Done is OFF, the error parameter is not changed.

Table 12-12 Example program of S7-200 SMART CPU operating as a Modbus slave

LAD		STL

	Initialize the Modbus Slave protocol on the first scan. Set the slave address to 1, set port 0 to 9600 baud with even parity, all access to all I, Q and AI values, allow access to 1000 holding registers (2000 bytes) starting at VB0.	`Network 1`
		`LD SM0.1`
		`CALL MBUS_INIT, 1, 1, 9600, 2, 0, 128, 32, 1000, &VB0, M0.1, MB1`

	Execute the Modbus Slave protocol on every scan.	`Network 2`
		`LD SM0.0`
		`CALL MBUS_SLAVE, M0.2, MB2`

See also

Modbus RTU slave execution error codes (Page 595)

12.4.3.4 Modbus RTU slave execution error codes

Error code	Description
0	No error
1	Memory range error
2	Illegal baud rate or parity
3	Illegal slave address
4	Illegal value for Modbus parameter
5	Holding registers overlap Modbus Slave symbols
6	Receive parity error
7	Receive CRC error
8	Illegal function request/function not supported
9	Illegal memory address in request
10	Slave function not enabled
11	Invalid port number
12	Signal board port 1 missing or not configured
13	Signal board port 2 missing or not configured

12.4.4 Modbus RTU master example program

This example program shows how to use the Modbus Master instructions to write and read four holding registers to and from a Modbus slave each time input I0.0 turns on.

The CPU writes four words starting at VW100 to the holding registers of the Modbus slave starting at address 40001.

The CPU then reads four holding registers from 40010 to 40013 from the Modbus slave and places the data into the V memory of the CPU starting at VW200.

This example uses a single master and the MBUS_CTRL and MBUS_MSG instructions. The same concepts apply to examples with a second master and the MB_CTRL2 and MB_MSG2 instructions or a third master and the MB_CTRL3 and MB_MSG3 instructions.

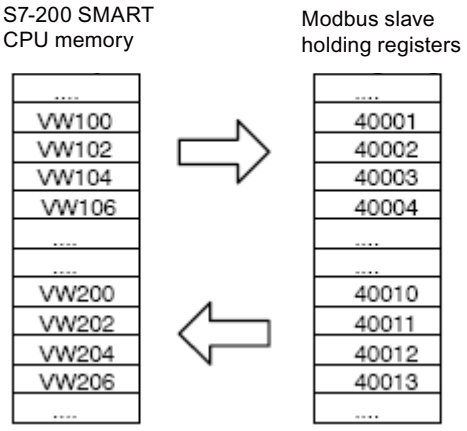
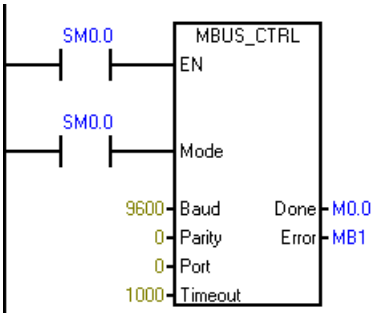
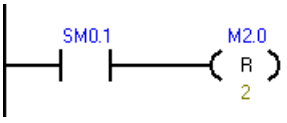
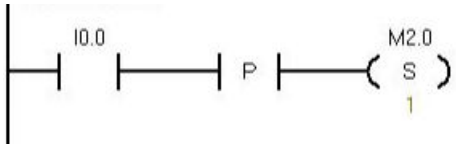
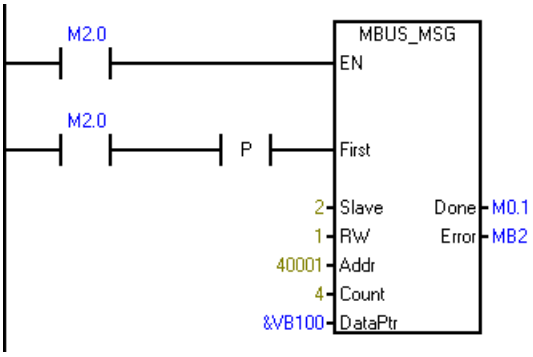
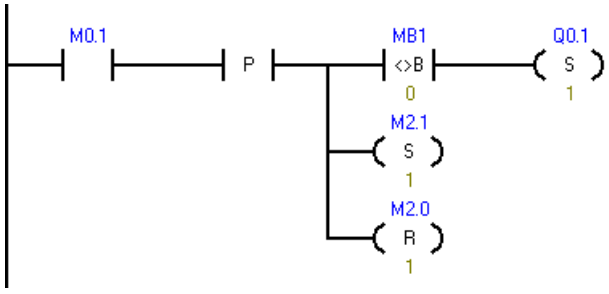
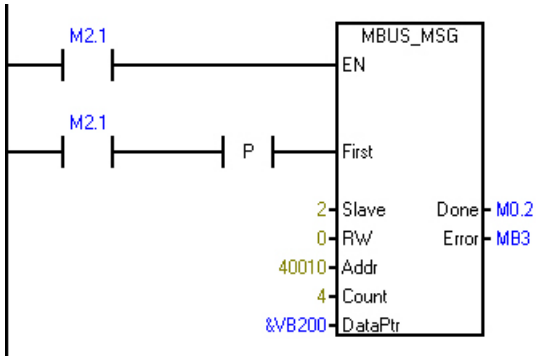
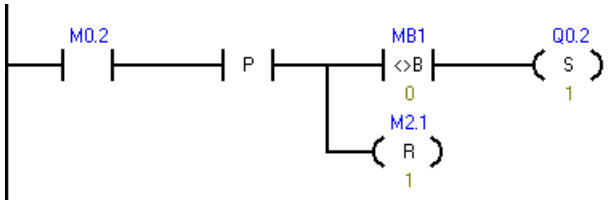


Figure 12-1 Example Program Data Transfers

The following program turns on outputs Q0.1 and Q0.2 if the MBUS_MSG instruction returns an error.

Table 12-13 Example Modbus master program

LAD	Description
	Network 1 Initialize and monitor the Modbus master by calling MBUS_CTRL on every scan. The Modbus master is set for 9.6 Kbps and no parity. The slave device is allowed 1000 milliseconds (1 second) to respond.
	Network 2 On the first scan, reset the enable flags (M2.0 and M2.1) used for the two MBUS_MSG instructions.
	Network 3 When I0.0 changes from OFF to ON, set the enable flag for the first MBUS_MSG instruction (M2.0).
	Network 4 Call the MBUS_MSG instruction when the first enable flag (M2.0) is ON. The First parameter must be set for only the first scan that the instruction is enabled. This instruction writes (RW = 1) 4 holding registers to slave 2. The write data is taken from VB100-VB107 (4 words) in the CPU and written to address 40001 - 40004 in the Modbus slave.

LAD	Description
	Network 5 When the first MBUS_MSG instruction is complete (Done goes from 0 to 1), clear the enable for the first MBUS_MSG and set the enable for the second MBUS_MSG instruction. If Error (MB1) is not zero, then set Q0.1 to show the error.
	Network 6 Call the second MBUS_MSG instruction when the second enable flag (M2.1) is ON. The First parameter must be set for only the first scan that the instruction is enabled. This instruction reads (RW = 0) 4 holding registers from slave 2. The data is read from address 40010 - 40013 in the Modbus slave and copied to VB200 - VB207 (4 words) in the CPU.
	Network 7 When the second MBUS_MSG instruction is complete (Done goes from 0 to 1), clear the enable for the second MBUS_MSG instruction. If Error (MB1) is not zero, then set Q0.2 to show the error.

12.4.5 Modbus RTU advanced user information

Overview

This topic contains information for advanced users of the Modbus RTU master library. Most users do not need this information and will not need to modify the default operation of the Modbus RTU master library.

Retries

The Modbus master instructions automatically resend the request to the slave device if one of the following errors is detected:

- There is no response within the response timeout time (parameter Timeout on the MBUS_CTRL / MB_CTRL2 / MB_CTRL3) instruction (Error code 3).
- The time between characters of the response exceeded the allowed value (Error code 3).
- There is a parity error in the response from the slave (Error code 1).

- There is a CRC error in the response from the slave (Error code 8).
- The returned function did not match the request (Error code 7).

The Modbus Master resends the request two additional times before setting the Done and Error output parameters.

You can change the number of retries by finding the variable mModbusRetries in the Modbus master variable table and changing this value after the program executes MBUS_CTRL / MB_CTRL2 / MB_CTRL3. The mModbusRetries value is a BYTE with a range of 0 to 255 retries.

Inter-character timeout

Modbus master execution aborts a response from a slave device if the time between characters in the response exceeds an assigned time limit. The default time is set to 100 milliseconds which should allow the Modbus master instructions to work with most slave devices over wire or telephone modems. If the CPU detects this error, the MBUS CTRL / MB_CTRL2 / MB_CTRL3 instruction returns error code 3 in the Error parameter.

Communication might possibly require a longer time between characters, either because of the transmission medium (for example, telephone modem) or because the slave device itself requires more time. You can lengthen this timeout by finding the variable *mModbusCharTimeout* in the Modbus master variable table and changing this value after MBUS_CTRL / MB_CTRL2 / MB_CTRL3 has been executed. The mModbusCharTimeout value is an INT with a range of 1 to 30000 milliseconds.

Single vs. multiple bit / word write functions

Some Modbus slave devices do not support the Modbus functions to write a single discrete output bit (Modbus function 5) or to write a single holding register (Modbus function 6). These devices only support the multiple bit write (Modbus function 15) or multiple register write (Modbus function 16) instead. The MBUS_MSG / MB_MSG2 / MB_MSG3 instruction returns an error code 101 if the slave device does not support the single bit/word Modbus functions.

The Modbus master protocol allows you to force the MBUS_MSG / MB_MSG2 / MB_MSG3 instruction to use the multiple bit/word Modbus functions instead of the single bit/word Modbus functions. You can force the multiple bit/word instructions by finding the variable *mModbusForceMulti* in the Modbus master variable table and changing this value after the program executes MBUS_CTRL / MB_CTRL2 / MB_CTRL3. Set mModbusForceMulti to TRUE to force the use of the multiple bit/word functions when writing a single bit or register.

Accumulator usage

The Modbus master instructions use accumulators (AC0, AC1, AC2, AC3), which appear in the Cross Reference listing. The Modbus master instructions save and restore the values in the accumulators. All CPU preserves all user data in the accumulators while executing the instructions.

Holding register addresses greater than 49999

Modbus holding addresses are within the range of 40001 to 49999. This range is adequate for most applications but there are some Modbus slave devices with data mapped into holding registers at a higher address range.

The MBUS_MSG / MB_MSG2 / MB_MSG3 instruction allows an additional range for the parameter **Addr** to support an extended range of holding register addresses at addresses 400001 to 465536.

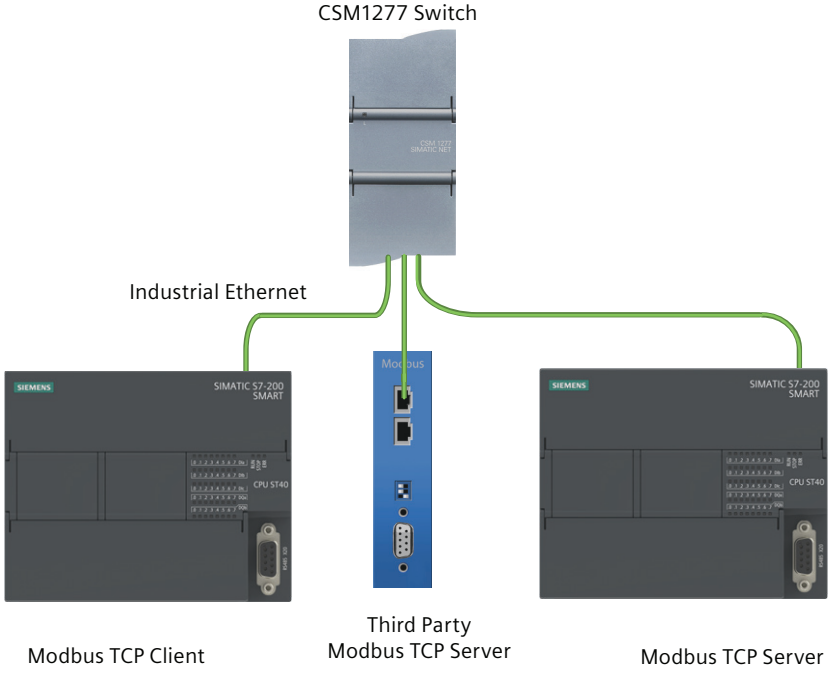
For example: to access holding register 16768, the **Addr** parameter of MBUS_MSG/MB_MSG2/MB_MSG3 should be set to 416768.

The extended addressing allows access to the full range of 65536 possible addresses supported by the Modbus protocol. This extended addressing is only applicable for holding registers.

12.5 Modbus TCP library

12.5.1 Modbus TCP library features

Modbus TCP is Modbus communication transmitted over an Industrial Ethernet TCP/IP network. The S7-200 SMART uses a client-server approach, in which a Modbus client device initiates a TCP/IP connection with a Modbus server device. With a connection established, a client makes a request to a server, which responds to the client's requests. The client can request to read a section of memory from the server device or to write a quantity of data to the memory of the server device. The server replies to the request with a response if the request is valid or an error message if the request is invalid.



STEP 7-Micro/WIN SMART provides two Modbus TCP library instructions, which are available from the Libraries folder of the Instructions folder in the STEP 7-Micro/WIN SMART V3 project tree:

- MBUS_CLIENT (Page 601)
- MBUS_SERVER (Page 605)

Modbus TCP client protocol

The Modbus client instruction (MBUS_CLIENT) uses the following resources from the CPU:

- One active connection resource for every connection to a Modbus server. MBUS_CLIENT automatically generates the connection ID.
- The Modbus client uses the following program entities:
  - 1 subroutine
  - 2849 bytes of program space
  - A 662-byte block of V memory for the instruction variables
    You must assign the starting address for this block from the Library Memory command in STEP 7-Micro/WIN SMART V3. The Library Memory command is accessible from the Program Block or Program Block > Library folder in the project tree after you have placed an MBUS_CLIENT instruction in the program.

Modbus TCP server protocol

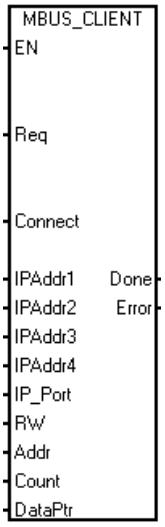
The Modbus server instruction (MBUS_SERVER) uses the following resources from the CPU:

- One passive connection resource for every connection to a Modbus server. MBUS_SERVER automatically generates the connection ID.
- The Modbus server uses the following program entities:
  - 1 subroutine
  - 2969 bytes of program space
  - A 445-byte block of V memory for the instruction variables
    You must assign the starting address for this block from the Library Memory command in STEP 7-Micro/WIN SMART V3. The Library Memory command is accessible from the Program Block or Program Block > Library folder in the project tree after you have placed an MBUS_SERVER instruction in the program.

12.5.2 Modbus TCP client

12.5.2.1 MBUS_CLIENT instruction

Table 12-14 MBUS_CLIENT instruction

LAD / FBD	STL	Description
	**Call MBUS_CLIENT Req, Connect, IPAddr1, IPAddr2, IPAddr3, IPAddr4, IP_Port, RW, Addr, Count, DataPtr, Done, Error**	MBUS_CLIENT communicates as a Modbus TCP client through the Ethernet port on the S7-200 SMART CPU. MBUS_CLIENT can make a client-server connection, send a Modbus function request, receive a client response, and connect to and disconnect from a Modbus TCP server.

The program execution cycle must call MBUS_CLIENT every scan until the Done output is TRUE. In each cycle, MBUS_CLIENT exits so that the program can continue to run. MBUS_CLIENT sets Done to TRUE when the client completes the request.

Table 12-15 Data types for the parameters

Parameter and type		Data type	Description
Req	IN	BOOL	The Req parameter allows your program to send a Modbus request to the server. FALSE: No Modbus communication request TRUE: Request to communicate with a Modbus TCP server
Connect	IN	BOOL	The Connect parameter allows your program to connect to and disconnect from a Modbus server device. If Connect = TRUE and a connection does not exist, then MBUS_CLIENT attempts to make a connection to the assigned IP address and port number. If Connect = FALSE and a connection exists, then MBUS_CLIENT attempts a disconnect operation. When Connect = FALSE, the CPU ignores any further requests. This means if the program calls MBUS_CLIENT with Req = TRUE but Connect = FALSE, the CPU ignores the request.
IPAddr1	IN	BYTE	First octet of the IP address of the server to which the client attempts to connect and subsequently communication using the Modbus application protocol.
IPAddr2	IN	BYTE	Second octet of the IP address of the server to which the client attempts to connect and subsequently communicate using the Modbus application protocol.

Parameter and type		Data type	Description
IPAddr3	IN	BYTE	Third octet of the IP address of the server to which the client attempts to connect and subsequently communicate using the Modbus application protocol.
IPAddr4	IN	BYTE	Fourth octet of the IP address of the server to which the client attempts to connect and subsequently communicate using the Modbus application protocol.
IP_Port	IN	WORD	Port number of the server to which the client attempts to connect and subsequently communicate using Modbus TCP. Default: 502 Set the port to the actual port number of your device.
RW	IN	BYTE	Assigns the type of request (read or write) where 0 = Read and 1 = Write. See the Modbus functions table below for details.
Addr	IN	DWORD	Modbus starting Address: Assigns the starting address of the data to be accessed by MBUS_CLIENT. See the Modbus functions table below for details.
Count	IN	INT	Modbus data length: Number of bits or holding registers to be accessed in this request The ranges 10001 to 19999 and 30001 to 39999 are read-only addresses. For input and output bits, the maximum Count value is 1920 bits. For input and holding registers, the maximum Count value is 120 WORDS. See the Modbus functions table below for details.
DataPtr	IN_OUT	DWORD	Pointer to the Modbus data register: DataPtr points to the V memory location for the data associated with the read or write request. For read requests, this location is the first memory location at which to store the data read from the Modbus server. For write requests, this location is the first memory location of the data to be written to the Modbus server.
Done	OUT	BOOL	TRUE: One of the following conditions is true: • Client has established a connection with a server • Client is disconnected to a server • Client received a Modbus response • Error occurred FALSE: Client is busy establishing a connection or waiting for a Modbus response from the server.
Error	OUT	BYTE	Instruction execution result Valid for only a single cycle after the error occurred

RW and Addr parameters select the Modbus communication function

The MBUS_CLIENT instruction uses the RW input to indicate either a read or write function and the Addr input to define what type of data to read or write.

The following table shows the Modbus functions that the MBUS_CLIENT instruction provides based on the RW and Addr input parameters:

FC	Function	RW	Addr	Count	CPU Address
01	Read Bits	0	00001 to 09999	1 to 1920 bits	Q0.0 - Q1151.7

02	Read Bits	0	10001 to 19999	1 to 1920 bits	I0.0 - I1151.7
03	Read Words	0	40001 to 49999 400001 to 465535	1 to 120 words	V memory
04	Read Words	0	30001 to 39999	1 to 120 words	AIW0 - AIW530
05	Write Single Bit	1	00001 to 09999	1 bit	Q0.0 - Q1151.7
06	Write single Word	1	40001 to 49999 400001 to 465535	1 word	V memory
15	Write Multiple Bits	1	00001 to 09999	1 to 1920 bits	Q0.0 - Q1151.7
16	Write Multiple Words	1	40001 to 49999 400001 to 465535	1 to 120 words	V memory

Multiple client connections

A Modbus TCP client can support multiple connections up to the maximum number of Open User Communications connections that the PLC allows. The total number of connections for a PLC, including Modbus TCP clients and servers, must not exceed the maximum number of supported Open User Communications connections (Page 475). Multiple client connections must have different IPAddr or IP_Port input parameters.

Establishing a connection

When the Connect input is TRUE, the client attempts to establish a connection with the server device at the provided IP address and IP Port. If the server device is unreachable, the connection request eventually times out, which can take several seconds. While a connection request is in progress, no other operation can interrupt or abort it. If the server is unavailable, it immediately refuses the client's connection request. If the server is available, the client establishes a connection and can send a request to the server. The MBUS_CLIENT instruction returns an error if there are no connections resources available for the Modbus client.

Processing a request

The client only processes requests when Connect = TRUE. Once the client has established a connection with the server, the program makes a new request by calling MBUS_CLIENT with Req = TRUE when no Modbus request is active. When the Modbus client executes the request, it captures all of the input values. Pulse the Req input through an edge detection element (for example, Positive Edge), which causes the instruction to transmit the request one time. Any subsequent changes to the input values while the request is active result in MBUS_CLIENT returning an error code.

Once the client has sent a request to the server, the client waits for a response up to the mReceiveTimeout period of time. While the client is waiting for a response, it is not

available for other Modbus operations. If the client does not receive a response within the mReceiveTimeout period of time, MBUS_CLIENT returns an error.

If the client receives a valid response from the server, it processes any further actions depending on the response. The client then returns to a ready state and is available for additional requests from the program.

Disconnecting an established connection

If the Connect input is FALSE, the client attempts to disconnect from the server, if there is an active connection between the client and server. If there is a connect or send operation in progress the disconnect operation returns an error. A disconnect request cannot interrupt an operation. If no operation is in progress, the CPU terminates the active connection and the client returns to an idle state. The connection resource is then available to other operations in the CPU.

12.5.2.2 Modbus TCP client execution error codes

The MBUS_CLIENT instruction (Page 601) can return the following error codes:

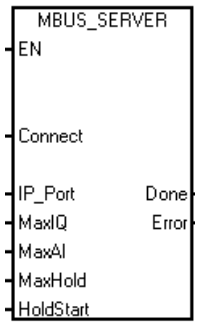
Error (decimal)	Description
0	No error
32	Unknown state Check network connections and check that the program is not modifying any library symbols that intefere with client/server communication.
33	Connection is busy with another request. A single connection can only service one Modbus request at a time.
34	Addr input is an illegal value.
35	Count input is an illegal value.
36	RW input is an illegal value.
37	Transaction ID of the request does not match the response from the server. This error indicates some problem in the server device or that the wrong server device answered the request. Received an invalid protocol ID from the server.
38	Received an invalid protocol ID from the server.
39	Number of bytes that the server sent does not match "Count" input value
40	Unit identifier of the request does not match the response from the server
41	Function code of the request does not match the response from the server
42	The data that the server sent does not match the data that a Modbus TCP write function requested
43	Receive timeout: The server did not respond within the mReceiveTimeout time. Check the connection to the Modbus server device.
44	The input values do not match the values for the active request.
45	The Modbus data register range exceeds the V memory range.

In addition to the MBUS_CLIENT errors listed above, refer also to the Modbus TCP general exception codes (Page 614) and Open User Communication error codes (Page 639)

12.5.3 Modbus TCP server

12.5.3.1 MBUS_SERVER instruction

Table 12-16 MBUS_SERVER instruction

LAD / FBD	STL	Description
	**`Call MBUS_SERVER Connect, IP_Port, MaxIQ, MaxAI, MaxHold, HoldStart, Done, Error`**	MBUS_SERVER communicates as a Modbus TCP server through the Ethernet port. MBUS_SERVER can accept a request to connect with Modbus TCP client, receive a Modbus function request, and send a response message.

Execute the MBUS_SERVER instruction in every scan in order for the Modbus server to respond to requests from Modbus clients in a reasonable amount of time. The MBUS_SERVER instruction handles establishing connections, receiving requests, and sending responses. The Modbus server cannot operate correctly unless the program calls MBUS_SERVER on every scan.

Table 12-17 Data types for the parameters

Parameter and type		Data type	Description
Connect	IN	BOOL	You use the Connect parameter to connect to or disconnect from a client device. The Modbus server attempts to create a "passive" connection, which means that the server will accept a connection request from any requesting IP address. If Connect = TRUE, and the client has not established a connection to the server, the server will passively listen for a TCP connection request. If Connect = FALSE and a connection does exist, the server initiates a disconnect operation. The program can thus use the Connect parameter to control when the server can accept a connection. When Connect = FALSE, MBUS_SERVER performs no other operation. Note that MBUS_SERVER can automatically initiate a disconnect operation when specific TCP errors occur.
IP_Port	IN	WORD	Port number of the server to which the client will attempt to connect and communicate using the Modbus application protocol. Default: 502 Set the port to the actual port number of your device
MaxIQ	IN	WORD	Parameter MaxIQ sets the number of I and Q points available to Modbus addresses 0xxxx and 1xxxx at values of 0 to 9216. A value of 0 disables all reads and writes to the inputs and outputs. The suggested value for MaxIQ is 9216.
MaxAI	IN	WORD	Parameter MaxAI sets the number of word input (AI) registers available to Modbus address 3xxxx at values of 0 to 266. A value of 0 disables reads of the analog inputs. To allow access to all of the CPU analog inputs, the suggested value for MaxAI is 266.

Parameter and type		Data type	Description
MaxHold	IN	WORD	Parameter MaxHold sets the number of word holding registers in V memory available to Modbus address 4xxxx or 4yyyyy. For example, if you want to allow Modbus client access for 2000 bytes of V memory, set MaxHold to a value of 1000 words (holding registers).
HoldStart	IN	DWORD	Parameter HoldStart is a pointer to the start of the holding registers in V memory. You typically set this value to &VB0 (address of VB0). You can set other V memory addresses as the starting address for the holding registers to allow VB0 to be used elsewhere in the project. The Modbus client has access to MaxHold number of words of V memory starting at HoldStart. If HoldStart points to a memory location that is outside the allowed range, the Modbus TCP library instruction returns an error. The CPU also generates the non-fatal error: Indirect addressing error (0x06).
Done	OUT	BOOL	TRUE: MBUS_SERVER performed one of the following actions: • Connected to a client device • Disconnected to a client • Responded to a Modbus request • Returned an error FALSE: No request serviced this program cycle
Error	OUT	BYTE	Instruction execution result Only valid for a single cycle after the error occurred

Opening a connection

When Connect = TRUE, the CPU uses a single passive connection resource from the Open User Communications available connections. Leave the Connect input TRUE while the program is expecting Modbus operations. You can set Connect to FALSE to free up a connection resource. The CPU captures the values of the input parameters when the Modbus server requests a connection. If the input values change while Connect = TRUE, MBUS_SERVER returns an error.

12.5.3.2 Modbus TCP server execution error codes

The MBUS_SERVER instruction (Page 605) can return the following error codes:

Error (decimal)	Description
0	No error
32	Unknown state Check network connections and check that the program is not modifying any library symbols that intefere with client/server communication.
33	Invalid value for input MaxIQ
34	Invalid value for input MaxAI
35	Invalid value for input MaxHold
36	HoldStart input is not in V memory or the range of holding registers exceeds the V memory range
37	Holding registers overlap Modbus server symbols
38	The input values do not match the values for the current connection. Reset the connection to update the input values.

In addition to the MBUS_SERVER errors listed above, refer also to the Modbus TCP general exception codes (Page 614) and Open User Communication error codes (Page 639)

12.5.4 Example: Modbus TCP application

The following example consists of a project with two Modbus TCP clients communication with two Modbus TCP servers. A unique IP Address identifies each server. The program logic monitors the Done outputs of the MBUS_CLIENT instructions to ensure that the program does not interrupt a communication request that is in progress. This example program performs the following functions:

- Write output bits
- Read output bits
- Write holding registers
- Read holding registers

The program, network, and symbol comments describe the functionality of the Modbus TCP example program in the following table.

The basic description for this example:

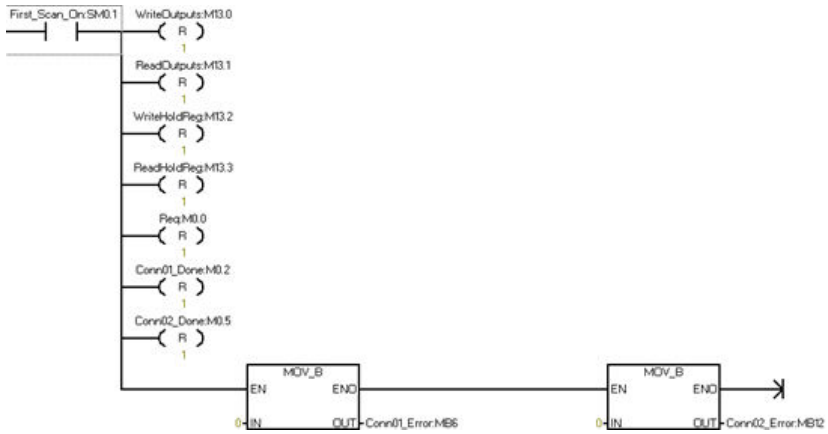
Two Modbus clients establish connections with two Modbus server devices.

Modbus server 01: IP Address 192.168.2.10, Port 502

Modbus server 02: IP Address 192.168.2.66, Port 502

When CPU Input 0 transitions to TRUE, a sequence starts in which the Modbus client sends write and read requests to both Modbus servers. CPU Input 0 set False turns off the sequence.

LAD

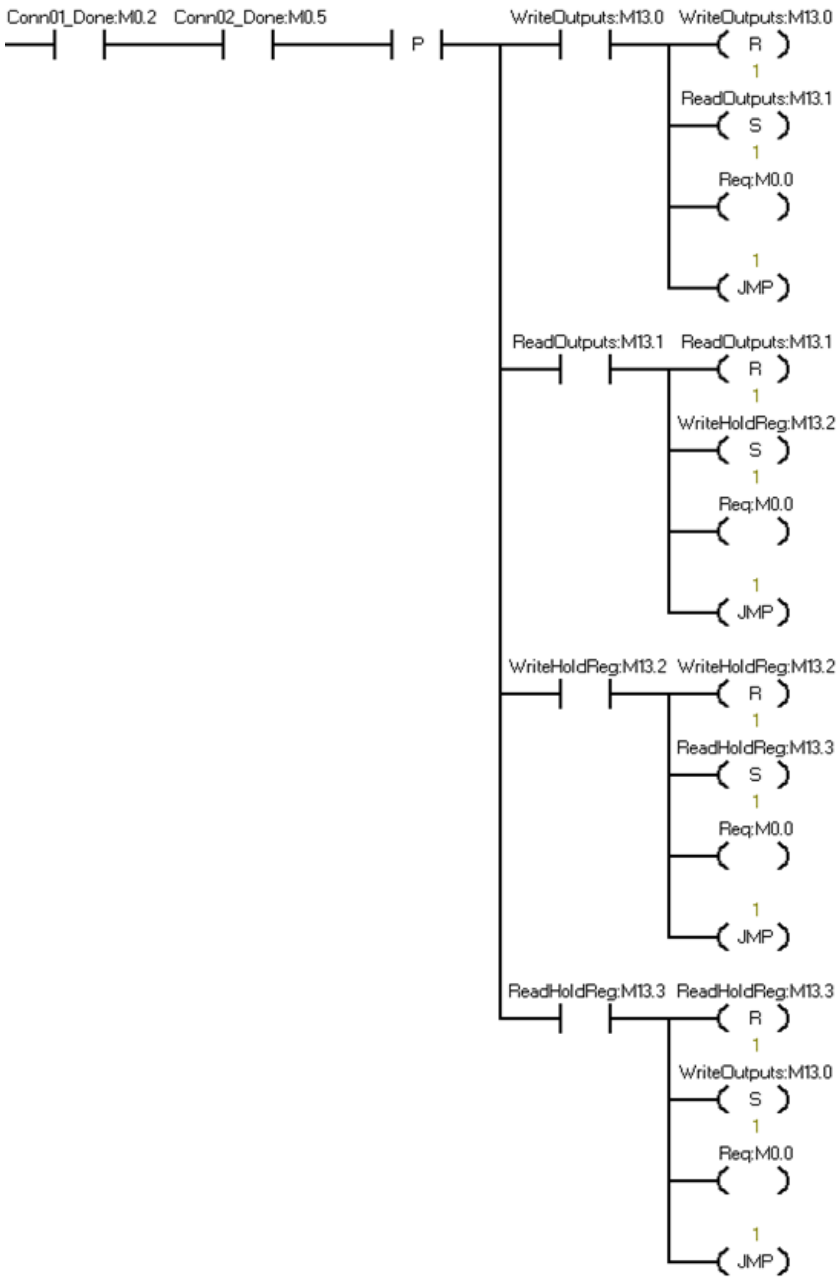


Symbol	Address	Comment
Conn01_Done	M0.2	Connection #1 has completed the current operation
Conn01_Error	MB6	Connection #1 Status
Conn02_Done	M0.5	Connection #2 has completed the current operation
Conn02_Error	MB12	Connection #2 Status
First_Scan_On	SM0.1	ON for the first scan cycle only
ReadHoldReg	M13.3	Initiate a Read Holding Register request
ReadOutputs	M13.1	Initiate a Read Output request
Req	M0.0	Initiate a Modbus request
WriteHoldReg	M13.2	Initiate a Write Holding Register request
WriteOutputs	M13.0	Initiate a Write Output request

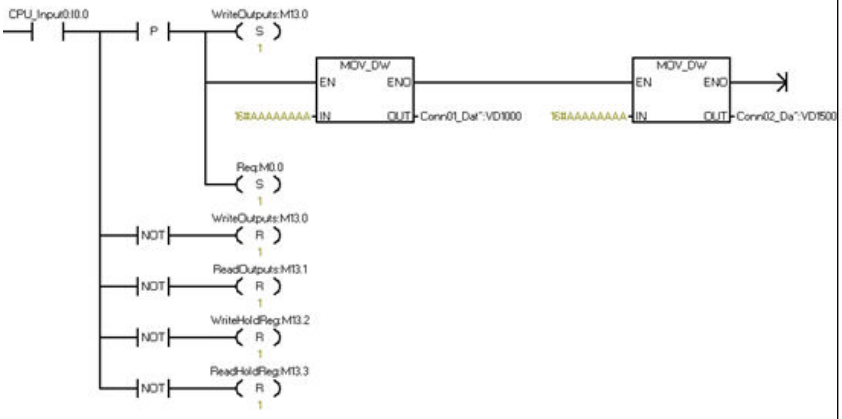
Description

Network 1:

On startup, clear all the flags and errors.

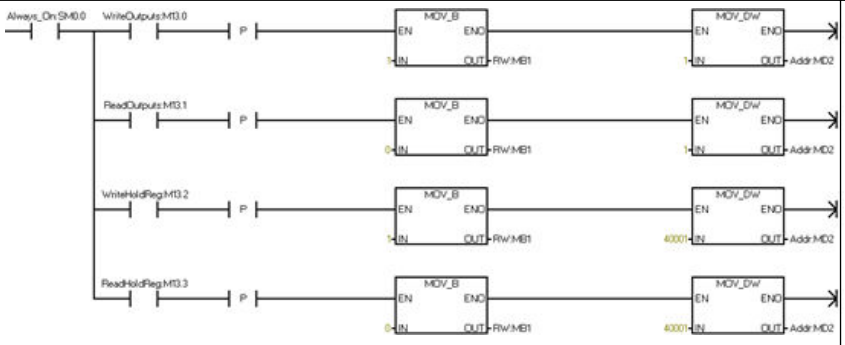
LAD	Description
	Network 2: When both clients complete the Modbus requests, start the next Modbus request.

<table>
<tr><th>LAD</th><th>Description</th></tr>
<tr><td>
<table>
<tr><th>Symbol</th><th>Address</th><th>Comment</th></tr>
<tr><td>Conn01_Done</td><td>M0.2</td><td>Connection #1 has completed the current operation</td></tr>
<tr><td>Conn02_Done</td><td>M0.5</td><td>Connection #2 has completed the current operation</td></tr>
<tr><td>ReadHoldReg</td><td>M13.3</td><td>Initiate a Read Holding Register request</td></tr>
<tr><td>ReadOutputs</td><td>M13.1</td><td>Initiate a Read Output request</td></tr>
<tr><td>Req</td><td>M0.0</td><td>Initiate a Modbus request</td></tr>
<tr><td>WriteHoldReg</td><td>M13.2</td><td>Initiate a Write Holding Register request</td></tr>
<tr><td>WriteOutputs</td><td>M13.0</td><td>Initiate a Write Output request</td></tr>
</table>
</td><td></td></tr>
<tr><td>

</td><td>Network 3</td></tr>
<tr><td>


<table>
<tr><th>Symbol</th><th>Address</th><th>Comment</th></tr>
<tr><td>Conn01_Data</td><td>VD1000</td><td>Connection #1 Data Location</td></tr>
<tr><td>Conn02_Data</td><td>VD1500</td><td>Connection #2 Data Location</td></tr>
<tr><td>CPU_Input0</td><td>I0.0</td><td></td></tr>
<tr><td>ReadHoldReg</td><td>M13.3</td><td>Initiate a Read Holding Register request</td></tr>
<tr><td>ReadOutputs</td><td>M13.1</td><td>Initiate a Read Output request</td></tr>
<tr><td>Req</td><td>M0.0</td><td>Initiate a Modbus request</td></tr>
<tr><td>WriteHoldReg</td><td>M13.2</td><td>Initiate a Write Holding Register request</td></tr>
<tr><td>WriteOutputs</td><td>M13.0</td><td>Initiate a Write Output request</td></tr>
</table>
</td><td>Network 4:
CPU_Input 0 rising edge triggers the Modbus request sequence to begin Write some data in V memory to send to the Modbus servers.
Set the Req input bit TRUE.
When the CPU_Input 0 is False, stop sending Modbus requests.</td></tr>
</table>

LAD



Symbol	Address	Comment
Addr	MD2	Modbus Address
Always_On	SM0.0	Always ON
ReadHoldReg	M13.3	Initiate a Read Holding Register request
ReadOutputs	M13.1	Initiate a Read Output request
RW	MB1	0 = Read Data; 1 = Write Data
WriteHoldReg	M13.2	Initiate a Write Holding Register request
WriteOutputs	M13.0	Initiate a Write Output request

Description

Network 5:

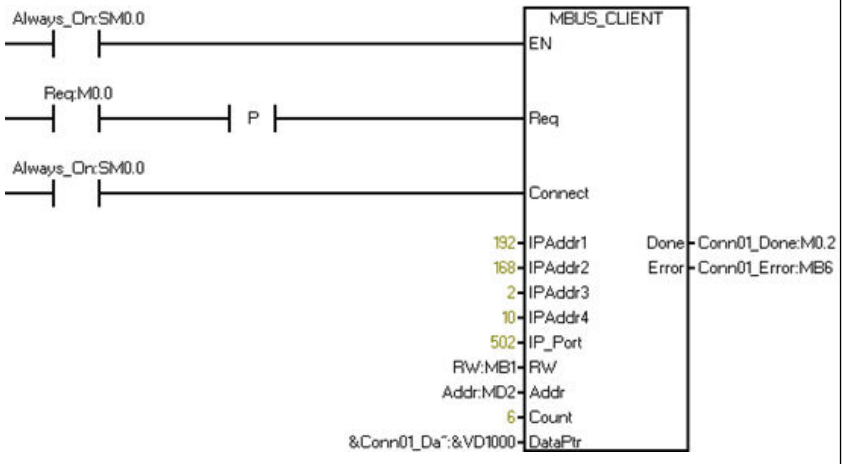
Set the Read/Write mode and the address for the selected Modbus function.

Write Ouputs = Function Code 5 (single)/ 15 (multiple).

Read Outputs = Function Code 1.

Write Holding Registers = Function Code 6 (single) / 16 (multiple).

Read Holding Registers = Function Code 3.



Symbol	Address	Comment
Addr	MD2	Modbus Address
Always_On	SM0.0	Always ON
Conn01_Data	VD1000	Connection #1 Data Location
Conn01_Done	M0.2	Connection #1 has completed the current operation
Conn01_Error	MB6	Connection #1 Status
Req	M0.0	Initiate a Modbus request
RW	MB1	0 = Read Data; 1 = Write Data

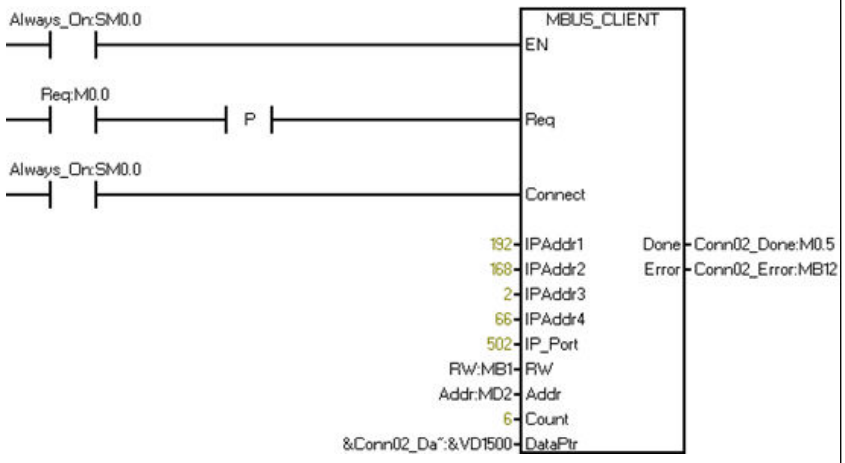
Network 6:

Modbus client 01 establishes a connection with Modbus server 01.

When Req is TRUE, send a Modbus Request to the server.

Once the Modbus Client receives and processes the response from the server, the MBUS_CLIENT instruction sets the Done output to TRUE.

LAD



Symbol	Address	Comment
Addr	MD2	Modbus Address
Always_On	SM0.0	Always ON
Conn02_Data	VD1500	Connection #2 Data Location
Conn02_Done	M0.5	Connection #2 has completed the current operation
Conn02_Error	MB12	Connection #2 Status
Req	M0.0	Initiate a Modbus request
RW	MB1	0 = Read Data; 1 = Write Data

Description

Network 7:

Modbus client 02 establishes a connection with Modbus server 02.

When Req is TRUE, send a Modbus Request to the server.

Once the Modbus Client receives and processes the response from the server, the MBUS_CLIENT instruction sets the Done output to TRUE.

12.5.5 Modbus TCP advanced user information

Overview

This topic contains information for advanced users of the Modbus TCP library. Most users do not need this information and will not need to modify the default operation of the Modbus TCP library.

MBUS_CLIENT variables

The table below shows some Modbus client variables that you can modify in your program to adjust the operation of the Modbus client if the default value is not suitable for your application:

Variable	Data type	Default	Description
mBlocked_Proc_Timeout	REAL	3000	Blocked process timeout: Amount of time (in milliseconds) to wait upon a blocked Modbus client instance before removing this instance as being ACTIVE. This can occur, for example, when the program has issued a client request and the application stops executing the client function before completely finishing the request.
mModbus_Unit_ID	WORD	255	Modbus unit identifier: The mModbus_Unit_ID parameter corresponds to the slave address in the Modbus RTU protocol. If a Modbus TCP server is used for a gateway to a Modbus RTU protocol, the MB_UNIT_ID can be used to identify the slave device connected on the serial network. The MB_UNIT_ID would be used to forward the request to the correct Modbus RTU slave address. Some Modbus TCP devices may require the MB_UNIT_ID parameter to be within a restricted range.
mReceiveTimeout	REAL	2000	Receive message timeout: Time in milliseconds that the MBUS_CLIENT waits for a server to respond to a request. Range: 500 - 65,535 milliseconds.
mConnected	BOOL	FALSE	Connection State: Indicates whether the connection to the assigned server is connected or disconnected: TRUE: Connected FALSE: Disconnected The program can check mConnected after processing an MBUS_CLIENT request.
mRetries	BYTE	3	Retries: Number of times a client attempts to disconnect and resend the request after an initial request returns with a connection error Range: 0 to 255

Retries

The Modbus client instruction automatically restarts the connection and resends the request to the server device if there is a connection-related error:

The Modbus client resends the request two additional times before setting the Done and Error output parameters.

You can change the number of retries by finding the symbol mModbusRetries in the Modbus client symbol table and changing the value before the program executes MBUS_CLIENT. The mRetries value is a BYTE with a range of 0 to 255 retries.

Single vs. multiple bit / word write functions

Some Modbus server devices do not support the Modbus functions to write a single discrete output bit (Modbus function 5) or to write a single holding register (Modbus function 6). These devices only support the multiple bit write (Modbus function 15) or multiple register write (Modbus function 16) instead. The MBUS_CLIENT instruction returns an error code 1 if the server device does not support the single bit/word Modbus functions.

The Modbus client protocol allows you to force the MBUS_CLIENT instruction to use the multiple bit/word Modbus functions instead of the single bit/word Modbus functions. You can force the multiple bit/word instructions by finding the symbol mModbusForceMulti in the Modbus client symbol table and changing this value before the program executes MBUS_CLIENT. Set mModbusForceMulti to TRUE to force the use of the multiple bit/word functions when writing a single bit or register.

Holding register addresses greater than 49999

Modbus holding addresses are within the range of 40001 to 49999. This range is adequate for most applications but there are some Modbus slave devices with data mapped into holding registers at a higher address range.

The MBUS_CLIENT instruction allows an additional range for the parameter **Addr** to support an extended range of holding register addresses at addresses 400001 to 465536.

For example, to access holding register 16768, set the **Addr** parameter of MBUS_CLIENT to 416768.

The extended addressing allows access to the full range of 65536 possible addresses supported by the Modbus protocol. This extended addressing is only applicable for holding registers.

MBUS_SERVER variables

The table below shows some Modbus server variables that you can modify in your program to adjust the operation of the Modbus server if the default value is not suitable for your application:

Variable	Data type	Default	Description
mConnected	BOOL	0	Connection state: Indicates whether the connection to the assigned client is connected or disconnected: TRUE: Connected FALSE: Disconnected The connection state is up to date after every MBUS_SERVER instruction execution.

12.5.6 Modbus TCP general exception codes

Error #	Description
1	Modbus Exception (code 0x01): Illegal Function - Server does not support requested function
2	Modbus Exception (code 0x02): Illegal Data Address - The requested address range of Addr plus Count is outside the allowed address range of the Server
3	Modbus Exception (code 0x03): Illegal Data Value - There is an error in the Modbus protocol received by the Server
4	Modbus Exception (code 0x04): Server Device Failure - An unrecoverable error occurred while the Server was attempting to perform the requested action

5	Modbus Exception (code 0x05): Acknowledge - Response from Server may be delayed; resend the request at a later time
6	Modbus Exception (code 0x06): Server Device Busy - Server rejected the message; resend request
7	Modbus Exception (code 0x07): Negative Acknowledgement - Server rejected the message for an unknown reason
10	Modbus Exception (code 0x0A): Gateway Path Unavailable - Usually means that the gateway is misconfigured or overloaded. (Modbus TCP only)
11	Modbus Exception (code 0x0B): Gateway Target Device Failed to Respond - Usually means that the device is not present on the network. (Modbus TCP only)

12.5.7 Modbus TCP general communication exception codes

The Modbus TCP communication exception codes are as follows:

Error code	Description
161	The data length parameter is greater than the maximum allowed (1024 bytes).
162	The data buffer is not in I, Q, M, or V memory areas.
163	The data buffer does not fit in the memory area.
164	The table parameter does not fit into the memory area.
165	The connection is locked in another context. You are attempting to access the same connection in both the background (the Main) and in an interrupt routine at the same time.
166	A UDP IP address or port error
167	An instance mismatch: The connection is busy with another instance or the input data does not match the data stored for the requested connection ID when the request was initiated.
168	The Connection ID does not exist because the connection has never been created, or the connection was terminated at your request (using the TDCON instruction).
169	A TCON operation is in progress with this Connection ID.
170	A TDCON operation is in progress with this Connection ID.
171	A TSEND instruction is in progress with this Connection ID.
172	A temporary communication error has occurred. The connection cannot be started at this time. Try again later.
173	The connection partner refused or actively dropped the connection (the partner issued a disconnect to this CPU).
174	The connection partner cannot be reached (no answer to the connect request).
175	The connection aborted due to inconsistencies. Disconnect and reconnect to correct the situation.
176	The Connection ID is already in use with a different IP address, port, or TSAP combination.
177	No connection resource is available. All connections of the requested type (active/passive) are in use.
178	The local or remote port number is reserved or the port number is already in use for another server (passive) connection.
179	One of the following IP address errors have occurred: • The IP address is invalid (for example, address 0.0.0.0). • This IP address is the IP address of this CPU. • This CPU has IP address 0.0.0.0. • The IP address is a broadcast or multicast address.

Error code	Description
180	A local or remote TSAP error (ISO-on-TCP only)
181	An invalid connection ID (65535 is reserved)
182	An active/passive error (UDP only allows passive)
183	The connection type is not one of the allowed types.
184	There is no operation pending so there is no status to report.
185	The receive buffer is too small: The CPU received more bytes than the buffer length supports. The CPU discards the extra bytes.
191	Unknown error

12.6 Open user communication library

12.6.1 Overview

The STEP 7-Micro/WIN SMART Open User Communication (OUC) library instructions create the tables required by the OUC instructions (Page 285) (TCON, TSEND, TRECV, and TDCON). The library instructions build the table as needed, call the OUC instruction, and then present you the status values on the outputs of the library instruction. The CPU uses library memory to create a table to pass to the OUC instructions. The Open User Communication library requires 50 bytes of V memory.

The library instructions are as follows:

- TCP_CONNECT: Create a TCP connection.
- ISO_CONNECT: Create an ISO-on-TCP connection.
- UDP_CONNECT: Create a UDP connection.
- TCP_SEND: Send data instruction for TCP and ISO-on-TCP connections.
- TCP_RECV: Receive data instruction for TCP and ISO-on-TCP connections.
- UDP_SEND: Send data instruction for UDP connections.
- UDP_RECV: Receive data instruction for UDP connections.
- DISCONNECT: Terminate the connection for all protocols.

Note

Only call the library functions from either the main program or from interrupt routines, but not both.

12.6.2 Parameters common to the OUC library instructions

The following parameters are common to the OUC library instructions:

- EN: You set the EN input to TRUE to call the instruction. You must set the EN input to TRUE until the instruction completes (until either Done or Error is set). The CPU only updates the outputs when your program sets EN and calls the instruction.
- Req: You use the Req (Request) input to initiate the operation. The Req input bit is level-triggered. You should connect the Req input to the library instruction with a positive edge instruction so that the action is initiated only one time. The program ignores the Req input while the instruction is Busy.
- Active: The Active input commands the connect instructions whether to create an active client connection (Active = TRUE) or a passive server connection (Active = FALSE). An active connection is one in which the local CPU initiates communications to the remote device. A passive connection is one in which the local CPU waits for the remote device to initiate communications.
  The S7-200 SMART CPUs support eight active connections and eight passive connections for Open User Communication. UDP connections are counted as passive connections since there is no active communication establishment.
- Done: The OUC instruction sets the Done output when the operation is complete and there are no errors. If the instruction sets the Done output, Busy, Error, and Status outputs are zero. Other outputs (for example, the number of received bytes) are valid only when the Done output is set.
- Busy: The Busy output indicates that the operation is in progress. The OUC instruction sets the Busy output when you initiate an operation with Req set to TRUE. The Busy output remains set for all subsequent calls to the instruction until the operation is complete.
- Error: The Error output indicates that the operation completed with an error. If the OUC instruction sets the Error output, then the Done and Busy outputs are set to FALSE. If the OUC instruction sets the Error output, the Status output indicates the reason for the error. All other outputs are not valid if the Error output is set.
- ConnID: The ConnID number is the identifier for the connection. You establish the ConnID when you create the connection with TCP_CONNECT, ISO_CONNECT, or UDP_CONNECT. You can pick any value in the range 0 to 65534 for the ConnID. Each connection must have a unique ConnID. The program uses the ConnID to specify the desired connection for subsequent send, receive, and disconnect operations.

- IPaddr1, IPaddr2, IPaddr3 and IPaddr4: These are the four IP address octets of the remote device. IPaddr1 is the most significant byte of the IP address, and IPaddr4 is the least significant byte of the IP address. For example: For the IP address 192.168.2.15, set these values:
  - IPaddr1 = 192
  - IPaddr2 = 168
  - IPaddr3 = 2
  - IPaddr4 = 15

  The IP address cannot be any of the following values:
  - 0.0.0.0 (for an active connection)
  - Any broadcast IP address (for example, 255.255.255.255)
  - Any multicast address
  - The IP address of the local CPU

  You can use the IP address of 0.0.0.0 for a passive connection. By selecting the IP address of 0.0.0.0, the S7-200 SMART CPU accepts a connection from any remote IP address. Selecting a non-zero IP address for a passive connection causes the CPU to only accept a connection from the specified address.
- RemPort: The RemPort is the port number on the remote device. You use port numbers for TCP and UDP protocols to route the message within the device.
  The rules for remote port numbers are as follows:
  - The valid port number range is 1 to 49151.
  - The suggested range for port numbers is 2000 to 5000.
  - The CPU ignores the remote port number for passive connections (You can set it to zero).
- LocPort: The LocPort parameter is the port number on the local CPU. You use port numbers for TCP and UDP protocols to route the message within the device. The local port number must be unique for all passive connections.
  The rules for local port numbers are as follows:
  - The valid port number range is 1 to 49151.
  - You cannot use port numbers 20, 21, 25, 80, 102, 135, 161, 162, 443, and 34962 to 34964. These ports have specific assignments.
  - The suggested range for port numbers is 2000 to 5000.
  - The local port numbers must be unique for passive connections (no duplicates).

- RemTsap: The RemTsap (remote Transport Service Access Point (TSAP)) parameter is a pointer to an S7-200 SMART string data type. You can only use the RemTsap parameter for the ISO-on-TCP protocol. The remote TSAP string serves the same purpose as a port number in routing the message to the proper connection.
  The rules for the RemTsap are as follows:
  - The TSAP is an S7-200 SMART string data type (a length byte followed by the characters).
  - The TSAP string must be at least 2 characters and no more than 16 characters.
- LocTsap: The LocTsap (local Transport Service Access Point (TSAP)) parameter is a pointer to an S7-200 SMART string data type. You can only use the local TSAP parameter for the ISO-on-TCP protocol. The local TSAP string serves the same purpose as a port number in routing the message to the proper connection.
  The rules for the LocTsap are as follows:
  - The TSAP is an S7-200 SMART string data type (a length byte followed by the characters).
  - The TSAP string must be at least 2 characters and no more than 16 characters.
  - If the TSAP is 2 characters, the first character must be a hexadecimal "E0".
  - The TSAP cannot start with the string "SIMATIC-".

12.6.3 Open user communication library instructions

12.6.3.1 TCP_CONNECT instruction

The TCP_CONNECT instruction creates a connection to another device using the TCP protocol.

LAD/FBD	STL	Description
	TCP_CONNECT Req, Active, ConnID, IPaddr1, IPaddr2, IPaddr3, IPaddr4, RemPort, LocPort, Done, Busy, Error, Status	The TCP_CONNECT creates a TCP communications connection from the CPU to a communication partner.

The connect operation is asynchronous and can take several scans to complete. The Busy output has a value of TRUE while the connect operation is pending. When the CPU completes the operation, the instruction sets either the Done or Error output. If there is an error, the Status output contains the error code.

You must not change the input parameters of the TCP_CONNECT while the instruction is busy. The CPU requires this so that it knows that this is a continuation of the call that started the connection process.

You assign the connection ID (ConnID) input to the connection and then use this ConnID to reference this connection when sending, receiving, or disconnecting.

The Active input bit determines whether this is an active connection (Active set to TRUE) or a passive connection (Active set to FALSE).

If this is an active connection (a client), then the S7-200 SMART CPU attempts to contact and create a connection to the specified IP address and remote port number (RemPort). The CPU opens a local port (LocPort) to receive messages from the remote device.

When the Active input is set to FALSE, the S7-200 SMART CPU creates a passive (server) connection. In this case, the CPU opens the requested local port (LocPort) and accepts connection requests from a remote device. You should set the IP address to 0.0.0.0 if you wish to accept a connection request from any remote IP address. If the IP address is non-zero, the CPU accepts a connection request only from the specified IP address. The CPU ignores the remote port number (RemPort) for a passive connection, and RemPort can be set to zero.

You can call the TCP_CONNECT instruction anytime to determine the current status of a connection. Set the Req input to FALSE and provide a valid connection ID (ConnID), and TCP_CONNECT returns the following:

- Busy if the connect process is still progressing.
- Done if the connection is active and ready to send or receive.
- Error if the connection is not usable. Status contains one of the error codes to specify the problem.

Note that active connections can require up to 30 seconds to determine if the remote device will allow the connection or not. Passive connections show a Busy status until a remote device attempts to connect to the CPU.

Note that the S7-200 SMART does not automatically attempt to reconnect to a device after the connection has been closed. If the remote device breaks the device connection, your program must execute another TCP_CONNECT instruction to reconnect the device. This is true for both active and passive connections.

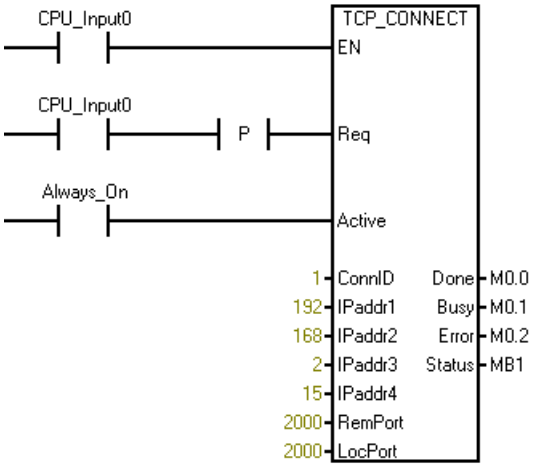
Table 12-18 Parameters of the TCP_CONNECT instruction

Parameter	Declaration	Data type	Description
EN	IN	BOOL	Enable input
Req	IN	BOOL	The CPU starts the connect operation if Req = TRUE. If Req = FALSE, then the outputs show the current status of the connection.
Active	IN	BOOL	• TRUE = active connection • FALSE = passive connection
ConnID	IN	WORD	The CPU uses the Connection ID (ConnID) number to identify this connection for the other instructions. The possible ConnID range is 0 to 65534.

Parameter	Declaration	Data type	Description
IPaddr1 ... IPaddr4	IN	BYTE	These are the four IP address octets. IPaddr1 is the most significant byte and IPaddr4 is the least significant byte of the IP address.
RemPort	IN	WORD	The RemPort is the port number on the remote device. The remote port number range is 1 to 49151. Use zero for passive connections.
LocPort	IN	WORD	The LocPort is the port number on the local device. The local port number range is 1 to 49151 with some restrictions. Refer to the LocPort definition in "Parameters common to the OUC library instructions" (Page 617).
Done	OUT	BOOL	The instruction sets the Done output when the connect operation is complete with no errors.
Busy	OUT	BOOL	The instruction sets the Busy output while the connection operation is in progress.
Error	OUT	BOOL	The instruction sets the Error output when the connection operation is complete with an error. Refer to "Open user communication library instruction error codes" (Page 639) for further information.
Status	OUT	BYTE	The Status output shows the error code if the instruction sets the Error output. Status is zero (no error) if the instruction sets the Busy or Done outputs.

Example

This is an example usage of the TCP_CONNECT instruction:



12.6.3.2 ISO_CONNECT instruction

The ISO_CONNECT instruction creates a connection to another device using the ISO-on-TCP protocol. This protocol uses RFC1006 in addition to the TCP protocol to better delineate messages. The advantage of ISO-on-TCP is that every sent message results in a different received message. The ISO-on-TCP protocol never combines multiple received messages into one message, as can happen with TCP protocol. The ISO-on-TCP protocol uses TSAPs (Transport Services Access Point) to route the message in the device instead of ports.

LAD/FBD	STL	Description
	ISO_CONNECT Req, Active, ConnID, IPaddr1, IPaddr2, IPaddr3, IPaddr4, RemTsap, LocTsap, Done, Busy, Error, Status	The ISO_CONNECT creates an ISO-on-TCP communications connection from the CPU to a communication partner.

The connect operation is asynchronous and can take several scans to complete. The instruction sets the Busy output while the connect operation is pending. When the CPU completes the operation, the instruction sets either the Done or Error output. If there is an error, the Status output contains the error code.

You must not change the input parameters of the ISO_CONNECT while the instruction is busy. The CPU requires this so that it knows that this is a continuation of the call that started the connection process.

You assign the connection ID (ConnID) input to the connection and then use this ConnID to reference this connection when sending, receiving, or disconnecting.

The Active input bit determines whether this is an active connection (Active set to TRUE) or a passive connection (Active set to FALSE).

If this is an active connection (a client), then the S7-200 SMART CPU attempts to contact and create a connection to the specified IP address and remote TSAP (RemTsap). The CPU opens a local TSAP (LocTsap) to receive messages from the remote device.

When the Active input is FALSE, the S7-200 SMART CPU creates a passive (server) connection. In this case, the CPU opens the requested local TSAP (LocTsap) and accepts connection requests from a remote device. You should set the IP address to 0.0.0.0 if you wish to accept a connection request from any remote IP address. If the IP address is non-zero, the CPU accepts a connection request only from the specified IP address. The CPU ignores the remote TSAP string (RemTsap) for passive connections, and the RemTsap can be set to an empty string (for example, "").

You can call the ISO_CONNECT instruction anytime to determine the current status of a connection. Set the Req input to FALSE and provide a valid connection ID (ConnID), and ISO_CONNECT returns the following:

- Busy if the connect process is still progressing.
- Done if the connection is active and ready to send or receive.
- Error if the connection is not usable. Status contains one of the error codes to specify the problem.

Note that active connections can require up to 30 seconds to determine if the remote device will allow the connection or not. Passive connections show a Busy status until a remote device attempts to connect to the CPU.

Note that the S7-200 SMART does not automatically attempt to reconnect to a device after the connection has been closed. If the remote device breaks the device connection, your program must execute another ISO_CONNECT instruction to reconnect the device. This is true for both active and passive connections.

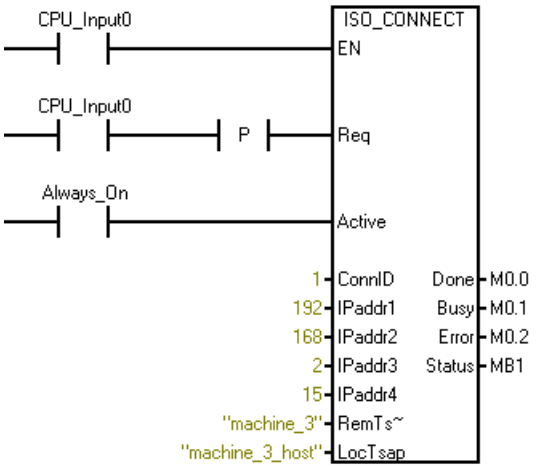
Table 12-19 Parameters of the ISO_CONNECT instruction

Parameter	Declaration	Data type	Description
EN	IN	BOOL	Enable input
Req	IN	BOOL	The CPU starts the connect operation if Req = TRUE. If Req = FALSE, then the outputs show the current status of the connection.
Active	IN	BOOL	• TRUE = active connection • FALSE = passive connection
ConnID	IN	WORD	The CPU uses the Connection ID (ConnID) number to identify this connection for the other instructions. The possible ConnID range is 0 to 65534.
IPaddr1 ... IPaddr4	IN	BYTE	These are the four IP address octets. IPaddr1 is the most significant byte and IPaddr4 is the least significant byte of the IP address.
RemTsap	IN	String<254>	The RemPort is the remote TSAP string. The program uses a pointer to pass the string. (Refer to the example following this table for more information.)
LocTsap	IN	String<254>	The LocPort is the local TSAP string. The program uses a pointer to pass the string. (Refer to the example following this table for more information.)
Done	OUT	BOOL	The instruction sets the Done output when the connect operation is complete with no errors.
Busy	OUT	BOOL	The instruction sets the Busy output while the connection operation is in progress.

Parameter	Declaration	Data type	Description
Error	OUT	BOOL	The instruction sets the Error output when the connection operation is complete with an error. Refer to "Open user communication library instruction error codes" (Page 639) for further information.
Status	OUT	BYTE	The Status output shows the error code if the instruction sets the Error output. Status is zero (no error) if the instruction sets the Busy or Done outputs.

Example

This is an example usage of the ISO_CONNECT instruction:

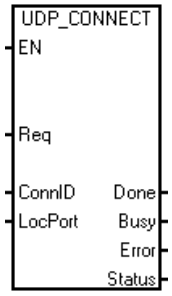


STEP7-Micro/WIN SMART always uses pointers to pass a string to the ISO_CONNECT instruction for the RemTsap and the LocTsap. If you use a constant string (as in the example above), STEP7-Micro/WIN SMART automatically creates the string and the pointer. If you wish to create strings in a Data Block and then pass a pointer to one of these strings, you would follow these steps:

1. In the Data Block, create the strings:
   - VB100 "machine_1"
   - VB120 "machine_2"
2. Use "&VB100" or "&VB120" (without the quote marks) for the TSAP parameters on the ISO_CONNECT instruction.

12.6.3.3 UDP_CONNECT instruction

The UDP_CONNECT instruction creates a passive connection using UDP protocol. UDP is a connectionless protocol so there is no actual connection created between this CPU and the remote device. The UDP connection opens the selected local port to use with UDP protocol.

LAD/FBD	STL	Description
	**UDP_CONNECT Req, ConnID, LocPort, Done, Busy, Error, Status**	The UDP_CONNECT creates a passive connection using UDP protocol to open a selected local port.

The UDP_CONNECT instruction only requires a connection ID and a local port number to create the connection. One UDP connection can send messages to any number of other devices since the IP address and remote port are supplied with each UDP_SEND instruction. You will need multiple UDP connections only if you require multiple local ports. You cannot use the same local port number for multiple UDP connections. All local port numbers must be unique.

The connect operation is asynchronous and can take several scans to complete. There is no active connection establishment with the remote device, nor are we waiting for another device to connect to this CPU. The Busy output is set while the connect operation is pending. When the connect operation is complete, the program sets the Done output. The program sets the Error output only if there is a problem with the input parameters or there is no passive connection available. The Status output byte contains the error code if the program sets the Error bit.

You must not change the parameters of the UDP_CONNECT while the instruction is busy so that the CPU knows this is a continuation of the call that started the connection process.

You can call the UDP_CONNECT instruction to determine the current status of a connection. Set the Req input FALSE and provide a valid connection ID (ConnID), and the UDP_CONNECT instruction returns the following:

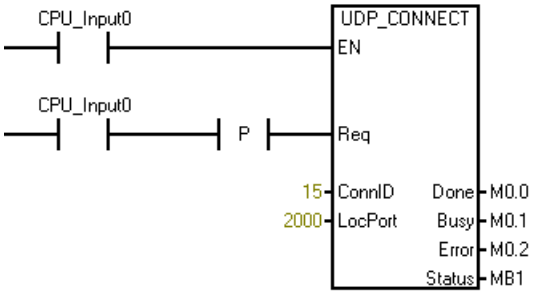
- The instruction sets the Done output if the connection is active and ready to send or receive. This only means that you can use the connection. This is not an indication that there is any remote device present.
- The instruction sets the Busy output if the connect is still processing.
- The instruction sets the Error output if the connection is not usable. The Status output byte contains one of the error codes to specify the problem.

Table 12-20 Parameters of the UDP_CONNECT instruction

Parameter	Declaration	Data type	Description
EN	IN	BOOL	Enable input
Req	IN	BOOL	The CPU starts the connect operation if Req = TRUE. If Req = FALSE, then the outputs show the current status of the connection.
ConnID	IN	WORD	The CPU uses the Connection ID (ConnID) number to identify this connection for the other instructions. The possible ConnID range is 0 to 65534.
LocPort	IN	WORD	The LocPort is the port number on the local device. The local port number range is 1 to 49151 with some restrictions. Refer to the LocPort definition in "Parameters common to the OUC library instructions" (Page 617).
Done	OUT	BOOL	The instruction sets the Done output when the connect operation is complete with no errors.
Busy	OUT	BOOL	The instruction sets the Busy output while the connection operation is in progress.
Error	OUT	BOOL	The instruction sets the Error output when the connection operation is complete with an error. Refer to "Open user communication library instruction error codes" (Page 639) for further information.
Status	OUT	BYTE	The Status output shows the error code if the instruction sets the Error output. Status is zero (no error) if the instruction sets the Busy or Done outputs.

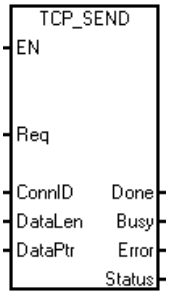
Example

This is an example usage of the UDP_CONNECT instruction:



12.6.3.4 TCP_SEND instruction

The TCP_SEND instruction transmits the requested number of bytes (DataLen) from the requested buffer location (DataPtr) over the existing connection (ConnID). You use the instruction for both TCP protocol and for ISO-on-TCP protocol.

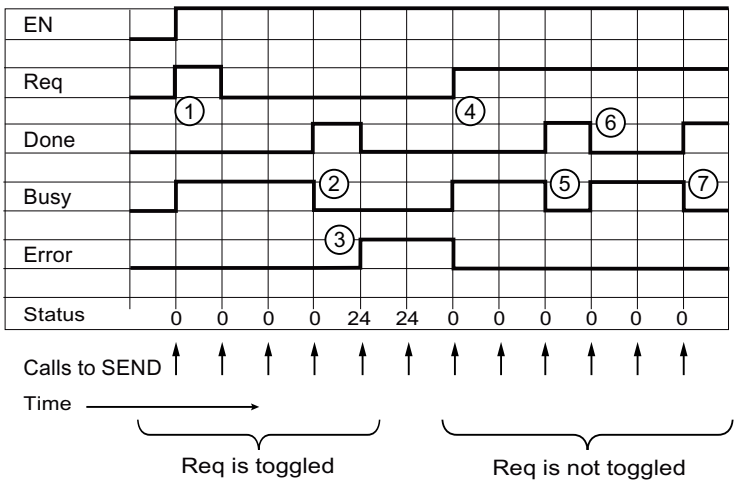
LAD/FBD	STL	Description
	**TCP_SEND Req, ConnID, DataLen, DataPtr, Done, Busy, Error, Status**	The TCP_SEND transmits the requested number of bytes from the requested buffer location over an existing connection.

The TCP_SEND instruction initiates sending the specified number of bytes when the following occur:

- The program calls the instruction with the Req input set to TRUE.
- The connection is not currently busy with another send operation.

The Req input is level-triggered. It is recommended that you put a positive edge trigger on the Req input so that the instruction does not initiate unintended send operations. The program ignores the Req input while the TCP_SEND is busy. The Done, Busy, and Error outputs and the Status output byte show the status of the TCP_SEND for each call.

The instruction displays the Done or Error status for one call of TCP_SEND after the send operation is complete. After that, the TCP_SEND responds with error code 24, which means no operation pending, if called with the Req input set to FALSE. If the Req input is left set to TRUE, the program initiates another send operation. The figure below shows the relationship of the input and output parameters.



① Req is set TRUE so that the message send begins. Busy is set TRUE.

② The message send is complete. Done is set, and Busy is cleared.

③ EN is TRUE, and Req is FALSE, but no message send is in progress. So, Error is set with error code 24.

④ Req is set TRUE again, so another message send begins. Busy is set TRUE.

⑤ The message send is complete. Done is set, and Busy is cleared for one scan.

⑥ Req remains TRUE, so another message send begins.

⑦ The message send is complete.

The maximum amount of data that you can send in one send operation is 1024 bytes. The program copies the data from the send buffer in user memory to an internal buffer when the TCP_SEND executes with the Req input set TRUE. You can change the program send buffer after the TCP_SEND executes and the instruction sets the Busy output.

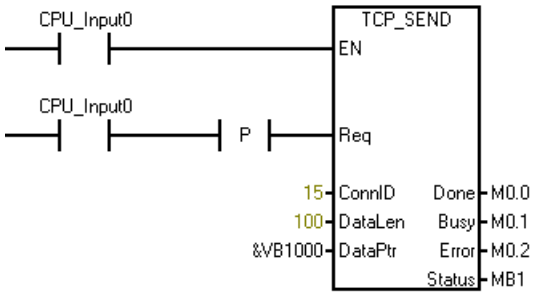
Table 12-21 Parameters of the TCP_SEND instruction

Parameter	Declaration	Data type	Description
EN	IN	BOOL	Enable input
Req	IN	BOOL	The CPU starts the send operation if Req = TRUE. If Req = FALSE, then the outputs show the current status of the send operation.
ConnID	IN	WORD	The Connection ID (ConnID) is the number of the connection for this send operation. Use the ConnID that you selected for the TCP_CONNECT operation.
DataLen	IN	WORD	The DataLen is the number of bytes to transmit (1 to 1024).
DataPtr	IN	DWORD	The DataPtr is the pointer to the data to be sent. This is an S7-200 SMART pointer to I, Q, M, or V memory (for example, &VB100).
Done	OUT	BOOL	The instruction sets the Done output when the send operation is complete with no errors.
Busy	OUT	BOOL	The instruction sets the Busy output while the send operation is in progress.

Parameter	Declaration	Data type	Description
Error	OUT	BOOL	The instruction sets the Error output when the send operation is complete with an error. Refer to "Open user communication library instruction error codes" (Page 639) for further information.
Status	OUT	BYTE	The Status output shows the error code if the instruction sets the Error output. Status is zero (no error) if the instruction sets the Busy or Done outputs.

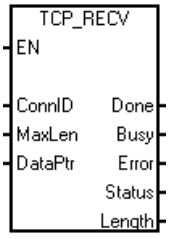
Example

This is an example usage of the TCP_SEND instruction:



12.6.3.5 TCP_RECV instruction

The TCP_RECV instruction retrieves data over an existing connection. You use this instruction for both TCP protocol and ISO-on-TCP protocol.

LAD/FBD	STL	Description
	**TCP_RECV ConnID, MaxLen, DataPtr, Done, Busy, Error, Status, Length**	The TCP_RECV retrieves data over an existing connection.

The TCP_RECV instruction only has an EN (Enable) input. The TCP_RECV instruction has no Req (Request) input. After the first execution of the TCP_RECV instruction, the status bits show the instruction is busy. Subsequent calls to TCP_RECV show a busy status until the CPU receives data on the specified connection.

After the CPU receives a message on the specified connection, the next execution of the TCP_RECV instruction performs the following tasks:

- Copies the message data to your program's data area (DataPtr)
- Sets the Length output to the number of bytes received
- Sets the Done output, clears the Busy and Error outputs, and sets the Status output byte value to zero (no error)

You should assign the receive area/buffer (DataPtr) and the maximum length of the receive buffer (MaxLen) so there is no possibility of a buffer overrun. If the CPU receives more bytes than can fit into your program's buffer (as specified by MaxLen), the TCP_RECV instruction copies MaxLen bytes to your program's data area and discards the rest of the received bytes. In this situation, the instruction sets the Error output and the Status output byte displays error code 25, which means the receive buffer is too small.

The maximum amount of data that you can receive in one message is 1024 bytes. The TCP_RECV instruction always operates in a mode that allows receipt of messages of varying lengths.

The TCP_RECV instruction operates differently depending on the protocol used. You selected the protocol for the connection when you called either TCP_CONNECT (TCP protocol) or ISO_CONNECT (ISO-on-TCP protocol) to create the connection.

Using TCP protocol, the TCP_RECV instruction returns all the bytes received by the S7-200 SMART CPU on the specified connection since the last time your program called the TCP_RECV instruction. Your program must call the TCP_RECV instruction often enough to delineate the messages properly because TCP acts as a "streaming" protocol. There is no delineation of the messages (no begin or end markers) in TCP protocol. Therefore, the CPU does not know when messages start or end.

For example, let us suppose that there is a TCP client that sends four 20-byte messages to the CPU in rapid succession, and your program does not call the TCP_RECV instruction during this time. When the program does call the TCP_RECV instruction after all four messages have been accepted by the CPU, the TCP_RECV instruction returns this as one receive message of 80 bytes. Your program is responsible for calling the TCP_RECV instruction often enough to receive each message as it is sent.

When you create a connection using the ISO-on-TCP protocol, the protocol itself delineates the messages. The TCP_RECV instruction receives and holds all messages sent from the remote device as separate messages in the CPU, no matter when or how often the program calls the TCP_RECV instruction.

For example, let us suppose this time there is an ISO-on-TCP client that sends four 20-byte messages to the CPU in rapid succession. Assume also that your program does not call the TCP_RECV instruction during this time. The ISO-on-TCP protocol delivers the four messages during four subsequent calls to the TCP_RECV instruction (one message per call). This

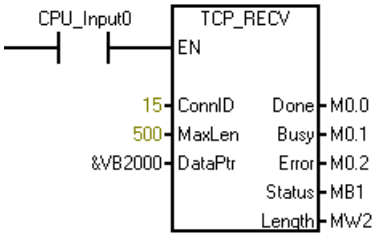
happens because ISO-on-TCP has start and end markers in the protocol to delineate the messages and separate them in the receiving device.

Table 12-22 Parameters of the TCP_RECV instruction

Parameter	Declaration	Data type	Description
EN	IN	BOOL	Enable input
ConnID	IN	WORD	The Connection ID (ConnID) is the number of the connection for this receive operation (defined during the connect process).
MaxLen	IN	WORD	The MaxLen is the maximum number of bytes to accept (for example, the size of the buffer at DataPtr (1 to 1024)).
DataPtr	IN	WORD	The DataPtr is the pointer to where the receive data should be stored. This is an S7-200 SMART pointer to I, Q, M, or V memory (for example, &VB100).
Done	OUT	BOOL	The instruction sets the Done output when the receive operation is complete with no errors. When the instruction sets the Done output, the Length output is valid.
Busy	OUT	BOOL	The instruction sets the Busy output while the receive operation is in progress.
Error	OUT	BOOL	The instruction sets the Error output when the receive operation is complete with an error. Refer to "Open user communication library instruction error codes" (Page 639) for further information.
Status	OUT	BYTE	The Status output shows the error code if the instruction sets the Error output. Status is zero (no error) if the instruction sets the Busy or Done outputs.
Length	OUT	WORD	The Length is the actual number of bytes received. Only when the instruction sets the Done or Error outputs is the Length valid. If the instruction sets the Done output, then the instruction received the entire message. If the instruction sets the Error output, the message was greater than the buffer size (MaxLen) and has been truncated.

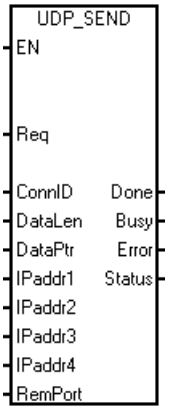
Example

This is an example usage of the TCP_RECV instruction:



12.6.3.6 UDP_SEND instruction

The UDP_SEND instruction transmits the requested number of bytes (DataLen) from the requested buffer location (DataPtr) to the device specified by the IP address (IPaddr1 – IPaddr4) and port (RemPort). This instruction is used only for UDP protocol and connections created with UDP_CONNECT.

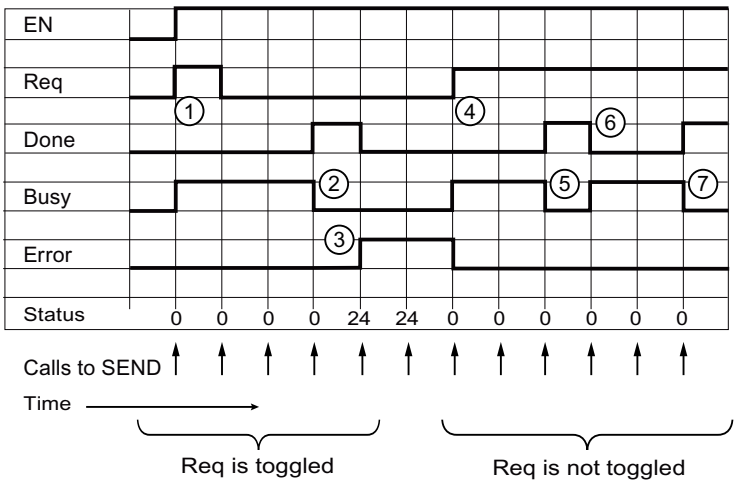
LAD/FBD	STL	Description
	**UDP_SEND Req, ConnID, DataLen, DataPtr, IPaddr1, IPaddr2, IPaddr3, IPaddr4, RemPort, Done, Busy, Error, Status**	The UDP_SEND instruction transmits the requested number of bytes from the requested buffer location to the device specified by the IP address and port.

The UDP_SEND instruction initiates sending the specified number of bytes when the following occur:

- The program calls the instruction with the Req input set to TRUE.
- The connection is not currently busy with another send operation.

The Req input is level-triggered. It is recommended that you put a positive edge trigger on the Req input so that the instruction does not initiate unintended send operations. The program ignores the Req input while the UDP_SEND is busy. The Done, Busy, and Error outputs and the Status output byte show the status of the UDP_SEND for each call.

The instruction displays the Done or Error status for one call of UDP_SEND after the send operation is complete. After that, the UDP_SEND responds with error code 24, which means no operation pending, if called with the Req input set FALSE. If the Req input is left set to TRUE, the program initiates another send operation. The figure below shows the relationship of the input and output parameters.



① Req is set TRUE so that the message send begins. Busy is set TRUE.

② The message send is complete. Done is set, and Busy is cleared.

③ EN is TRUE, and Req is FALSE, but no message send is in progress. So, Error is set with error code 24.

④ Req is set TRUE again, so another message send begins. Busy is set TRUE.

⑤ The message send is complete. Done is set, and Busy is cleared for one scan.

⑥ Req remains TRUE, so another message send begins.

⑦ The message send is complete.

The maximum amount of data that you can send in one send operation is 1024 bytes. The program copies the data from the send buffer in user memory to an internal buffer when the UDP_SEND executes with the Req input set to TRUE. You can change the program send buffer after the UDP_SEND executes and the instruction sets the Busy output.

The UDP_SEND instruction requires the IP address and port number on the remote device. When the UDP_CONNECT created the connection, the local port was set. The IP address (IPaddrx) and the remote port number (RemPort) are subject to the same rules and restrictions as described earlier. (Refer to "Parameters common to the OUC library instructions" (Page 617) for these rules.)

Note that UDP messages are not guaranteed to be delivered. There is no error returned if the remote device is not present.

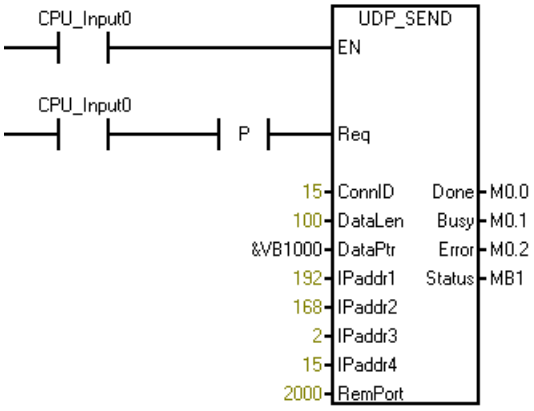
Table 12-23 Parameters of the UDP_SEND instruction

Parameter	Declaration	Data type	Description
EN	IN	BOOL	Enable input
Req	IN	BOOL	The CPU starts the send operation if Req = TRUE. If Req = FALSE, then the outputs show the current status of the send operation.
ConnID	IN	WORD	The Connection ID (ConnID) is the number of the connection for this send operation (defined during the connect process with UDP_CONNECT).
DataLen	IN	WORD	The DataLen is the number of bytes to transmit (1 to 1024).

Parameter	Declaration	Data type	Description
DataPtr	IN	DWORD	The DataPtr is the pointer to the data to be sent. This is an S7-200 SMART pointer to I, Q, M, or V memory (for example, &VB100).
IPaddr1 ... IPaddr4	IN	BYTE	These are the four IP address octets. IPaddr1 is the most significant byte and IPaddr4 is the least significant byte of the IP address.
RemPort	IN	WORD	The RemPort is the port number on the remote device. The remote port number range is 1 to 49151.
Done	OUT	BOOL	The instruction sets the Done output when the connect operation is complete with no errors.
Busy	OUT	BOOL	The instruction sets the Busy output while the connection operation is in progress.
Error	OUT	BOOL	The instruction sets the Error output when the connection operation is complete with an error. Refer to "Open user communication library instruction error codes" (Page 639) for further information.
Status	OUT	BYTE	The Status output shows the error code if the instruction sets the Error output. Status is zero (no error) if the instruction sets the Busy or Done outputs.

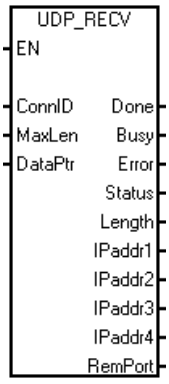
Example

This is an example usage of the UDP_SEND instruction:



12.6.3.7 UDP_RECV instruction

The UDP_RECV instruction retrieves data over an existing connection. This instruction is used only for UDP protocol on connections created with UDP_CONNECT.

LAD/FBD	STL	Description
	**UDP_RECV ConnID, MaxLen, DataPtr, Done, Busy, Error, Status, Length, IPaddr1, IPaddr2, IPaddr3, IPaddr4, RemPort**	The UDP_RECV retrieves data over an existing connection.

The UDP_RECV instruction only has an EN (Enable) input. The UDP_RECV instruction has no Req (Request) input. After the first execution of the UDP_RECV instruction, the status outputs show the instruction is busy. Subsequent calls to UDP_RECV show a busy status until the CPU receives data on the specified connection.

After the CPU receives a message on the specified connection, the next execution of the UDP_RECV instruction performs the following tasks:

- Copies the message data to your program's data area (DataPtr)
- Sets the returned Length to the number of bytes received
- Sets the IP address to the remote device that sent the message
- Set the remote port number (RemPort) to the remote device’s port
- Sets the Done output, clears the Busy and Error outputs, and sets the Status output byte value to zero (no error)

You should assign the receive area/buffer (DataPtr) and the maximum length of the receive buffer (MaxLen) so there is no possibility of a buffer overrun. If the CPU receives more bytes than can fit into your program's buffer (as specified by MaxLen), the UDP_RECV instruction copies MaxLen bytes to your program's data area and discards the rest of the received bytes. In this situation, the instruction sets the Error output and the Status output byte displays error code 25, which means the receive buffer is too small.

The maximum amount of data that you can receive in one message is 1024 bytes. The UDP_RECV instruction always operates in a mode that allows receipt of messages of varying lengths. Each message from a remote device is delineated as a separate message in the S7-200 SMART CPU. The UDP_RECV instruction only returns one received message for each call of the instruction.

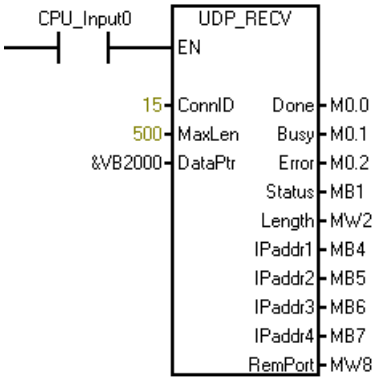
The UDP_RECV instruction also returns the IP address and port number of the remote device. This allows your program to respond to the remote device with a UDP_SEND where the remote IP address and port are required parameters.

Table 12-24 Parameters of the UDP_RECV instruction

Parameter	Declaration	Data type	Description
EN	IN	BOOL	Enable input
ConnID	IN	WORD	The CPU uses the Connection ID (ConnID) number for this receive operation (defined during the connect process).
MaxLen	IN	WORD	The MaxLen is the maximum number of bytes to accept (for example, the size of the buffer at DataPt (1 to 1024)).
DataPtr	IN	DWORD	The DataPtr is the pointer to where the receive data should be stored. This is an S7-200 SMART pointer to I, Q, M, or V memory (for example, &VB100).
Done	OUT	BOOL	The instruction sets the Done output when the receive operation is complete with no errors. When the instruction sets the Done output, the Length output is valid.
Busy	OUT	BOOL	The instruction sets the Busy output while the receive operation is in progress.
Error	OUT	BOOL	The instruction sets the Error output when the receive operation is complete with an error. Refer to "Open user communication library instruction error codes" (Page 639) for further information.
Status	OUT	BYTE	The Status output shows the error code if the instruction sets the Error output. Status is zero (no error) if the instruction sets the Busy or Done outputs.
Length	OUT	WORD	The Length is the actual number of bytes received. Only when the instruction sets the Done or Error outputs is the Length valid. If the instruction sets the Done output, then the instruction received the entire message. If the instruction sets the Error output, the message was greater than the buffer size (MaxLen) and has been truncated.
IPaddr1 ... IPaddr4	OUT	BYTE	These are the four IP address octets of the remote device that sent the message. IPaddr1 is the most significant byte and IPaddr4 is the least significant byte of the IP address.
RemPort	OUT	WORD	The RemPort is the port number of the remote device that sent the message.

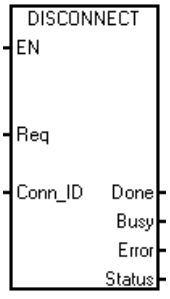
Example

This is an example usage of the UDP_RECV instruction:



12.6.3.8 DISCONNECT instruction

The DISCONNECT instruction terminates an existing communication connection. The DISCONNECT instruction is used for all protocols.

LAD/FBD	STL	Description
	**DISCONNECT ConnID, LocPort, Done, Busy, Error, Status**	The DISCONNECT terminates an existing communication connection for all protocols.

The DISCONNECT instruction initiates the connection termination when the program calls the DISCONNECT instruction with the Req input set TRUE. The Req input is level-triggered. It is recommended that you put a positive edge trigger on the Req input.

If the requested connection (ConnID) is currently busy connecting, disconnecting, or cannot be found because the connection has been reused, the DISCONNECT instruction returns an error.

The DISCONNECT instruction displays the Done or Error output status for at least one call of the instruction after the disconnect operation is complete. The instruction may return the status of the DISCONNECT instruction for the specified connection for subsequent calls until the CPU reuses the connection for another connect operation.

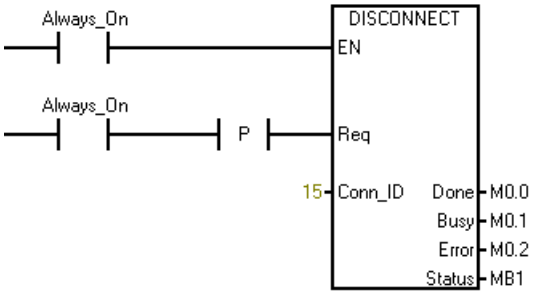
After the DISCONNECT instruction completes a disconnect, if the program calls the DISCONNECT instruction with Req set FALSE, the instruction returns error code 24, which means no operation pending.

Table 12-25 Parameters of the DISCONNECT instruction

Parameter	Declaration	Data type	Description
EN	IN	BOOL	Enable input
Req	IN	BOOL	The CPU starts the disconnect operation if Req = TRUE.
ConnID	IN	WORD	The CPU uses the Connection ID (ConnID) number to identify the connection to be terminated (defined during the connect process).
Done	OUT	BOOL	The instruction sets the Done output when the disconnect operation is complete with no errors.
Busy	OUT	BOOL	The instruction sets the Busy output while the disconnect operation is in progress.
Error	OUT	BOOL	The instruction sets the Error output when the disconnect operation is complete with an error. Refer to "Open user communication library instruction error codes" (Page 639) for further information.
Status	OUT	BYTE	The Status output shows the error code if the instruction sets the Error output. Status is zero (no error) if the instruction sets the Busy or Done outputs.

Exampled

This is an example usage of the DISCONNECT instruction:



12.6.4 Open user communication library instruction error codes

The following table lists the Open User Communication (OUC) library (Page 616) instruction error codes:

Error code	Description	CONNECT	SEND	RECV	DISCONNECT
0	No error	X	X	X	X
1	The data length input parameter is greater than the maximum allowed (1024 bytes).		X	X	
2	The data buffer is not in I, Q, M, or V memory areas.		X	X	
3	The data buffer does not fit in the memory area.		X	X	
5	The connection is locked in another context. You are attempting to access the same connection in both the background (the Main) and in an interrupt program organizational unit (POU) at the same time.	X	X	X	X
6	A UDP IP address or port error		X		
7	An instance mismatch: The connection is busy with another instance or the input data does not match the data stored for the requested connection ID when the request was initiated.	X	X	X	X
8	The Connection ID does not exist because the connection has never been created, or the connection was terminated by your program using the DISCONNECT instruction.	X	X	X	X
9	A TCP_CONNECT, ISO_CONNECT, or UDP_CONNECT instruction is in progress with this Connection ID.		X	X	X
10	A DISCONNECT instruction is in progress with this Connection ID.	X	X	X	
11	A TCP_SEND or UDP_SEND instruction is in progress with this Connection ID.		X		X
12	A temporary communication error has occurred. The connection cannot be started at this time. Try again.	X	X	X	
13	The connection partner refused or actively dropped the connection. The partner issued a disconnect to this CPU.	X	X	X	
14	The CPU cannot reach the connection partner. There was no answer to the connect request.	X	X	X	
15	The CPU aborted the connection due to inconsistencies. Disconnect and reconnect to correct the situation.	X	X	X	X
16	The Connection ID is already in use with a different IP address, port, or TSAP combination.	X			
17	No connection resource is available. All connections of the requested type (active/passive) are in use.	X			

Error code	Description	CONNECT	SEND	RECV	DISCONNECT
18	The local or remote port number is reserved or the port number is already in use for another server (passive) connection.	X			
19	One of the following IP address errors have occurred: • The IP address is invalid (for example, address 0.0.0.0). • This IP address is the IP address of this CPU. • This CPU has IP address 0.0.0.0. • The IP address is a broadcast or multicast address.	X			
20	A local or remote TSAP error (ISO-on-TCP only)	X			
21	An invalid connection ID (65535 is reserved)	X			
24	There is no operation pending so there is no status to report.		X		X
25	The receive buffer is too small: The CPU received more bytes than the buffer length supports. The CPU discards the extra bytes.			X	
31	Unknown error. Disconnect and reconnect to fix the problem.	X	X	X	X

12.6.5 Open user communication library example

12.6.5.1 Active partner (client)

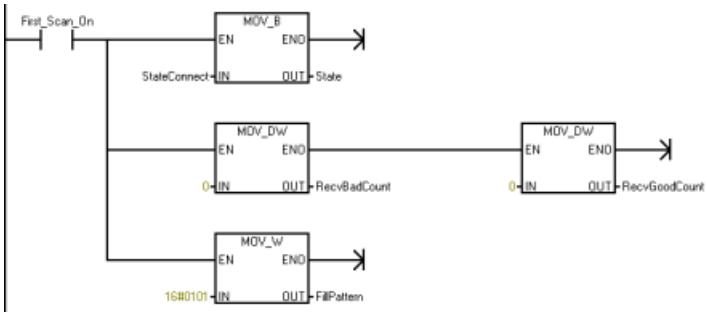
This program implements a simple state machine to manage the creating of a connection, cyclically sending/receiving a message, and handling errors.

The flow of this state machine is to connect, then repeatedly send and receive a message. If the connection is dropped, the state machine tries to reconnect.

Refer to the "Active partner symbol table" (Page 650) to see the symbol table for this program.

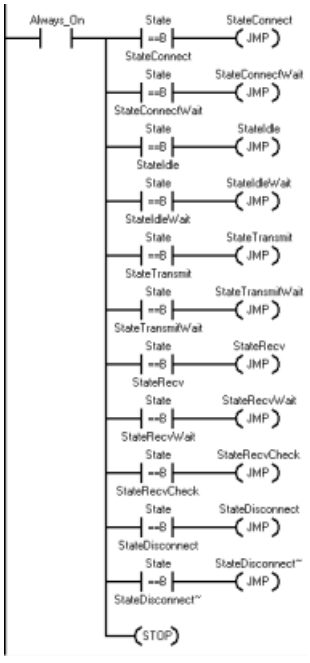
Network 1: On the first scan....

Initialize the State variable to initiate a connection. Clear the good and bad receive counts, and initialize some data to send.



Network 2: Process the state machine...

Determine the current state of the state machine and jump to the label for the state handler. If the state is ever illegal, the CPU goes to the STOP mode.

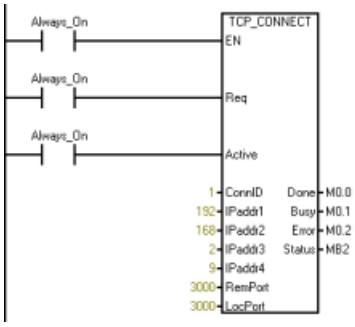


Network 3: State "Connect"...

Start the connection process.

Network 4: Create an active connection to the passive device.

The program only calls the TCP_CONNECT instruction one time in this state. Set the Req input to TRUE to start the connection process. Since this is the active side of the connection, the instruction sets the Active input to TRUE.

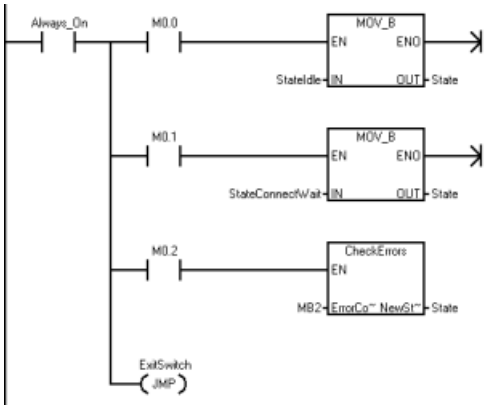


Network 5: If Done is TRUE, the CPU established the connection to go to the "Idle" state.

If Busy is TRUE, go to the "Connect Wait" state to wait for the connection to be established.

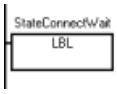
If Error is TRUE, there is probably an error with the input parameters, so check to see what state to go to next.

In all cases, exit the state machine for this scan. The program continues to the next state on the next scan.

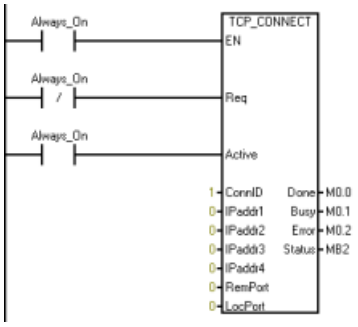


Network 6: State "Connect Wait"...

Wait in this state until there is a connection created to the passive device.



Network 7: Call the TCP_CONNECT instruction with Req = FALSE and the same Connection ID (ConnID) as above. Do this to check the connection status and ensure that the CPU established the connection.

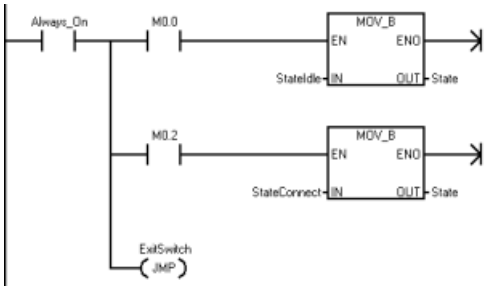


Network 8: If Done is TRUE, this means that the CPU established the connection, so continue in the "Idle" state.

If Busy is TRUE, stay in the "Connect Wait" state. The TCP_CONNECT instruction eventually times out and returns an error if the other device is not present, so you do not need to have a time out mechanism for the connect process.

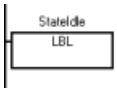
If Error is TRUE, there is a problem connecting to the passive device. In this case, just try again by going back to the "Connect" state. Note that if the passive device is present but it rejects the connection request, the connection errors very quickly and utilizes a great amount of bandwidth as the CPU continues to attempt to create a connection.

In all cases, exit the state machine for this scan. The program continues to the next state on the next scan.

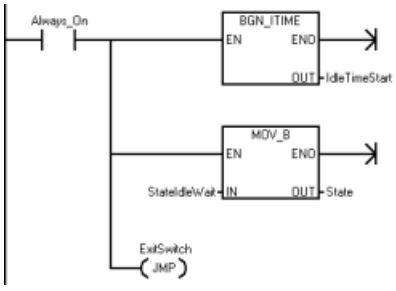


Network 9: State "Idle"...

This state puts a time delay between messages so that you do not flood the network. The symbol "IdleTimeDelay" specifies the delay time.

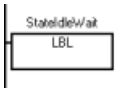


Network 10: Capture the interval timer and then change to the "Idle Wait" state to delay until it is time to transmit again.

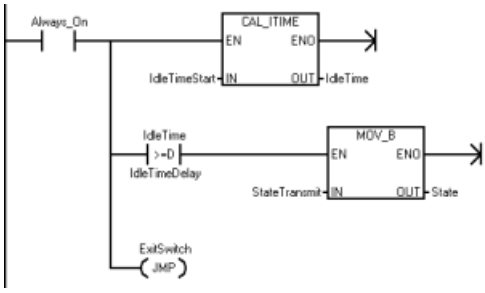


Network 11: State "Idle Wait"...

Stay in this state for the number of milliseconds specified (IdleTimeDelay).



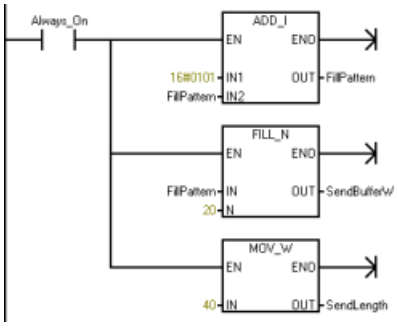
Network 12: Calculate the time since you entered the "Idle" state and, if this is greater than the "IdleTimeDelay" value, change the state to the "Transmit" state.



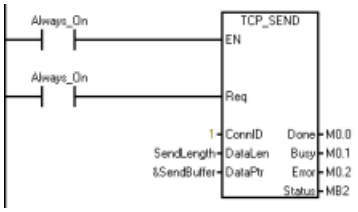
Network 13: State 'Transmit"...

Network 14: Create the message to send to the passive device.

For the test program, fill the send buffer with 40 bytes (20 words). Write the number of bytes to send into a variable so that you can use the same value in the "TransmitWait" state.



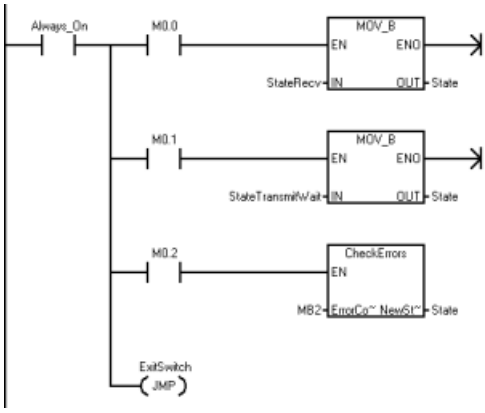
Network 15: Send the new message to the passive device. Set Req to TRUE to initiate the new send operation.



Network 16: If Done is TRUE, the send is complete (probably will never happen this quickly), so go to the "Receive" state on the next scan.

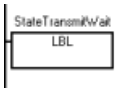
If Busy is TRUE (this is the normal situation), go to the "Transmit Wait" state to wait for the transmit to finish.

If Error is TRUE, check the reason, and you may need to change state if there is a connection issue.

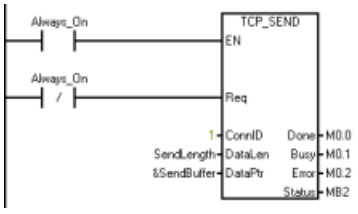


Network 17: State "Transmit Wait"...

Wait in this state until the transmit is complete.



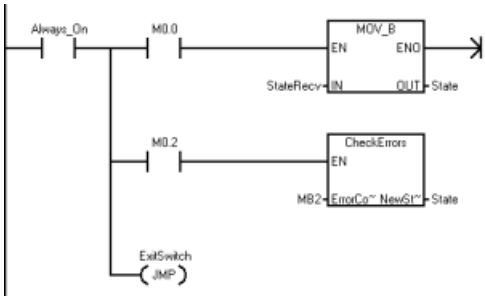
Network 18: Call the TCP_SEND instruction with Req = FALSE to determine if the send is complete. Be sure to use the same send length and buffer pointer that you used to initialize the send request.



Network 19: If Done is TRUE, the send is complete, so go to the "Receive" state.

If Busy is TRUE, stay in the "Transmit Wait" state.

If Error is TRUE, check the reason for the error and change the state if there is a connection issue.

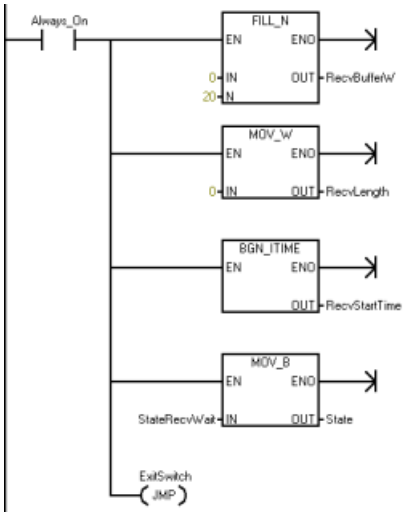


Network 20: State "Receive"...

Clean up the receive data area and prepare to receive the response.

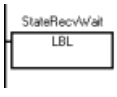
Network 21: Clear the receive buffer and the "RecvLength" so that there is no data left from the last received message.

Capture the current interval time value (in "RecvStartTime") to support a receive timeout and then go to the "Receive Wait" state.

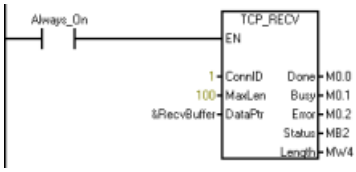


Network 22: State "Receive Wait"...

Stay in this state until you either receive some data or time out.



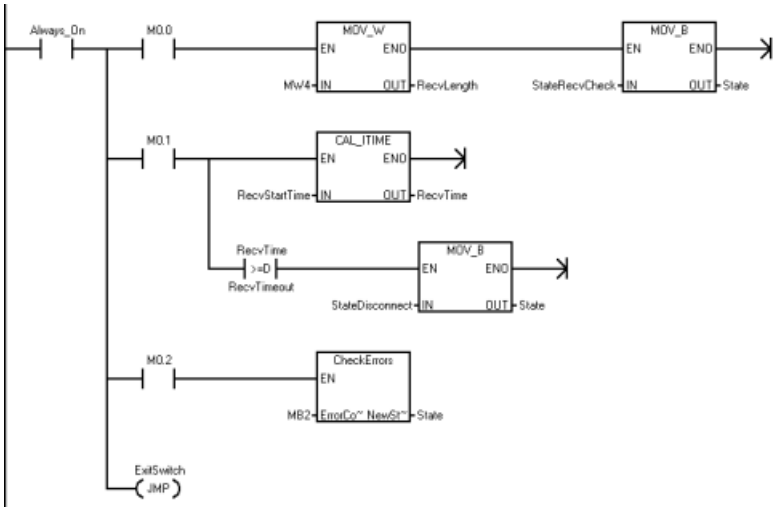
Network 23: Call the TCP_RECV instruction to get any messages that the CPU has received.



Network 24: If Done is TRUE, the instruction received new data, so go to the "Receive Check" state.

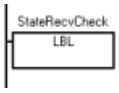
If Busy is TRUE, stay in the "Receive" state until you reach the receive timeout value. If you time out in this state, disconnect the device and then reconnect it.

If Error is TRUE, check the error code to determine what to do next.

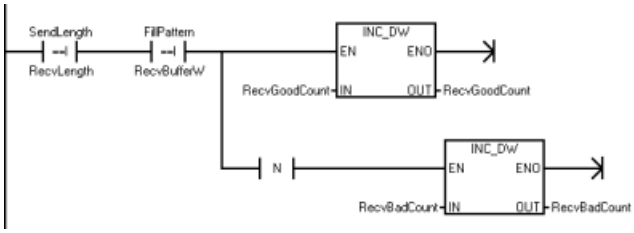


Network 25: State "RecvCheck"...

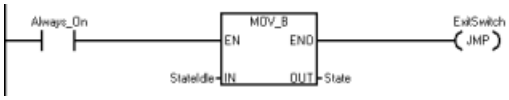
Process the data returned from the passive device.



Network 26: This program only checks the returned data to ensure that you received the same amount of data as was sent and that the first word matches the "Fill Pattern". Log whether the response was good or bad, and then go to the "Idle" state to wait to send the next message.



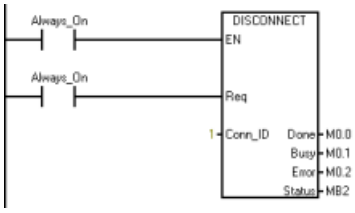
Network 27: Go to the "Idle" state in all situations.



Network 28: State "Disconnect"...

Initiate a disconnect.

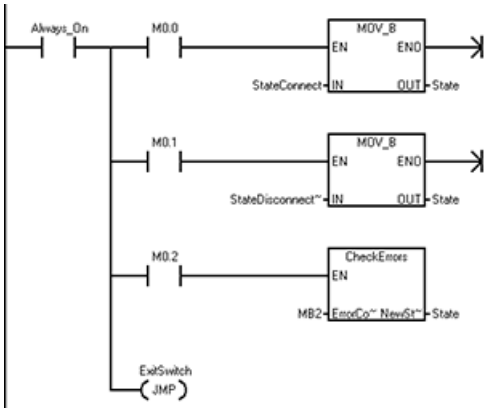
Network 29: Initiate the disconnect with the ConnID by calling the DISCONNECT instruction with Req = TRUE.



Network 30: If Done is TRUE, this means that the disconnect is complete (probably will never happen), so go try to reconnect.

If Busy is TRUE (this would be normal), go to the "Disconnect Wait" state.

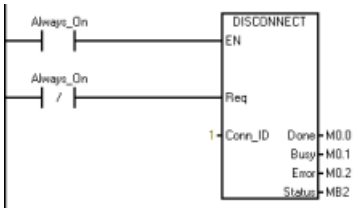
If Error is TRUE, check the reason, and you may need to change state if there is a connection issue.



Network 31: State "Disconnect Wait"...

Wait in this state for the disconnect operation to complete.

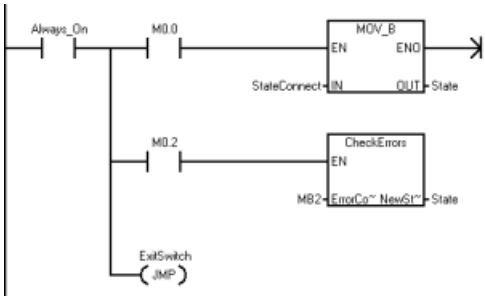
Network 32: Call the DISCONNECT instruction with Req = FALSE to check the status of the disconnect operation.



Network 33: If Done is TRUE, this means that the disconnect is complete, so try to reconnect on the next scan.

If Busy is TRUE (this would be normal), stay in the "Disconnect Wait" state.

If there is an error, check the reason, and you may need to change state based upon the error code.



Network 34: Exit Switch.

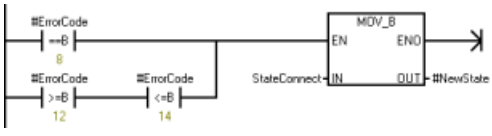
12.6.5.2 CheckErrors subroutine

The CheckErrors subroutine checks the Open User Communication error codes and determines if the program requires a state change. You use the same CheckErrors subroutine for both the Active partner (client) and Passive partner (server).

Network 1: The program malfunctioned if there is no error code. Disconnect and then reconnect to correct this problem.

Network 2: If one of the partners drops the connection, the program displays error codes 8, 12, 13, and 14. The partners are currently disconnected.

In all these cases, reconnect to the partner. Set the state to "Connect".

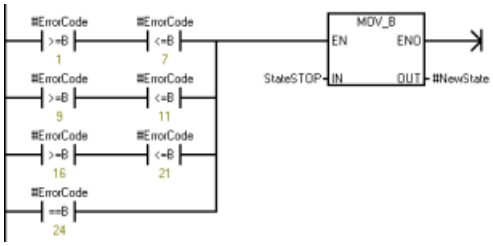


Network 3: If the error is a parameter error (error codes 1 - 7), stop the program because there is a configuration error with one of inputs to the function. Correct this in the program.

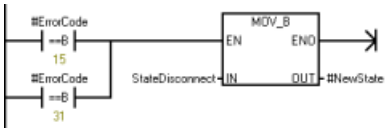
If the error code is 9 (connect in progress), 10 (disconnect in progress), or 11 (send in progress), stop because the state machine is broken. The state machine is programmed to stay in the associated waiting state until the operation is complete, and these errors should never occur.

If the error code is between 16 and 21, these constitute connect parameter errors and should not appear here. If these errors occur, there is a malfunction, so stop program execution.

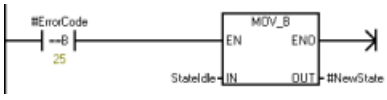
If the SEND and DISCONNECT instructions return error 24, there is no operation currently pending. This probably means that the operation is complete; however, error 24 should never appear based upon the state machine. Consider this an error, and go to stop.



Network 4: The program returns errors 15 and 31 if there are problems with the connection. Disconnect and then reconnect to correct this problem.



Network 5: The program returns error 25 when the RECV function receives more data than the buffer can support. In our case, continue as if nothing happened.



12.6.5.3 Active partner variable table

The table below lists the variable names, addresses, and comments for the Active partner (client) program.

Variable	Address	Comments
Always_On	SM0.0	Always ON
First_Scan_On	SM0.1	On for the first scan cycle only
State	VB0	
IdleTime	VD4	
IdleTimeStart	VD8	
RecvGoodCount	VD20	
RecvBadCount	VD24	

Variable	Address	Comments
RecvStartTime	VD28	
RecvTime	VD32	
FillPattern	VW12	
RecvLength	VW16	
SendLength	VW18	
SendBufferW	VW1000	
RecvBufferW	VW2000	
StateConnect	1	
StateConnectWait	2	
StateIdle	3	
StateIdleWait	4	
StateTransmit	5	
StateTransmitWait	6	
StateRecv	7	
StateRecvWait	8	
StateRecvCheck	9	
StateDisconnect	10	
StateDisconnectWait	11	
ExitSwitch	19	
RecvTimeout	100	Milliseconds for receive timeout
IdleTimeDelay	250	Milliseconds between send commands

12.6.5.4 Passive partner (server)

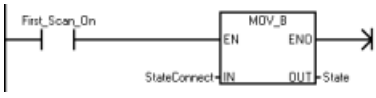
This program implements a simple state machine to manage the opening of a connection, receiving a message, sending a response, and handling errors.

The flow of this state machine is to connect, then repeatedly receive a message and send a response. If the connection is dropped, the state machine reopens the passive connection.

Refer to the "Passive partner symbol table" (Page 658) to see the symbol table for this program.

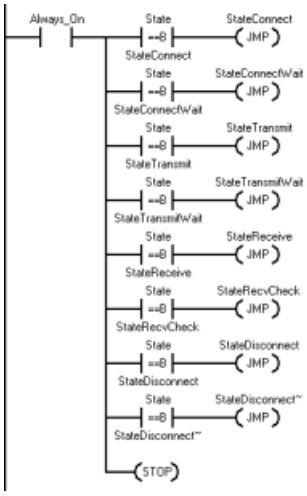
Network 1: On the first scan....

Initialize the State variable to initiate a connection.



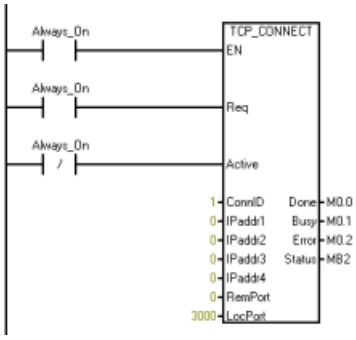
Network 2: Process the state machine...

Determine the current state of the state machine and jump to the label for the state handler. If the state is ever illegal, the CPU goes to the STOP mode.



Network 3: State Connect...

Network 4: Start the connection process. This is a server so set the Active input FALSE. Set the IPaddr inputs to zero so that the server accepts a connection from any address. Set the RemPort to zero because you do not need it for a server connection. Call the TCP_CONNECT instruction with the Req input TRUE to start the connection process.

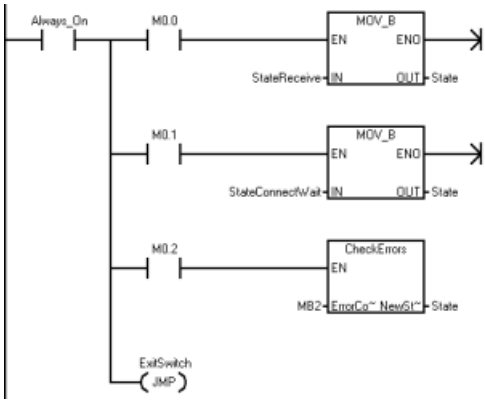


Network 5: If Done is TRUE, the CPU established the connection so go to the "Idle" state.

If Busy is TRUE, go to the "Connect Wait" state to wait for the connection to be established.

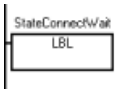
If Error is TRUE, there is probably an error with the input parameters, so check to see what state to go to next.

In all cases, exit the state machine for this scan. The program continues to the next state on the next scan.

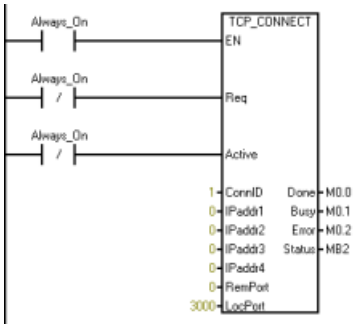


Network 6: State "Connect Wait"...

Wait in this state until the active partner creates a connection to this CPU.



Network 7: Call the TCP_CONNECT instruction with Req = FALSE and the same Connection ID (ConnID) as above. Do this to check the connection status.

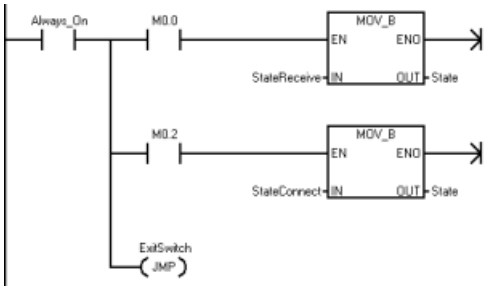


Network 8: If Done is TRUE, this means that the CPU established the connection so continue in the "Idle" state.

If Busy is TRUE, stay in the "Connect Wait" state. Since this is a passive connection, stay in the "Busy" state until the active partner connects to the CPU. There is no timeout for a passive connection.

If Error is TRUE, a problem occurred so go back and try again to connect on the next scan.

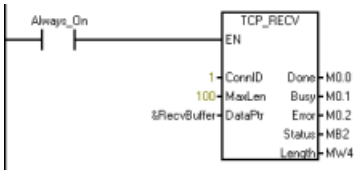
In all cases, exit the state machine for this scan. The program continues to the next state on the next scan.



Network 9: State Receive...

Stay in this state until the server receives some data.

Network 10: Call the TCP_RECV instruction to get any messages that the CPU has received.

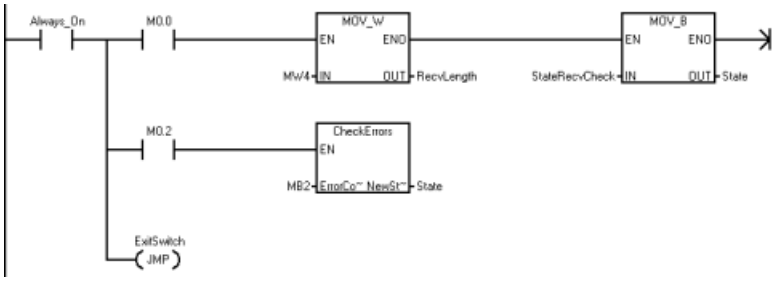


Network 11: If Done is TRUE, the CPU received new data so go to the "Receive Check" state.

If Busy is TRUE, stay in the "Receive" state until you receive some data.

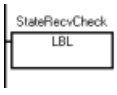
If Error is TRUE, check the error code to determine what to do next.

In all cases, exit the state machine for this scan. The program continues to the next state on the next scan.



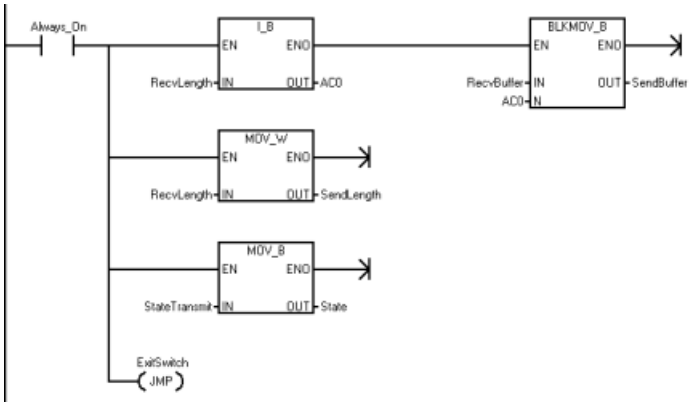
Network 12: State "Receive Check"...

Check and process the received data.



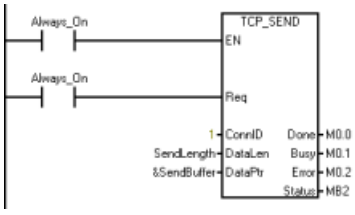
Network 13: In this program, you echo the data back to the partner. Copy all of the received bytes to the send buffer.

Change the state to "Transmit", and exit until the next scan.



Network 14: State "Transmit"...

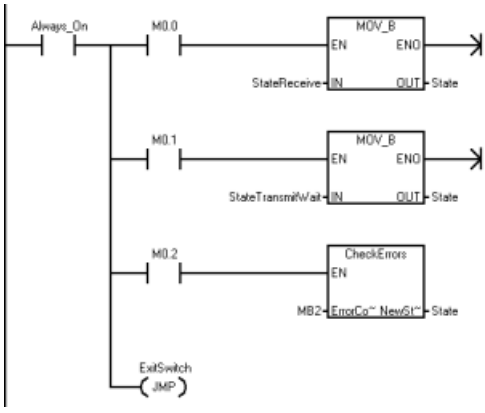
Network 15: Send the data back to the partner.



Network 16: If Done is TRUE, the send is complete (probably never happen this quickly), so go to the "Receive" state on the next scan.

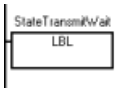
If Busy is TRUE (this is the normal situation), go to the "Transmit Wait" state to wait for the transmit to finish.

If Error is TRUE, check the reason, and you may need to change state if there is a connection issue.

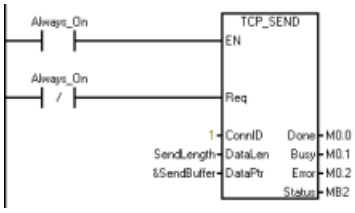


Network 17: State "Transmit Wait"...

Wait in this state until the transmit is complete.



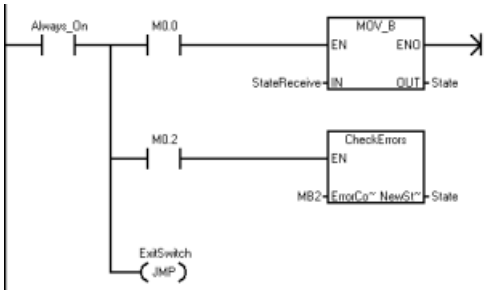
Network 18: Call the TCP_SEND instruction with Req = FALSE to determine when the send is complete. Be sure to use the same send length and buffer pointer that you used to initialize the send request.



Network 19: If Done is TRUE, the send is complete so go to the "Receive" state.

If Busy is TRUE, stay in the "Transmit Wait" state.

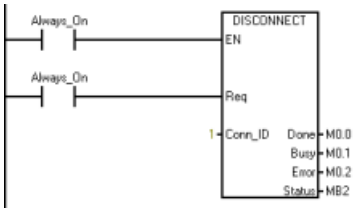
If Error is TRUE, check the reason for the error and change the state if there is a connection issue.



Network 20: State "Disconnect"...

Initiate a disconnect.

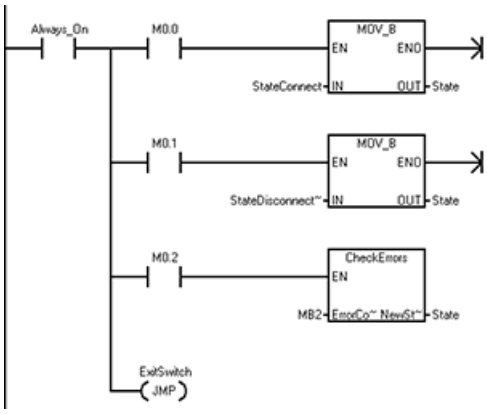
Network 21: Initiate the disconnect with the ConnID by calling the DISCONNECT instruction with Req = TRUE.



Network 22: If Done is TRUE, this means that the disconnect is complete (probably will never happen), so try to reconnect.

If Busy is TRUE (this would be normal), go to the "Disconnect Wait" state.

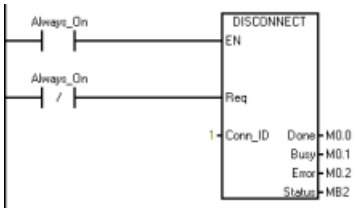
If Error is TRUE, check the reason, and you may need to change state if there is a connection issue.



Network 23: State "Disconnect Wait"...

Wait in this state for the disconnect operation to complete.

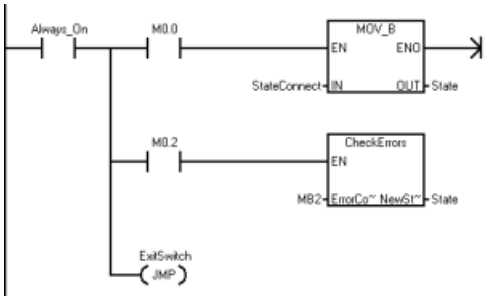
Network 24: Call the DISCONNECT instruction with Req = FALSE to check the status of the disconnect operation.



Network 25: If Done is TRUE, this means that the disconnect is complete, so try to reconnect on the next scan.

If Busy is TRUE (this would be normal), stay in the "Disconnect Wait" state.

If there is an error, check the reason, and you may need to change state based upon the error code.



Network 26: Exit Switch.

12.6.5.5 CheckErrors subroutine

The CheckErrors subroutine (Page 649) checks the Open User Communication error codes and determines if the program requires a state change. You use the same CheckErrors subroutine for both the Active partner (client) and Passive partner (server).

12.6.5.6 Passive partner variable table

The table below lists the Variable names, addresses, and comments for the Passive partner (server) program.

Variable	Address	Comments
Always_On	SM0.0	Always ON
First_Scan_On	SM0.1	On for the first scan cycle only
State	VB0	
SendBuffer	VB1000	
RecvBuffer	VB2000	
RecvLength	VW16	
SendLength,	VW18	
SendBufferW	VW1000	
RecvBufferW	VW2000	
StateConnect	1	
StateConnectWait	2	
StateTransmit	3	
StateTransmitWait	4	
StateReceive	5	
StateRecvCheck	6	
StateDisconnect	7	
StateDisconnectWait	8	
ExitSwitch	10	

12.7 PN Read Write Record library

12.7.1 PN Read Write Record features

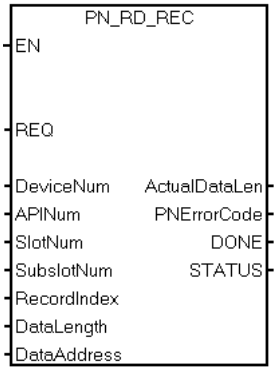
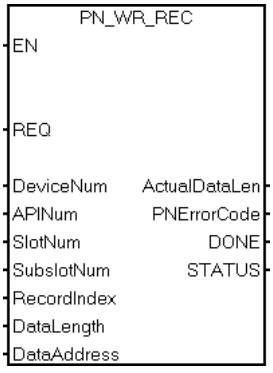
PN Read Write Record library includes the following two instructions:

- PN_RD_REC: Read a data record from any connected PROFINET device
- PN_WR_REC: Write a data record to any connected PROFINET device.

12.7.2 Input and output interface of PN Read Write Record library

The PN_RD_REC and PN_WR_REC instructions are as follows:

Table 12-26 PN_RD_REC and PN_WR_REC

LAD/FBD	STL	Description
	CALL PN_RD_REC, REQ,DeviceNum, APINum, SlotNum, SubslotNum, RecordIndex, DataLength, DataAddress, ActualDataLen, PnErrorCode, Done, STATUS	Use the PN_RD_REC instruction to read a data record from PROFINET device.
	CALL PN_WR_REC, REQ,DeviceNum, APINum, SlotNum, SubslotNum, RecordIndex BufferLength, DataAddress, ActualDataLen, PnErrorCode, Done, STATUS	Use the PN_WR_REC instruction to write a data record to PROFINET device.

The parameters of the PN_RD_REC and PN_WR_REC instructions are as follows:

Table 12-27 Parameters of PN_RD_REC and PN_WR_REC instructions

Parameter and type		Data type	Description
REQ	IN	BOOL	REQ=1: Transfer data record
Device Number	IN	WORD **Note:** The value range is from 1 to 8.	Device Number, API Number, Slot Number and SubSlot Number are used to address a submodule. You can find the Device Number, API Number, Slot Number and SubSlot Number in the PROFINET wizard.
API Number	IN	DWORD	
Slot Number	IN	WORD	
SubSlot Number	IN	WORD	
Record Index	Input	WORD	The Record Index includes the record index from protocol or the user-defined record index. For detailed information of the index from the protocol, refer to Technical Specification for PROFINET IO (Version 2.3).
Data Length	Input	WORD	This parameter refers to the number of bytes of the buffer. The buffer stores the data record read from or written to the device. The value range: from 1 to 1024.
Data Address	Input	DWORD	Address of the buffer read from or written to the device. **Note**: If the buffer length is greater than the actual record data length, the buffer contains all the record data.If the buffer length is smaller than actual record data length, the buffer contains partial record data and an error occurs.
Actual record data length	Output	WORD	This parameter is valid for the instruction RDREC and returns the actual data length specified by the device.
PROFINET Error Code	Output	DWORD	The error code defined by the PROFINET protocol. 0 = no error. If the value is not 0, check the specific error code in Technical Specification for PROFINET IO (Version 2.3).
DONE	OUT	BOOL	The instruction is completed.
STATUS	OUT	BYTE	The status of the current operation. For detailed information, refer to Definition of parameters for input signal "STATUS" (Page 660).

12.7.3 Definition of parameters for input signal "STATUS"

The following table lists the parameter information of "STATUS":

Table 12-28 STATUS

Byte Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	A [1]	E [2]	Error code [3]					

[1] A : 1 = a request is in process

[2] E : 1= an error occurs

[3] Error code: The system error code. For detailed information, refer to System_defined error code of the library PN Read Write Record (Page 661).

12.7.4 System_defined error code of the library PN Read Write Record

The error codes are as follows:

Error code	Description
0	No error.
1	The data length parameter is 0 or is greater than the supported maximum length (1024 bytes).
2	The data buffer is not in I, Q, M, or V memory areas.
3	The data buffer does not fit in the memory area.
4	The table doesn't match with the memory.
5	The device number is invalid and not within the range: from 1 to 8.
6	An instance mismatch: The connection is busy with another instance, whose device number, API number, slot number and subslot number are same as the requested instance, but with a different buffer size and data address.
7	The PROFINET device is not connected.
8	The size of the received buffer exceeds 1024 bytes.
9	Call sequence is invalid.
10	Parameters are invalid (for example, out-of-range).
11	The AR is created afresh in the meanwhile.
12	The RPC reports a timeout error.
13	The RPC reports a communication error.
14	The RPC Server of the IOD signaled “busy” (for example, the call can be repeated later).
15	CLRPC reports an error or the PDU cannot be parsed.
16	CM response is OK, but has a PROFINET protocol defined error.
17	The instruction parameter is invalid.
24	REQ is not enabled.
25	The buffer length is smaller than the actual data record length.
63	Unknown error.

12.8 USS Protocol library instructions

12.8.1 USS communication overview

12.8.1.1 USS protocol overview

STEP 7-Micro/WIN SMART V3 instruction libraries make controlling Siemens drives easier by including pre-configured subroutines and interrupt routines that are specifically designed for using the USS protocol to communicate with a motor drive. You can control the physical drive and the read/write drive parameters with the USS instructions.

Siemens designed the USS communications library for use with Siemens general purpose drives such as the Siemens Micromaster series. Siemens does not intend for the USS communications library to be used with special purpose drives such as the V90 servo drive. The control interface of the V90 servo drive is different from that of a general purpose drive. For this reason, do not use the USS communications library with the V90 servo drive.

You find these instructions in the "Libraries" folder of the STEP 7-Micro/WIN SMART V3 instruction tree. When you select a USS instruction, one or more associated subroutines and interrupts are added automatically.

The USS protocol library overview discusses the following subjects:

- Requirements for using the USS protocol (Page 663)
- Calculating the time required for communicating with the drive (Page 663)

Refer to "Using the USS protocol instructions (Page 664)" for a listing of USS protocol instructions, error codes, and example programs.

Note

Only call the library functions from either the main program or from interrupt routines, but not both.

12.8.1.2 Requirements for using the USS protocol

The STEP 7-Micro/WIN SMART instruction libraries provide subroutines, interrupt routines, and instructions to support the USS protocol. The USS instructions use the following resources in the S7-200 SMART CPU:

- The USS protocol is an interrupt driven application. In the worst case, the receive message interrupt routine requires up to 2.5 ms to execute. During this time, all other interrupt events are queued for service after the receive message interrupt routine has been executed. If your application cannot tolerate this worst case delay, then you may want to consider other solutions for controlling drives.
- Initializing the USS protocol dedicates an S7-200 SMART CPU port for USS communications. You use the USS_INIT instruction to select either USS or PPI for port 0, port 1 or port 2. (USS refers to the USS protocol for Siemens drives.) When a port is set to use the USS protocol for communicating with drives, you cannot use the port for any other purpose, including communicating with an HMI. The second communications port allows STEP 7-Micro/WIN SMART V3 to monitor the control program while USS protocol is running.
- The USS instructions affect all of the SM locations that are associated with Freeport communication on the assigned port.
- The USS subroutines and interrupt routines are stored in your program. The USS instructions increase the amount of memory required for your program by up to 3050 bytes. Depending on the specific USS instructions used, the support routines for these instructions can increase the overhead for the control program by at least 2150 bytes and up to 3050 bytes.
- The variables for the USS instructions require a 402-byte block of V memory. The starting address for this block is assigned by the user and is reserved for USS variables.
- Some of the USS instructions also require a 16-byte communications buffer. As a parameter for the instruction, you provide a starting address in V memory for this buffer. It is recommended that a unique buffer be assigned for each instance of USS instructions.
- When performing calculations, the USS instructions use accumulators AC0 to AC3. You can also use the accumulators in your program; however, the values in the accumulators will be changed by the USS instructions.
- The USS instructions cannot be used in an interrupt routine.

12.8.1.3 Calculating the time required for communicating with the drive

Communications with the drive are asynchronous to the S7-200 SMART CPU scan. The CPU typically completes several scans before one drive communications transaction is completed. The following factors help determine the amount of time required:

- Number of drives present
- Baud rate
- Scan time of the CPU

Some drives require longer delays when using the parameter access instructions. The amount of time required for a parameter access is dependent on the drive type and the parameter being accessed.

After a USS_INIT instruction assigns Port 0 to use the USS Protocol, the CPU regularly polls all active drives at the intervals shown in the following table. You must set the time-out parameter of each drive to allow for this task:

Table 12-29 Communications times

Baud rate	Time between polls of active drives (with no parameter access instructions active)
1200	240 ms (maximum) times the number of drives
2400	130 ms (maximum) times the number of drives
4800	75 ms (maximum) times the number of drives
9600	50 ms (maximum) times the number of drives
19200	35 ms (maximum) times the number of drives
38400	30 ms (maximum) times the number of drives
57600	25 ms (maximum) times the number of drives
115200	25 ms (maximum) times the number of drives

12.8.2 USS program instructions

12.8.2.1 Using the USS protocol instructions

Procedure

To use the USS protocol instructions in your S7-200 SMART program, follow these steps:

1. Insert the USS_INIT instruction in your program and execute the USS_INIT instruction for one scan only. You can use the USS_INIT instruction either to initiate or to change the USS protocol communication parameters.
   When you insert the USS_INIT instruction, several hidden subroutines and interrupt routines are automatically added to your program.
2. Place only one USS_CTRL instruction in your program for each active drive.
   You can add as many USS_RPM_x and USS_WPM_x instructions as required, but only one of these can be active at a time.
3. Click the Memory button Memory from the Libraries area of the File menu ribbon strip to assign a starting address for the V Memory that the USS library requires. Alternatively, you can right-click the Program Block node in the project tree and select "Library Memory" from the context menu.

4. Configure the drive parameters to match the baud rate and address used in the program.
5. Connect the communications cable between the S7-200 SMART CPU and the drives. Ensure that all of the control equipment, such as the S7-200 SMART CPU, that is connected to the drive be connected by a short, thick cable to the same ground or star point as the drive.

CAUTION

Avoiding unwanted current flow

Interconnecting equipment with different reference potentials can cause unwanted currents to flow through the interconnecting cable. These unwanted currents can cause communications errors or damage equipment.

Ensure that all equipment that is connected with a communications cable either shares a common circuit reference or is isolated to prevent unwanted current flows.

The shield must be tied to chassis ground or pin 1 on the 9-pin connector. It is recommended that you tie terminal 2-0V on the drive to chassis ground.

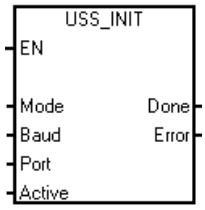
The USS protocol instructions consist of the following:

- USS_INIT (Page 665)
- USS_CTRL (Page 667)
- USS_RPM_X (Page 670)
- USS_WPM_x (Page 673)

USS protocol program examples (Page 675) and a listing of USS protocol error codes (Page 677) are also discussed in this section.

12.8.2.2 USS_INIT instruction

Table 12-30 USS_INIT instruction

LAD / FBD	STL	Description
	**CALL USS_INIT, Mode, Baud, Port, Active, Done, Error**	The USS_INIT instruction is used to enable and initialize, or to disable Siemens drive communications. Before any other USS instruction can be used, the USS_INIT instruction must be executed without errors. The instruction completes and the "Done" bit is set immediately, before continuing to the next instruction.

The instruction is executed on each scan when the "EN" input is on.

Execute the USS_INIT instruction only once for each change in communications state. Use an edge detection instruction to pulse the "EN" input on. To change the initialization parameters, execute a new USS_INIT instruction.

Table 12-31 Parameters for the USS_INIT instruction

Inputs/outputs	Data type	Operands
Mode, Port	BYTE	VB, IB, QB, MB, SB, SMB, LB, AC, Constant, *VD, *AC, *LD
Baud, Active	DWORD	VD, ID, QD, MD, SD, SMD, LD, Constant, AC *VD, *AC, *LD
Done	BOOL	I, Q, M, S, SM, T, C, V, L
Error	BYTE	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD

Table 12-32 USS_INIT parameter descriptions

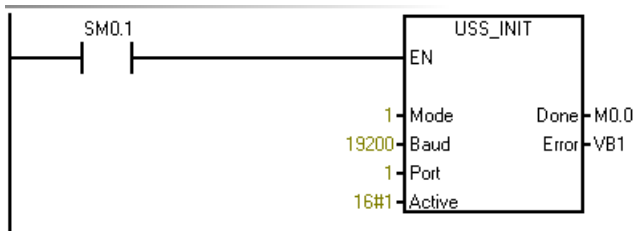
Parameter	Description
Mode	This value selects the communications protocol: • An input value of 1 assigns the port to USS protocol and enables the protocol. • An input value of 0 assigns the port to PPI and disables the USS protocol.
Baud	Sets the baud rate at 1200, 2400, 4800, 9600, 19200, 38400, 57600, or 115200
Port	Sets the physical communication port (0 = RS485 integrated in CPU, 1 = RS485 or RS232 located on the first optional CM01 signal board, 2 = RS485 or RS232 located on the second optional CM01 signal board)
Active	Indicates which drives are active. Some drives only support addresses 0 through 30. When its value is set to 0, all drivers are activated.
Done	Turned on when the USS_INIT instruction completes
Error	This output byte contains the result of executing the instruction. The USS protocol execution error codes (Page 677) define the error conditions that could result from executing the instruction.

Table 12-33 Format for the Active drive parameter

MSB LSB 31 30 29 28 3 2 1 0 D31 D30 D29 D2 D1 D0	This figure shows the description and format of the active drive input. Any drive that is marked as "Active" is automatically polled in the background to control the drive, collect status, and prevent serial link time-outs in the drive.
• D0 (Drive 0 active bit): – 0 - drive not active – 1 - drive active • D1 (Drive 1 active bit): – 0 - drive not active – 1 - drive active • ...	

Refer to the USS protocol execution error codes (Page 677) to compute the time between status polls and the error conditions that could result from executing the instruction.

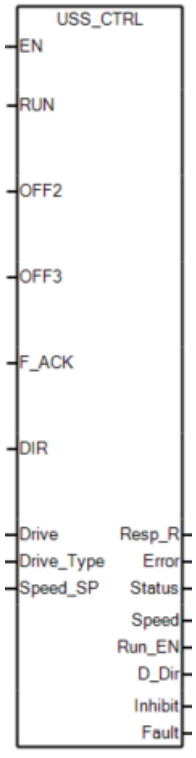
Table 12-34 USS_INIT example program

Network 1	Network 1
	**LD SM0.1** **CALL USS_INIT, 1, 19200, 1, 16#1, M0.0, VB1**

Refer to "Using the USS protocol instructions" (Page 664) for and a listing of USS protocol instructions and error codes and example programs.

12.8.2.3 USS_CTRL instruction

Table 12-35 USS_CTRL instruction

LAD / FBD	STL	Description
	**CALL USS_CTRL, RUN, OFF2, OFF3, F_ACK, DIR, Drive, Type, Speed_SP, Resp_R, Error, Status, Speed, Run_EN, D_Dir, Inhibit, Fault**	The USS_CTRL instruction is used to control an active Siemens drive. The USS_CTRL instruction places the selected commands in a communications buffer, which is then sent to the addressed drive ("Drive" parameter), if that drive has been selected in the "Active" parameter of the USS_INIT instruction.

Only one USS_CTRL instruction should be assigned to each drive.

Some drives report speed only as a positive value. If the speed is negative, the drive reports the speed as positive, but reverses the "D_Dir" (direction) bit.

The "EN" bit must be on to enable the USS_CTRL instruction. This instruction should always be enabled.

Table 12-36 Parameters of the USS_CTRL instruction

Inputs/outputs	Data types	Operands
RUN, OFF 2, OFF 3, F_ACK, DIR	BOOL	I, Q, M, S, SM, T, C, V, L, Power Flow
Resp_R, Run_EN, D_Dir, Inhibit, Fault	BOOL	I, Q, M, S, SM, T, C, V, L
Drive, Type	BYTE	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD, Constant
Error	BYTE	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD
Status	WORD	VW, T, C, IW, QW, SW, MW, SMW, LW, AC, AQW, *VD, *AC, *LD
Speed_SP	REAL	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *AC, *LD, Constant
Speed	REAL	VD, ID, QD, MD, SD, SMD, LD, AC, *VD, *AC, *LD

RUN parameter

RUN (RUN/STOP) indicates whether the drive is on (1) or off (0). When the "RUN" bit is on, the drive receives a command to start running at the specified speed and direction. In order for the drive to run, the following must be true:

- Drive must be selected as "Active" in USS_INIT.
- "OFF2" and "OFF3" must be set to 0.
- "Fault" and "Inhibit" must be 0.

When "RUN" is off, a command is sent to the drive to ramp the speed down until the motor comes to a stop:

- The "OFF2" bit is used to allow the drive to coast to a stop.
- The "OFF3" bit is used to command the drive to stop quickly.

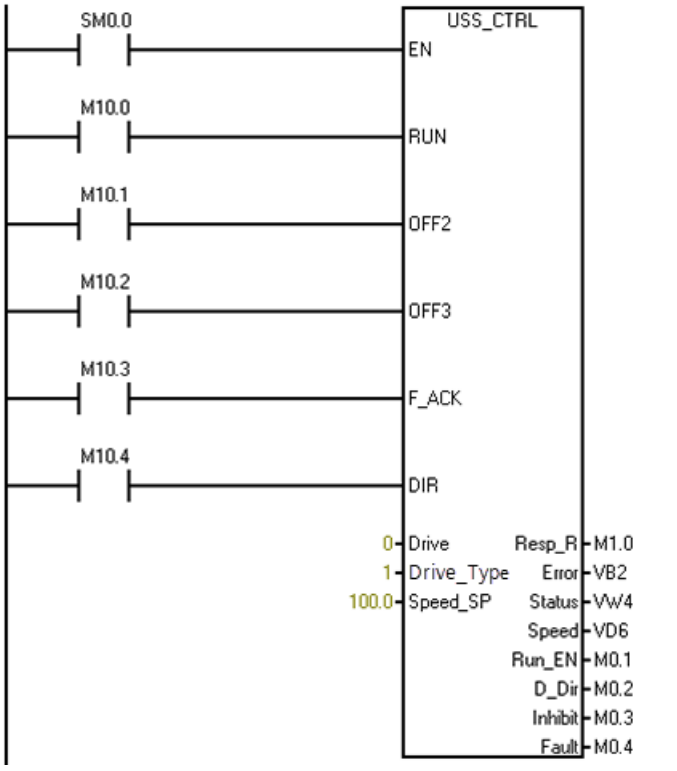
Resp_R parameter

The "Resp_R" (response received) bit acknowledges a response from the drive. All the Active drives are polled for the latest drive status information. Each time the CPU receives a response from the drive, the "Resp_R" bit is turned on for one scan and all the following values are updated:

Parameter	Description
F_ACK (fault acknowl-edge)	Bit that acknowledges a fault in the drive. The drive clears the fault ("Fault" bit) when "F_ACK" goes from 0 to 1.
DIR (direction)	Bit that indicates in which direction the drive should move.
Drive (drive address)	Input for the address of the drive to which the USS_CTRL command is to be sent. Valid addresses: 0 to 31
Type (drive type)	Input that selects the type of drive

Parameter	Description
Speed_SP (speed setpoint)	Drive speed as a percentage of full speed: • Negative values of "Speed_SP" cause the drive to reverse its direction of rotation. • Range: -200.0% to 200.0%
Error	Byte that contains the result of the latest communications request to the drive. The USS protocol execution error codes (Page 677) define the error conditions that could result from executing the instruction.
Status	Raw value of the status word returned by the drive. The figures below show the status bits for standard status word and main feedback.
Speed	Drive speed as a percentage of full speed. Range: -200.0% to 200.0%
Run_EN (RUN enable)	Indicates the drive condition: • Running (1) • Stopped (0)
D_Dir	Indicates the drive's direction of rotation
Inhibit	Indicates the state of the "Inhibit" bit on the drive: • 0: not inhibited • 1: inhibited To clear the "Inhibit" bit, the following bits must be OFF: • "Fault" • "RUN" • "OFF2" • "OFF3"
Fault	Indicates the state of the "Fault" bit: • 0: no fault • 1: fault The drive displays the fault code. (Refer to the manual for your drive). To clear the "Fault" bit, correct the cause of the fault and turn on the "F_ACK" bit.

Table 12-37 USS_CTRL example program



To display in LAD or FBD:

```
Network 1 // Control box for drive 0
LD SM0.0
= L60.0
LD M10.0
= L63.7
LD M10.1
= L63.6
LD M10.2
= L63.5
LD M10.3
= L63.4
LD M10.4
= L63.3
LD L60.0
CALL USS_CTRL, L63.7, L63.6, L63.5, L63.4,
L63.3, 0, 1, 100.0, M1.0, VB2, VW4, VD6,
M0.1, M0.2, M0.3, M0.4
```

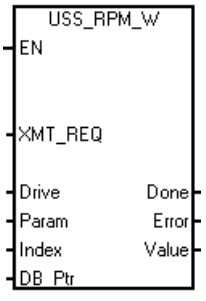
To display in STL only:

```
Network 1 // Control box for drive 0
LD SM0.0
CALL USS_CTRL, M10.0, M10.1, M10.2, M10.3,
M10.4, 0, 1, 100.0, M1.0, VB2, VW4, VD6,
M0.1, M0.2, M0.3, M0.4
```

Refer to "Using the USS protocol instructions" (Page 664) for a listing of USS protocol instructions and error codes and example programs.

12.8.2.4 USS_RPM_x instruction

Table 12-38 USS_RPM_x instructions

LAD / FBD	STL	Description
	**CALL USS_RPM_W, XMIT_REQ, Drive, Param, Index, DB_Ptr, Done, Error, Value** **CALL USS_RPM_D, XMIT_REQ, Drive, Param, Index, DB_Ptr, Done, Error, Value** **CALL USS_RPM_R, XMIT_REQ, Drive, Param, Index, DB_Ptr, Done, Error, Value**	There are three read instructions for the USS protocol: • USS_RPM_W instruction reads an unsigned word parameter. • USS_RPM_D instruction reads an unsigned double word parameter. • USS_RPM_R instruction reads a floating-point parameter. • Only one read (USS_RPM_x) or write (USS_WPM_x) instruction can be active at a time.

The USS_RPM_x transactions complete when the drive acknowledges receipt of the command or when an error condition is posted. The logic scan continues to execute while this process awaits a response.

Table 12-39 Valid operands for the USS_RPM_x instructions

Inputs/outputs	Data type	Operands
XMT_REQ	BOOL	I, Q, M, S, SM, T, C, V, L, Power Flow conditioned by a rising edge detection element
Drive	BYTE	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD, Constant
Param, Index	WORD	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, AIW, *VD, *AC, *LD, Constant
DB_Ptr	DWORD	&VB
Value	WORD DWORD, REAL	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, AQW, *VD, *AC, *LD VD, ID, QD, MD, SD, SMD, LD, *VD, *AC, *LD
Done	BOOL	I, Q, M, S, SM, T, C, V, L
Error	BYTE	VB, IB, QB, MB, SB, SMB, LB, AC. *VD, *AC, *LD

The "EN" bit must be on to enable transmission of a request, and should remain on until the "Done" bit is set, signaling completion of the process. For example, a USS_RPM_x request is transmitted to the drive on each scan when the "XMT_REQ" input is on. Therefore, the "XMT_REQ" input should be pulsed on through an edge detection element which causes one request to be transmitted for each positive transition of the "EN" input.

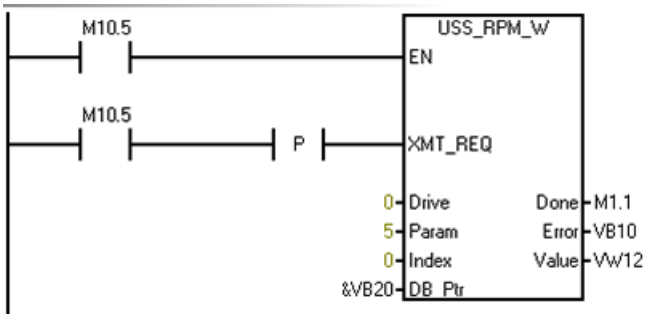
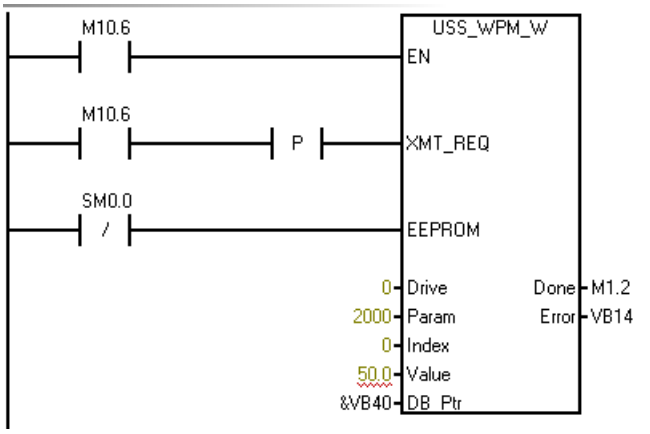
Table 12-40 USS_RPM_x parameter descriptions

Parameter	Description
XMT_REQ (transmit re-quest)	When ON, a USS_RPM_x request is transmitted to the drive on every scan.
Drive	Address of the drive to which the USS_RPM_x command is to be sent. Valid addresses of individual drives are 0 to 31.
Param	Parameter number
Index	Index value of the parameter that is to be read
DB_Ptr	The address of a 16-byte buffer must be supplied to the "DB_Ptr" input. This buffer is used by the USS_RPM_x instruction to store the results of the command issued to the drive.
Done	Turned on when the USS_RPM_x instruction completes
Error	This output byte contains the result of executing the instruction. The USS protocol execution error codes (Page 677) define the error conditions that could result from executing the instruction.
Value	Parameter value returned

When the USS_RPM_x instruction completes, the "Done" output is turned on and the "Error" output byte and the "Value" output contain the results of executing the instruction. The "Error" and "Value" outputs are not valid until the "Done" output turns on.

USS_RPM_x and USS_WPM_x example program

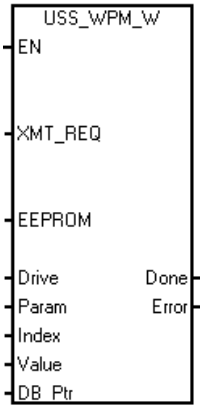
Table 12-41 USS_RPM_x and USS_WPM_x example program

Network 1	Network 1
	**LD M10.5** **= L60.0** **LD M10.5** **EU** **= L63.7** **LD L60.0** **CALL USS_RPM_W, L63.7, 0, 5, 0, &VB20, M1.1, VB10, VW12**
Network 2	**Network 2**
	**LD M10.6** **= L60.0** **LD M10.6** **EU** **= L63.7** **LDN SM0.0** **= L63.6** **LD L60.0** **CALL USS_WPM_W, L63.7, L63.6, 0, 2000, 0, 50.0, &VB40, M1.2, VB14**

Refer to "Using the USS protocol instructions" (Page 664) for and a listing of USS protocol instructions and error codes and example programs.

12.8.2.5 USS_WPM_x instruction

Table 12-42 USS_WPM_x instructions

LAD / FBD	STL	Description
	**CALL USS_WPM_W, XMT_REQ, EEPROM, Drive, Param, Index, Value, DB_Ptr, Done, Error** **CALL USS_WPM_D, XMT_REQ, EEPROM, Drive, Param, Index, Value, DB_Ptr, Done, Error** **CALL USS_WPM_R, XMT_REQ, EEPROM, Drive, Param, Index, Value, DB_Ptr, Done, Error**	There are three write instructions for the USS protocol: • USS_WPM_W instruction writes an unsigned word parameter. • USS_WPM_D instruction writes an unsigned double word parameter. • USS_WPM_R instruction writes a floating-point parameter. Only one read (USS_RPM_x) or write (USS_WPM_x) instruction can be active at a time.

The USS_WPM_x transactions complete when the drive acknowledges receipt of the command or when an error condition is posted. The logic scan continues to execute while this process awaits a response.

Table 12-43 Valid operands for the USS_WPM_x instructions

Inputs/outputs	Data type	Operands
XMT_REQ	BOOL	I, Q, M, S,SM,T,C,V,L, Power Flow conditioned by a rising edge detection element
EEPROM	BOOL	I, Q, M, S, SM, T, C, V, L, Power Flow
Drive	BYTE	VB, IB, QB, MB, SB, SMB, LB, AC, *VD, *AC, *LD, Constant
Param, Index	WORD	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, AIW, *VD, *AC, *LD, Constant
DB_Ptr	DWORD	&VB
Value	WORD DWORD, REAL	VW, IW, QW, MW, SW, SMW, LW, T, C, AC, AQW, *VD, *AC, *LD VD, ID, QD, MD, SD, SMD, LD, *VD, *AC, *LD
Done	BOOL	I, Q, M, S, SM, T, C, V, L
Error	BYTE	VB, IB, QB, MB, SB, SMB, LB, AC. *VD, *AC, *LD

The "EN" bit must be on to enable transmission of a request, and should remain on until the "Done" bit is set, signaling completion of the process. For example, a USS_WPM_x request is transmitted to the drive on each scan when "XMT_REQ" input is on. Therefore, the "XMT_REQ" input should be pulsed on through an edge detection element which causes one request to be transmitted for each positive transition of the "EN" input.

Table 12-44 USS_WPM_x parameter descriptions

Parameter	Description
XMT_REQ (transmit request)	When ON, a USS_WPM_x request is transmitted to the drive on every scan.
EEPROM	This input enables writing to both RAM and EEPROM of the drive when it is on and only to the RAM when it is off.

Parameter	Description
Drive	Address of the drive to which the USS_WPM_x command is to be sent. Valid addresses of individual drives are 0 to 31.
Param	Parameter number
Index	Index value of the parameter that is to be written
Value	Parameter value to be written to the RAM in the drive.
DB_Ptr	The address of a 16-byte buffer must be supplied to the "DB_Ptr" input. This buffer is used by the USS_RPM_x instruction to store the results of the command issued to the drive.
Done	Turned on when the USS_RPM_x instruction completes
Error	This output byte contains the result of executing the instruction. The USS protocol execution error codes (Page 677) define the error conditions that could result from executing the instruction.

When the USS_WPM_x instruction completes, the "Done" output is turned on and the "Error" output byte contains the result of executing the instruction. The "Error" output is not valid until the "Done" output turns on.

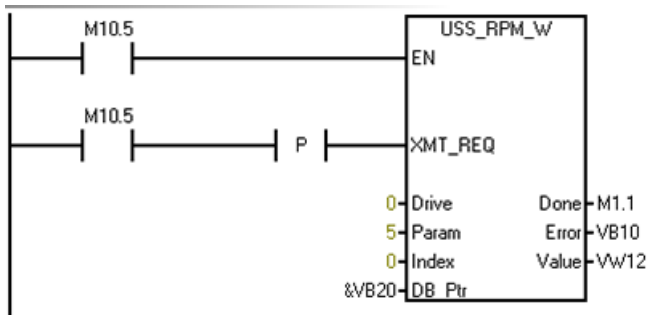
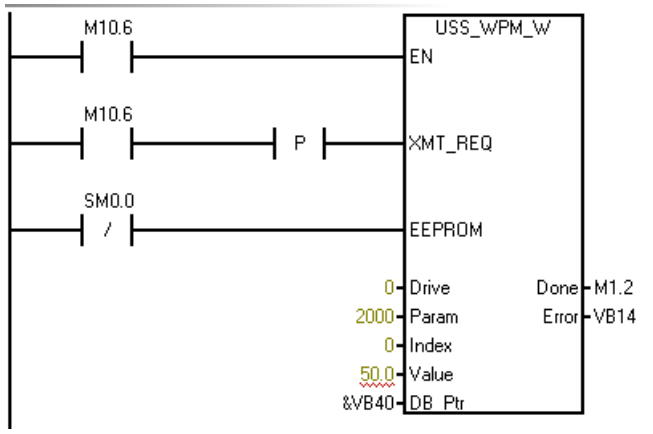
EEPROM

When the "EEPROM" input is turned on, the instruction writes to both the RAM and the EEPROM of the drive. When the input is turned off, the instruction writes only to the RAM of the drive.

NOTICE
Do not exceed the maximum number of write cycles to the EEPROM When you use an USS_WPM_x instruction to update the parameter set stored in drive EEPROM, you must ensure that the maximum number of write cycles (approximately 50,000) to the EEPROM is not exceeded. Exceeding the maximum number of write cycles will result in corruption of the stored data and subsequent data loss, and possible property damage. The number of read cycles is unlimited. Do not exceed the maximum number of write cycles to the EEPROM.

USS_RPM_x and USS_WPM_x example program

Table 12-45 USS_RPM_x and USS_WPM_x example program

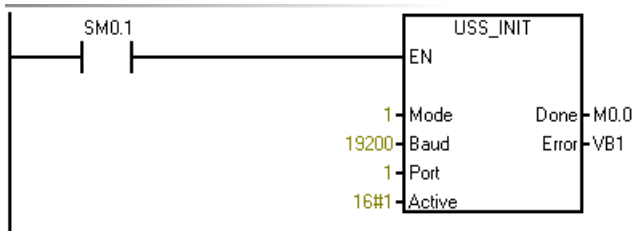
Network 1	Network 1
	**LD M10.5** **= L60.0** **LD M10.5** **EU** **= L63.7** **LD L60.0** **CALL USS_RPM_W, L63.7, 0, 5, 0, &VB20, M1.1, VB10, VW12**
Network 2	**Network 2**
	**LD M10.6** **= L60.0** **LD M10.6** **EU** **= L63.7** **LDN SM0.0** **= L63.6** **LD L60.0** **CALL USS_WPM_W, L63.7, L63.6, 0, 2000, 0, 50.0, &VB40, M1.2, VB14**

Refer to "Using the USS protocol instructions" (Page 664) for and a listing of USS protocol instructions and error codes and example programs.

12.8.2.6 USS protocol example program

Table 12-46 Sample USS program

Network 1

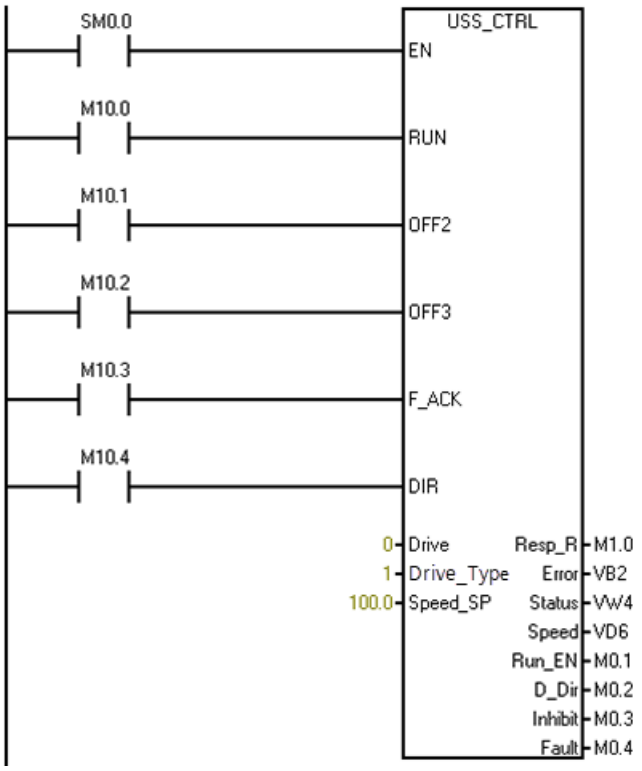


Network 1:
Initialize USS protocol: On the first scan, enable USS protocol for port 1 at 19200 with drive address "0" active.

```
LD SM0.1
CALL USS_INIT, 1, 19200, 16#00000001,
Q0.0, VB1
```

Network 2

Network 2:
Control parameters for Drive 0

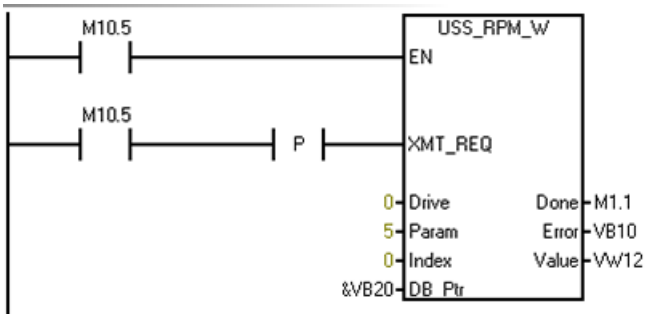



```
LD SM0.0
CALL USS_CTRL, M10.0, M10.1, M10.2,
M10.3, M10.4, 0, 1, 100.0, M1.0, VB2,
VW4, VD6, M0.1, M0.2, M0.3, M0.4
```

Network 3

Network 3:
Read a Word parameter from Drive 0.
Read parameter 5, index 0:

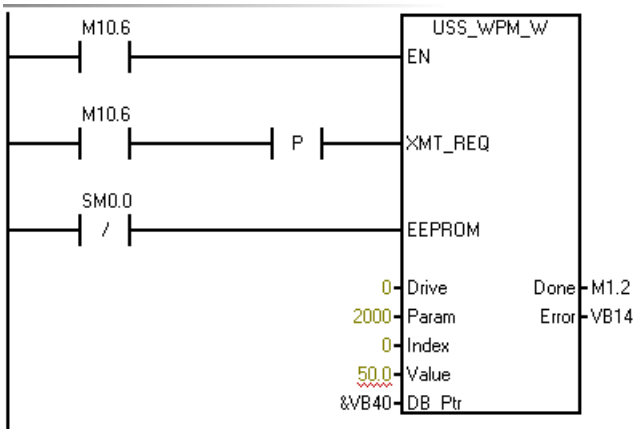
1. Save the state of M10.5 to a temporary location so that this network displays in LAD.
2. Save the rising edge pulse of I0.5 to a temporary L location so that it can be passed to the subroutine.




```
LD M10.5
= L60.0
LD M10.5
EU
= L63.7
LD L60.0
CALL USS_RPM_W, L63.7, 0, 5, 0, &VB20,
M1.1, VB10, VW12
```

Network 4

Network 4:
Write a Word parameter to Drive 0. Write parameter 2000, index 0.
Note: This STL code does not compile to LAD or FBD.




```
LD M10.6
= L60.0
LD M10.6
EU
= L63.7
LDN SM0.0
= L63.6
LD L60.0
CALL USS_WPM_R, L63.7, L63.6, 0, 2000, 0,
50.0, &VB40, M1.2, VB14
```

Refer to "Using the USS protocol instructions" (Page 664) for and a listing of USS protocol instructions and error codes and example programs.

12.8.3 USS protocol execution error codes

Table 12-47 USS protocol execution error codes

Error code	Description
0	No error
1	Drive did not respond.
2	A checksum error in the response from the drive was detected.
3	A parity error in the response from the drive was detected.
4	An error was caused by interference from the user program.
5	An illegal command was attempted.
6	An illegal drive address was supplied.
7	The communications port was not set up for USS protocol.
8	The communications port is busy processing an instruction.
9	The drive speed input is out-of-range.
10	The length of the drive response is incorrect.
11	The first character of the drive response is incorrect.
12	The length character in the drive response is not supported by USS instructions.
13	The wrong drive responded.
14	The DB_Ptr address supplied is incorrect.
15	The parameter number supplied is incorrect.
16	An invalid protocol was selected.
17	USS is active; change is not allowed.
18	An illegal baud rate was specified.
19	No communications: the drive is not ACTIVE.

Error code	Description
20	The parameter or value in the drive response is incorrect or contains an error code.
21	A double word value was returned instead of the word value requested.
22	A word value was returned instead of the double word value requested.
23	Invalid port number
24	Signal board (SB) port 1 is missing or not configured.
25	Signal board (SB) port 2 is missing or not configured.

Refer to "Using the USS protocol instructions" (Page 664) for and a listing of USS protocol instructions and error codes and example programs.

12.9 SINAMICS Library

12.9.1 SINAMICS library features

The SINAMICS library includes pre-configured subroutines that make controlling the drives easier. You can control the position and speed of physical drive, and read or modify the drive parameters with the SINAMICS library.

STEP 7-Micro/WIN SMART provides the following two groups of SINAMICS library instructions:

- SINAMICS_Control:
  - SINA_POS (Page 679): Control the drive position with 8 different operating modes
  - SINA_SPEED (Page 720): Control the drive speed
- SINAMICS_Parameter:
  SINA_PARA_S (Page 727): Read the parameters from the drive or modify the parameters to the drive

Open the Libraries folder in the Instruction folder of the project tree for access to the SINAMICS library instructions. When you place a SINAMICS library instruction in your program, STEP 7-Micro/WIN SMART V3 places one or more associated subroutines in your project.

SINAMICS_Control

SINAMICS_Control uses the following program entities:

- 2 subroutines:
  - SINA_POS
  - SINA_SPEED
- 3867 bytes of program space
- A 188-byte block of V memory for the instruction symbols

SINAMICS_Parameter

SINAMICS_Parameter uses the following program entities:

- 4 subroutines:
  - PN_RD_REC_PARA_S
  - PN_WR_REC_PARA_S
  - SINA_PARA_S
  - ERROR_HANDLER
- 5050 bytes of program space
- A 1314-byte block of V memory for the instruction symbols

12.9.2 SINA_POS instruction

12.9.2.1 Overview

EPOS of third generation supports not only LU (length unit) but also physical units. EPOS positions, speeds, accelerations, and jerks refer to the load-side (not motor-side). In the second generation, all positions were represented as 32-Bit integer values, whereas in the third generation all positions are represented as 32-Bit float values. So, in the third generation LU has fractional digits as well, but the maximum position range without loss of precision will be smaller. The effective resolution of 32-Bit float value is 24-Bit.

SIEMENS telegram 111 is the solution for the 32-Bit integer values and SIEMENS telegram 112 is the solution for the 32-Bit float values. In the second generation, only the SIMENS telegram 111 can be used and the function block SINA_POS is used to control the drives. In the third generation, the SIEMENS telegram 111 also supported and SINA_POS also can be used to control the drive. But in the third generation, the SIEMENS telegram 112 can be used and a new function block SINA_POS_112 is developed for the controlling.

12.9.2.2 Prerequisite of using the SINA_POS instruction

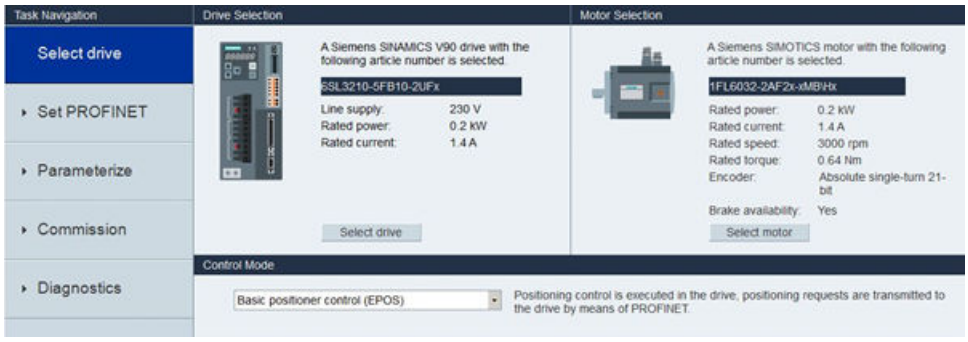
The prerequisite for using the SINA_POS instruction is as follows:

- The SINAMICS V90 PN drive and the servo motor are ready.
- The PROFINET network is connected between the drive and the S7-200 SMART CPU.
- The software V-assistant is connected with SINAMICS V90 PN. For the detailed information, refer to SINAMICS V-ASSISTANT Online Help (https://support.industry.siemens.com/cs/ww/en/view/109738316). You can download the SINAMICS V-ASSISTANT software (https://support.industry.siemens.com/cs/ww/en/view/109738387) and the SINAMICS V90: PROFINET GSD file (https://support.industry.siemens.com/cs/ww/en/view/109737269) in the SIEMENS Industry Online Support Website.

Procedure for configuring SINAMICS V90 PN parameters with the V-assistant

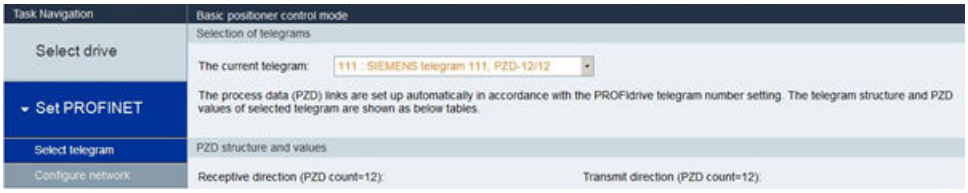
Configure the SINAMICS V90 PN parameters with the V-assistant as follows:

1. Start the V-assistant software.
2. Click "Online" to select the working mode.
3. Click the connected drive and click the "OK" button.
4. Click "Select drive" from the navigation tree, and select "Basic positioner control (EPOS)" in the "Control Mode" field.



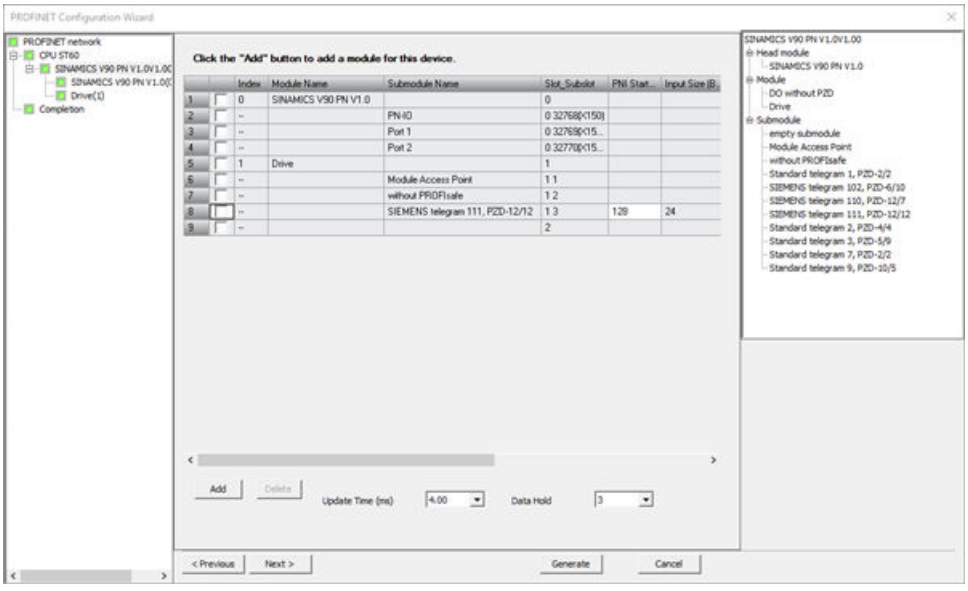
5. Click "Set PROFINET" from the navigation tree and click "Select telegram".

6. In the "Selection of telegram" field, select "SIEMENS telegram 111" as the current telegram. The SINA_POS instruction only supports SIEMENS telegram 111.



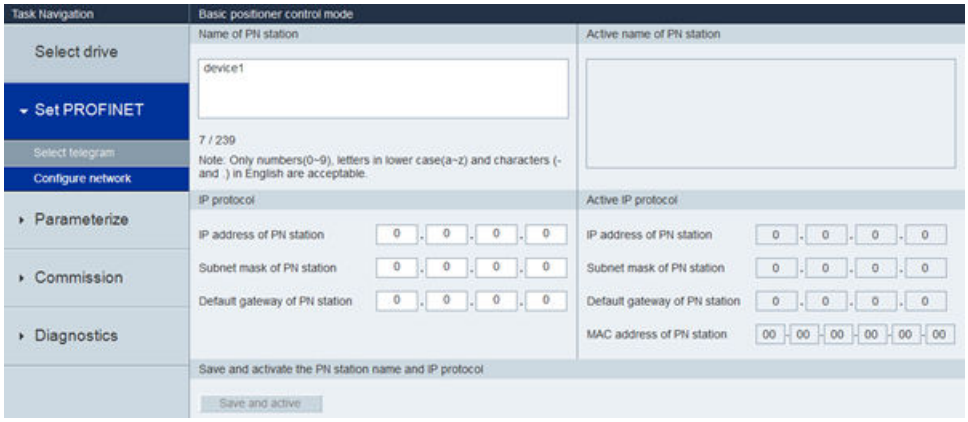
Note

When you configure the PROFINET network in PROFINET wizard, keep the telegram consistent with that in this step.



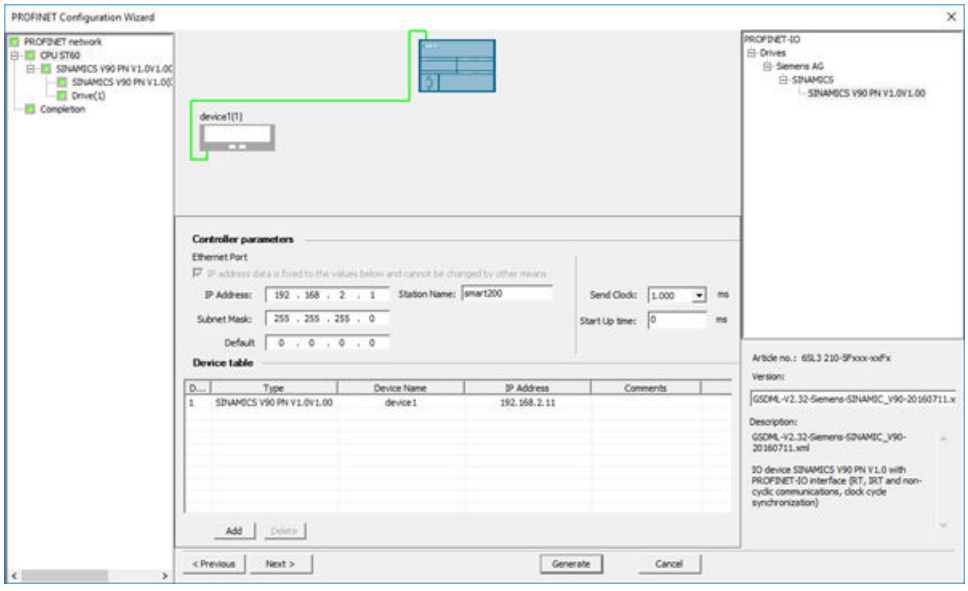
7. Click "Configure network" from the navigation tree.

8. Define the PN station name in the "Name of PN station" field.



Note

When you configure the PROFINET network in PROFINET wizard, keep the device name consistent with the PN station name in this step.



9. Click the "Save and active" button.

Result: The drive automatically restarts and the configured parameters take effect.

12.9.2.3 Input and output interface of SINA_POS instruction

The SINA_POS instruction can control and set the position for drives.

Table 12-48 SINA_POS instruction

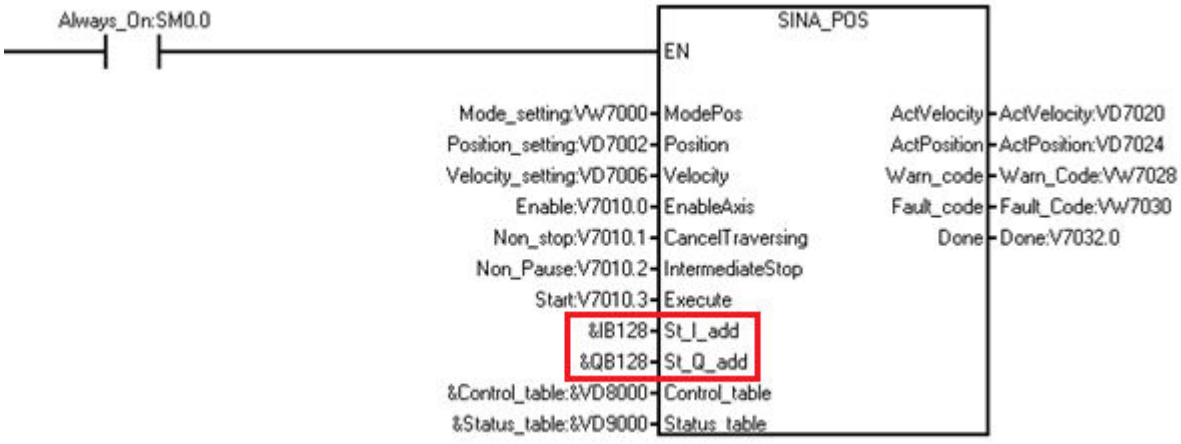
LAD/ FBD	STL	Description
SINA_POS EN ModePos — ActVelocity Position — ActPosition Velocity — Warn_code EnableAxis — Fault_code CancelTraversing — Done IntermediateStop Execute St_I_add St_Q_add Control_table Status_table	CALL SINA_POS, ModePos, Position, Velocity, EnableAxis, CancelTraversing, IntermediateStop, Execute, St_I_add, St_Q_add, Control_table, Status_table, ActVelocity, ActPosition, Warn_code, Fault_code, Done	The SINA_POS instruction enables the position control of the drive movement.

Note

For the four inputs "St_I_add", "St_Q_add", "Control_table", and "Status_table", the mode of addressing instruction operands is the indirect addressing.

You must enter an ampersand (&) at the beginning of the input operand and keep the offset consistent with that in the PROFINET wizard.

You can take the Relative positioning for example:



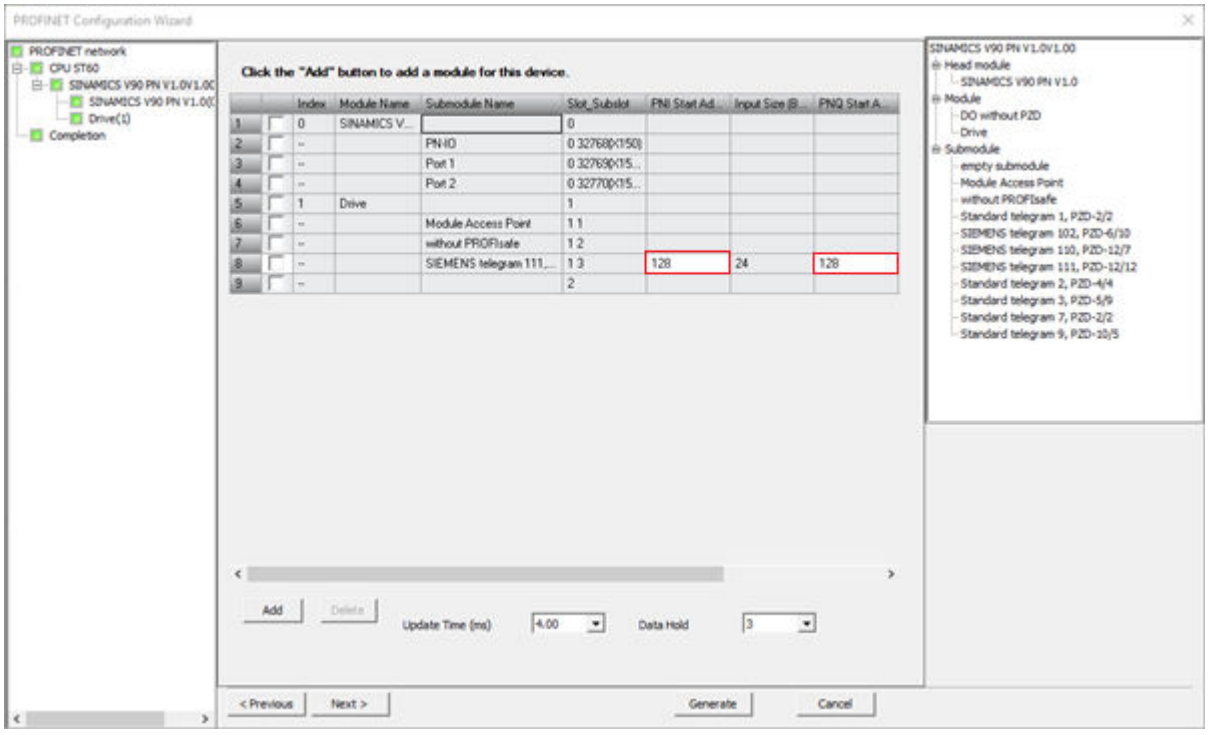


Table 12-49 Parameters of SINA_POS instruction

Parameter and type		Data type	Description
ModePos	IN	INT	Operating mode: 1 = relative positioning 2 = absolute positioning 3 = positioning as setup 4 = referencing (active homing) 5 = referencing (set reference point) 6 = traversing block 0 – 15 7 = jog mode 8 = incremental jog 9 = encoder adjustment (for 112 only)
Position	IN	111: DINT	Position setpoint in [LU] for direct setpoint input / MDI mode or traversing block number for traversing block mode. (Default = 0)
		112: REAL	Position setpoint in physical unit for MDI or traversing clock number for operating mode traversing block
Velocity	IN	111: DINT	Velocity in [LU/min] for MDI mode. (Default value = 0 [1000LU/min])
		112: REAL	Velocity setpoint in physical unit for MDI operating mode
EnableAxis	IN	BOOL	Switching command: 0 = OFF, 1 = ON
CancelTraversing	IN	BOOL	0 = reject active traversing task 1 = do not reject (Default)

Parameter and type		Data type	Description
IntermediateStop	IN	BOOL	0 = active traversing command is interrupted 1 = no intermediate stop (Default)
Execute	IN	BOOL	Activate traversing task/setpoint acceptance/ activate reference function.
St_I_add	IN	DWORD	Pointer of I memory area start address for PROFINET IO. For example, &IB128.
St_Q_add	IN	DWORD	Pointer of Q memory area start address for PROFINET IO. For example, &QB128.
Control_table	IN	DWORD	Pointer of the start address of control_table (Page 685). For example, &VD8000.
Status_table	IN	DWORD	Pointer of the start address of Status_table (Page 688). For example, &VD9000.
ActVelocity	OUT	111: DWORD	Actual velocity
		112: REAL	Actual velocity (MU)
ActPosition	OUT	111: DWORD	Actual position in LU
		112: REAL	Actual position in MU
Warn_code	OUT	WORD	The warning code information from V90. For detailed information, refer to *SINAMICS V90, SIMOTICS S-1FL6 Operating Instruction*.
Fault_code	OUT	WORD	The fault code information from V90. For detailed information, refer to *SINAMICS V90, SIMOTICS S-1FL6 Operating Instruction*.
Done	OUT	BOOL	Target position is reached when the operating mode is relative positioning or absolute positioning.

Definition of "Control_table" parameters

The parameters of "Control_table" are as follows:

Table 12-50 Control_table parameter for telegram 111

Byte Offset	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	Reserved	Reserved	AckError [1]	FlyRef [2]	Jog2 [3]	Jog1 [4]	Negative [5]	Positive [6]
1	Reserved							
2	OverV [7]							
3								
4	OverAcc [8]							
5								
6	OverDec [9]							
7								

Byte Offset	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
8	ConfigEpos [10]							
9								
10								
11								

Table 12-51 Control_table parameter for telegram 112

Byte Offset	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	Reserved	Reserved	AckError [1]	FlyRef [2]	Jog2 [3]	Jog1 [4]	Negative [5]	Positive [6]
1	Reserved							
2	OverV [7]							
3								
4	Acc							
5								
6								
7								
8	Dec							
9								
10								
11								
12	fixStopTorque							
13								
14								
15								
16	refCoordinate							
17								
18								
19								
20	ConfigEpos [10]							
21								
22								
23								

[1] AckError: Acknowledging errors. (1= Acknowledging errors is valid, 0 =Acknowledging errors is invalid)

[2] FlyRef: flying reference selection (1 = Reference point setting is activated, 0 = Reference point setting is not activated)

[3] Jog2: Jog signal source 2 (1= positive jog is activated, 0 = positive jog is not activated)

[4] Jog1: Jog signal source 1 (1= negative jog is activated, 0 = negative jog is not activated)

[5] Negative: Negative direction (1 = negative rotation is activated, 0 = negative rotation is not activated)

[6] Positive: Positive direction (1 = positive rotation is activated, 0 = positive rotation is not activated)

[7] OverV: Velocity override is active for all modes. The value range is 0%- 199% and the default value is 100%. For example, you can set OverV as 60%.

[8] OverAcc: Acceleration override is active. The value range is 0%-100% and the default value is 100%. For example, you can set OverAcc as 70%.

[9] OverDec: Deceleration override is active. The value range is 0%-100% and the default value is 100%. For example, you can set OverAcc as 50%.

[10] ConfigEpos: One input to control the EPos functions that are not directly specified at the block. For the detailed information, refer to Description of the configuration input "ConfigEPos" (Page 687).

Description of the configuration input "ConfigEPos"

The following table lists the bit mapping between "ConfigEpos" and "Telegram 111":

ConfigEpos	Telegram 111
ConfigEPos.%X0	STW1.%X1
ConfigEPos.%X1	STW1.%X2
ConfigEPos.%X2	EPosSTW2.%X14
ConfigEPos.%X3	EPosSTW2.%X15
ConfigEPos.%X4	EPosSTW2.%X11
ConfigEPos.%X5	EPosSTW2.%X10
ConfigEPos.%X6	EPosSTW2.%X2
ConfigEPos.%X7	STW1.%X13
ConfigEPos.%X8	EPosSTW1.%X12
ConfigEPos.%X9	STW2.%X0
ConfigEPos.%X10	STW2.%X1
ConfigEPos.%X11	STW2.%X2
ConfigEPos.%X12	STW2.%X3
ConfigEPos.%X13	STW2.%X4
ConfigEPos.%X14	STW2.%X7
ConfigEPos.%X15	STW1.%X14
ConfigEPos.%X16	STW1.%X15
ConfigEPos.%X17	EPosSTW1.%X6
ConfigEPos.%X18	EPosSTW1.%X7
ConfigEPos.%X19	EPosSTW1.%X11
ConfigEPos.%X20	EPosSTW1.%X13
ConfigEPos.%X21	EPosSTW2.%X3
ConfigEPos.%X22	EPosSTW2.%X4
ConfigEPos.%X23	EPosSTW2.%X6
ConfigEPos.%X24	EPosSTW2.%X7
ConfigEPos.%X25	EPosSTW2.%X12
ConfigEPos.%X26	EPosSTW2.%X13

ConfigEpos	Telegram 111
ConfigEPos.%X27	STW2.%X5
ConfigEPos.%X28	STW2.%X6
ConfigEPos.%X29	STW2.%X8
ConfigEPos.%X30	STW2.%X9

Definition of "Status_table" parameters

The definition of the "Status_table" bits is as follows:

Table 12-52 Status_table

Byte offset	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	Reserved	Over-range_Error [1]	AxisError [2]	AxisWarn [3]	Lockout [4]	AxisRef [5]	AxisPosOk [6]	Axisenabled [7]
1	For telegram 111: Error ID [8]							
	For telegram 112: Error ID [8], AxisSpFixed							
2	Actmode [9]							
3								
4	Epos_zsw1 [10]							
5								
6	Epos_zsw2 [11]							
7								

[1] Overrange_Error: The data you enter is out of the range. For detailed information, refer to Error code 3, 4, 5 (Page 689).

[2] AxisError: The drive has an error. (Default = 0)

[3] AxisWarn: Drive alarm is active. (Default = 0)

[4] Lockout: Switching-on inhibit. (Default = 0)

[5] AxisRef: Reference point set. (Default = 0)

[6] AxisPosOk: Target position of the axis is reached. (Default = 0)

[7] Axisenabled: Drive is ready and switched on. (Default = 0)

[8] Error ID: Identify the error type. For the detailed information, refer to Error codes for the "Status_table" parameter (Page 689).

[9] Actmode: Currently active mode. (Default = 0)

[10] Epos_zsw1: Status of EPos_zsw1 (bit-granular). For the detailed information, refer to Assignment of "Epos_zsw1" (Page 689).(Default = 0)

[11] Epos_zsw2: Status of EPos_zsw2 (bit-granular). For the detailed information, refer to Assignment of "Epos_zsw2" (Page 690).(Default = 0)

Error codes for the "Status_table" parameter

The following table lists the error code of the "Status_table" parameter:

Table 12-53 Error codes for the "Status_table" parameter

Error Code	Description
0	No error.
1	An error from the drive is detected.
2	The drive is disabled.
3	The selected mode is not supported.
4	The rate of parameters OverV, OverAcc and OverDec exceeds the supported value range.
5	The selected block is out of range under the motion mode "traversing block".

Assignment of "Epos_zsw1"

The following table lists the assignment information of "Epos_zsw1":

Table 12-54 Epos_zsw1

Bit	Abbr.	Designation	Drive parameter	Function chart
0	ActTrvBit0	Active traversing block, bit 0	r2670.0	3650
1	ActTrvBit1	Active traversing block, bit 1	r2670.1	3650
2	ActTrvBit2	Active traversing block, bit 2	r2670.2	3650
3	ActTrvBit3	Active traversing block, bit 3	r2670.3	3650
4	ActTrvBit4	Active traversing block, bit 4	r2670.4	3650
5	ActTrvBit5	Active traversing block, bit 5	r2670.5	3650
6	Bit6	Reserved		
7	Bit7	Reserved		
8	StpCamMinAct	STOP cam minus active	r2684.13	3630
9	StpCamPlsAct	STOP cam plus active	r2684.14	3630
10	JogAct	Jog mode is active	r2094.0 [1]	2460
11	RefAct	Reference point approach mode active	r2094.1 [1]	2460
12	FlyRefAct	Flying referencing active	r2684.1 [1]	3630
13	TrvBlAct	Traversing blocks mode active	r2094.2 [1]	2460
14	MdiStupAct	In the direct setpoint input / MDI mode, setup is active	r2094.4 [1]	2460
15	MdiPosAct	In the direct setpoint input / MDI mode, positioning is active	r2094.3 [1]	2460

[1] r2669 (function diagram 3630) displays bit-granular. P2099[0] = r2699 is interconnected at the input of the connector-bivector converter for this purpose.

Assignment of "Epos_zsw2"

The following table lists the assignment information of "Epos_zsw2":

Table 12-55 Epos_zsw2

Bit	Abbr.	Designation	Drive parameter	Function chart
0	TrkModeAct	Follow-up/tracking mode active	r2683.0	3645
1	VeloLimAct	Velocity limitation active	r2683.1	3645
2	SetPStat	Setpoint static	r2683.2	3645
3	111: PrntMrkOut	Print mark outside outer window	r2683.3	3614
	112	Target position reached		
4	FWD	Axis moves forward	r2683.4	3635
5	BWD	Axis moves backward	r2683.5	3635
6	SftSwMinAct	Minus software limit switch actuated	r2683.6	3635
7	SftSwPlsAct	Plus software limit switch actuated	r2683.7	3635
8	PosSmCam1	Position actual value <= cam switching position 1	r2683.8	4025
9	PosSmCam2	Position actual value <= cam switching position 2	r2683.9	4025
10	TrvOut1	Direct output 1 with the traversing block	r2683.10	3616
11	TrvOut2	Direct output 2 with the traversing block	r2683.11	3616
12	FxStpRd	Fixed stop reached	<not used> (r2683.12)	3645
13	FxStpTrRd	Fixed stop clamping torque reached	<not used> (r2683.13)	3645
14	TrvFxStpAct	Travel to fixed stop active	<not used> (r2683.14)	3645
15	CmdAct	Traversing active	r2683.15	3645

12.9.2.4 Mode selection of SINAMICS with SINA_POS instruction

The "ModePos" input is used for the operating mode selection. There are eight operating modes:

- Relative positioning
- Absolute positioning
- Setup mode
- Referencing (active referencing)
- Referencing (set reference point)
- Traversing blocks
- Jog
- Incremental jog
- Encoder adjustment (for telegram 112 only)

Basic requirements

The input operands "CancelTraversing" and "IntermediateStop" are relevant for all modes except for jog and must be set to "1" when using SINAMICS.

In each operating mode, follow these steps to enable the drive:

- Set the input operand "CancelTraversing" to 1.
- Set the input operand "Intermediatestop" to 1.
- In the Control_table, set "ConfigEpos" to 3 according to Decimal numeral system.

To enable the axis, set the input operand "EnableAxis" to 1.

You can use the input operand "ModePos" to set or change the operating mode.

12.9.2.5 Relative positioning

Relative positioning mode enables the motor axis start the positioning motion relative to the start position. The distance is incremental in each movement.

The Relative positioning mode is implemented with the "MDI relative positioning" function of SINAMICS V90 PN. It enables the position-controlled traversing of traversing paths using the integrated position controller of the SINAMICS V90 PN.

Requirements

- The mode is selected with ModePos=1.
- The device is switched on with "EnableAxis".
- The axis does not have to be referenced or the encoder adjusted.
- The axis is at standstill if selected by an operating mode greater than 3. A change within the MDI operating modes (1,2,3) is possible at any time.

Sequence

You configure the traversing path and dynamic responses with the following inputs:

- "Position"
- "Velocity"
- "OverV" (velocity override)
-
  - For telegram 111: "OverAcc" (acceleration override)
  - For telegram 112: "Acc" (acceleration)
-
  - For telegram 111: "OverDec" (deceleration override)
  - For telegram 112: "Dec" (deceleration)

The velocity override refers to the "Velocity". For example, the velocity that takes effect is 500LU/min and the OverV is 120%, the velocity in SINAMICS V90 is 600LU/min.

You must set the signal inputs "CancelTraversing" and "IntermediateStop" to "1". "Jog1" and "Jog2" have no effect and you must set them to "0" (false).

The direction of travel in relative positioning always results from the sign of the traversing path. For example, if the position is -1000, the direction is negative. If the position is 1000, the direction is positive.

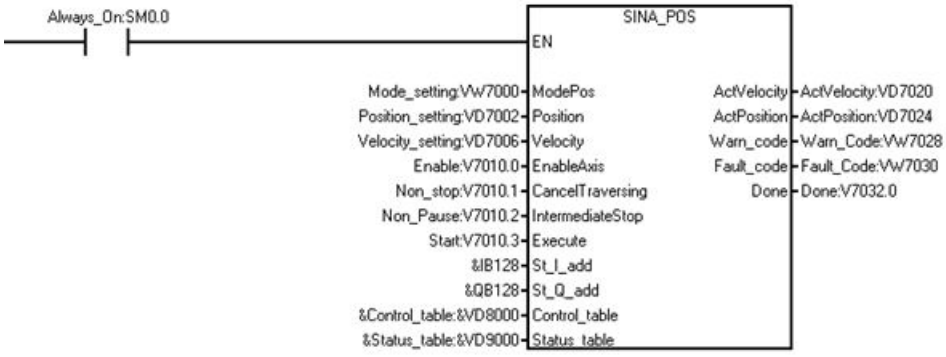
Traversing motion is started with a positive edge at "Execute". You can track the current state of the active command with "EPos_zsw1 / EPos_zsw2".

If the target position is reached, the value of bit "AxisPosOK" in the output signal "Status_table" is 1. If an error occurs during the traversing motion, the bit "AxisError" in the output signal "Status_table" is issued.

Note

The current command can be replaced on the fly by a new command with "ExecuteMode". This is only possible for the "ModePos" 1, 2 or 3 modes.

Example of the Relative positioning mode



Set the Relative positioning mode as follows:

1. Create the following variables and write the variables to the corresponding input operands.

Symbol	Address	Comment	Corresponding input operand	Data type	Value		
Mode_setting	VW7000	Mode selection	ModePos	WORD	1		
Position_setting	VD7002	Position length	Position	DWORD	2500		
Velocity_setting	VD7006	Velocity	Velocity	DWORD	500		
Enable	V7010.0	Enable the drive.	EnableAxis	BOOL	1		
Non_stop	V7010.1	The status is non-stop.	CancelTraversing	BOOL	1		
Non_Pause	V7010.2	No pausing.	IntermediateStop	BOOL	1		
Start	V7010.3	Start the drive.	Execute	BOOL	1		
Control_table	VD8000	The control parameters.	Control_table	DWORD	VW8002	OverV	100
					VW8004	OverAcc	100
					VW8006	OverDec	100
					VD8008	ConfigEpos	3

Telegram 111
For the cotrol table for telegram 112, refer to Definition of "Control_table" parameters (Page 685).

Note

The variable value in this example is for your reference, and you need to create the variable according to your actual situation.

2. Enter the data in the inputs "St_I_add" and "St_Q_add".

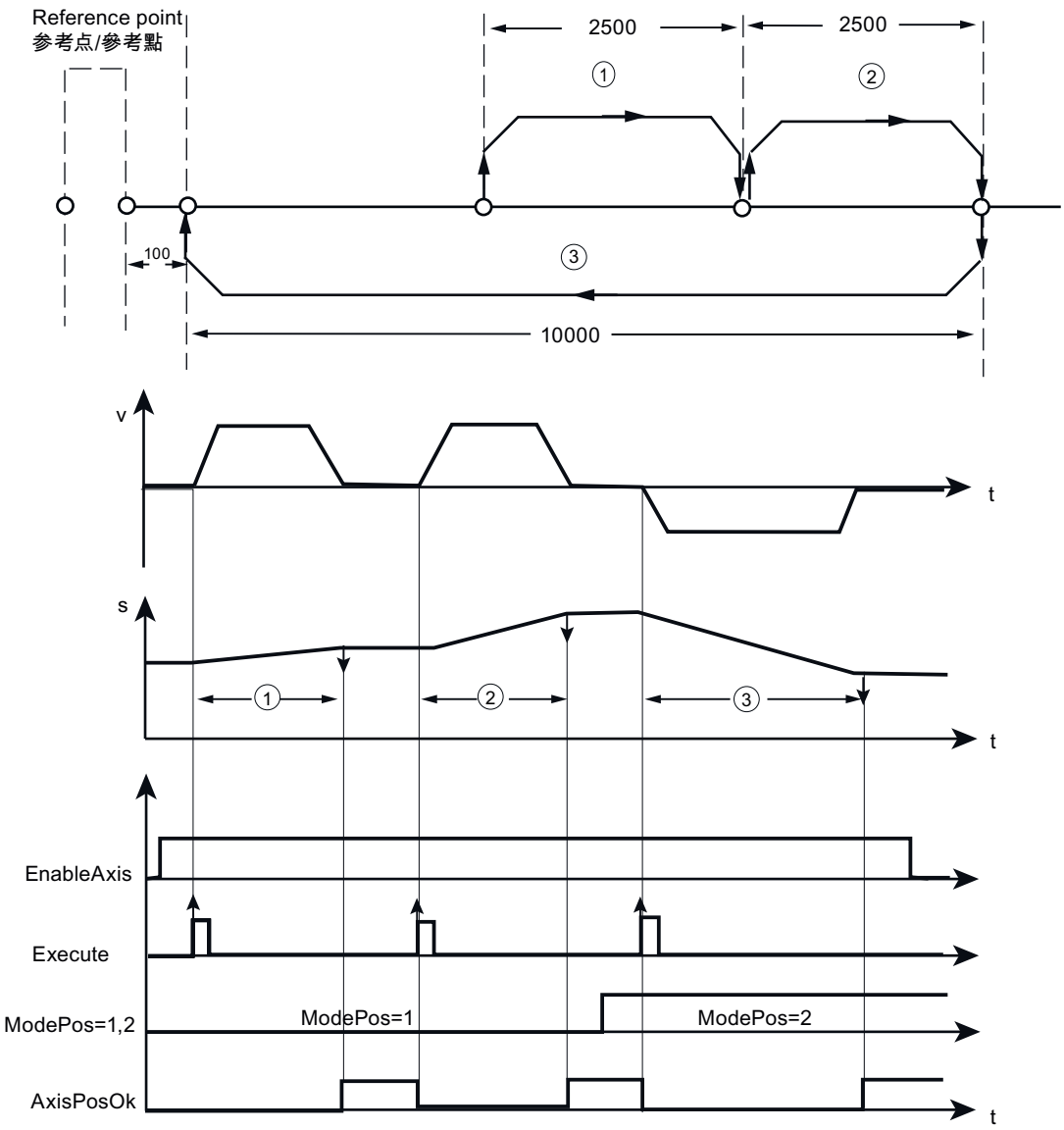
Note

For the four inputs "St_I_add", "St_Q_add", "Control_table", and "Status_table", the mode of addressing instruction operands is the indirect addressing. You must enter an ampersand (&) at the beginning of the input operand.

For the input operand "St_I_add" and "St_Q_add", keep the offset consistent with that in the PROFINET wizard.

Result: The drive moves 2500 LU on the basis of the previous position distance. For example, if the previous distance is 5000 LU, the drive stays at the distance of 7500 LU.

The following illustration displays the movement trajectory and dynamic parameters:



① Relative positioning
② Relative positioning
③ Absolute positioning

In this illustration, "v" refers to velocity, "s" refers to position, and "t" refers to time.

12.9.2.6 Absolute positioning

The Absolute positioning mode enables the motor axis start the positioning motion to an absolute position.

The Absolute positioning mode is implemented with the "MDI absolute positioning" function of SINAMICS V90 PN. It enables the absolute movement using the integrated position controller of the SINAMICS V90 PN.

Requirements

- The mode is selected with "ModePos"=2.
- The device is switched on with "EnableAxis".
- The axis must be referenced or the encoder adjusted.
- The axis is at a standstill if selected by an operating mode greater than 3. A change within the MDI operating modes (1,2 or 3) is possible at any time.

Sequence

The traversing path and dynamic responses are specified with the following inputs:

- "Position"
- "Velocity"
- "OverV" (velocity override)
-
  - For telegram 111: "OverAcc" (acceleration override)
  - For telegram 112: "Acc" (acceleration)
-
  - For telegram 111: "OverDec" (deceleration override)
  - For telegram 112: "Dec" (deceleration)

The velocity override refers to the "Velocity". For example, the velocity that takes effect is 500LU/min and the OverV is 120%, the velocity in SINAMICS V90 is 600LU/min.

You must set the input signal "CancelTraversing" and "IntermediateStop" to "1". "Jog1" and "Jog2" have no effect and you must set them to "0".

The direction of travel in Absolute positioning always results from the shortest distance to the target position. The inputs "Positive " and "Negative" are "0".

Note

You can use the parameters "Positive" or "Negative" to specify a preferred direction to approach the target position for an axis.

Note

You can select the absolute positioning direction with the following positioning control words:

- POS_STW1.9:
- POS_STW1.10:
  - 1 = Absolute positioning/MDI direction selectionpositive
  - 2 = Absolute positioning/MDI direction selectionnegative
  - 3 = Absolute positioning through the shortest distance

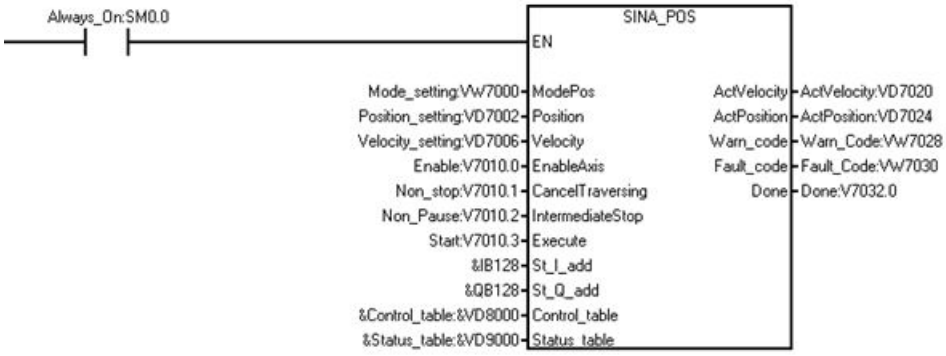
Traversing motion is started with a positive edge at "Execute". You can track the current state of the active command with "EPoszsw1 / EPoszsw2".

If the target position is reached, the value of bit "AxisPosOK" in the output signal "Status_table" is 1. If an error occurs during the traversing motion, the bit "AxisError" in the output signal "Status_table" is issued.

Note

The current command can be replaced on-the-fly by a new command via "Execute". This is only possible for the "ModePos" 1, 2, 3 modes.

Example of the Absolute positioning mode



Set the Absolute positioning mode as follows:

1. Create the following variables and write the variables to the corresponding input operands.
   Telegram 111:

Symbol	Address	Comment	Corresponding input operand	Data type	Value		
Mode_setting	VW7000	Mode selection	ModePos	WORD	2		
Position_setting	VD7002	Position length	Position	DINT	100		
Velocity_setting	VD7006	Velocity	Velocity	DINT	500		
Enable	V7010.0	Enable the drive.	EnableAxis	BOOL	1		
Non_stop	V7010.1	The status is non-stop.	CancelTraversing	BOOL	1		
Non_Pause	V7010.2	No pausing.	IntermidateStop	BOOL	1		
Start	V7010.3	Start the drive.	Execute	BOOL	1		
Control_table	VD8000	The control parameters.	Control_table	DWORD	VW8002	OverV	100
					VW8004	OverAcc	100
					VW8006	OverDec	100
					VD8008	ConfigEpos	3

Telegram 112:

Symbol	Address	Comment	Corresponding input operand	Data type	Value
Mode_setting	VW7000	Mode selection	ModePos	WORD	2
Position_setting	VD7002	Position length	Position	REAL	100.0
Velocity_setting	VD7006	Velocity	Velocity	REAL	500.0
Enable	V7010.0	Enable the drive.	EnableAxis	BOOL	1
Non_stop	V7010.1	The status is non-stop.	CancelTraversing	BOOL	1
Non_Pause	V7010.2	No pausing.	IntermidateStop	BOOL	1
Start	V7010.3	Start the drive.	Execute	BOOL	1

Symbol	Address	Comment	Corresponding input operand	Data type	Value		
Control_table	VD8000	The control parameters.	Control_table	DWORD	VW8002	OverV	100
					VD8004	Acc	100.0
					VD8008	Dec	100.0
					VD8012	fixStopTorque	100.0
					VD8016	refCoordinate	100.0
					VD8020	ConfigEpos	3

Note

The variable value in this example is for your reference, and you need to create the variable according to your actual situation.

2. Enter the data in the input "St_I_add" and "St_Q_add".

Note

For the four inputs "St_I_add", "St_Q_add", "Control_table", and "Status_table", the mode of addressing instruction operands is the indirect addressing. You must enter an ampersand (&) at the beginning of the input operand.

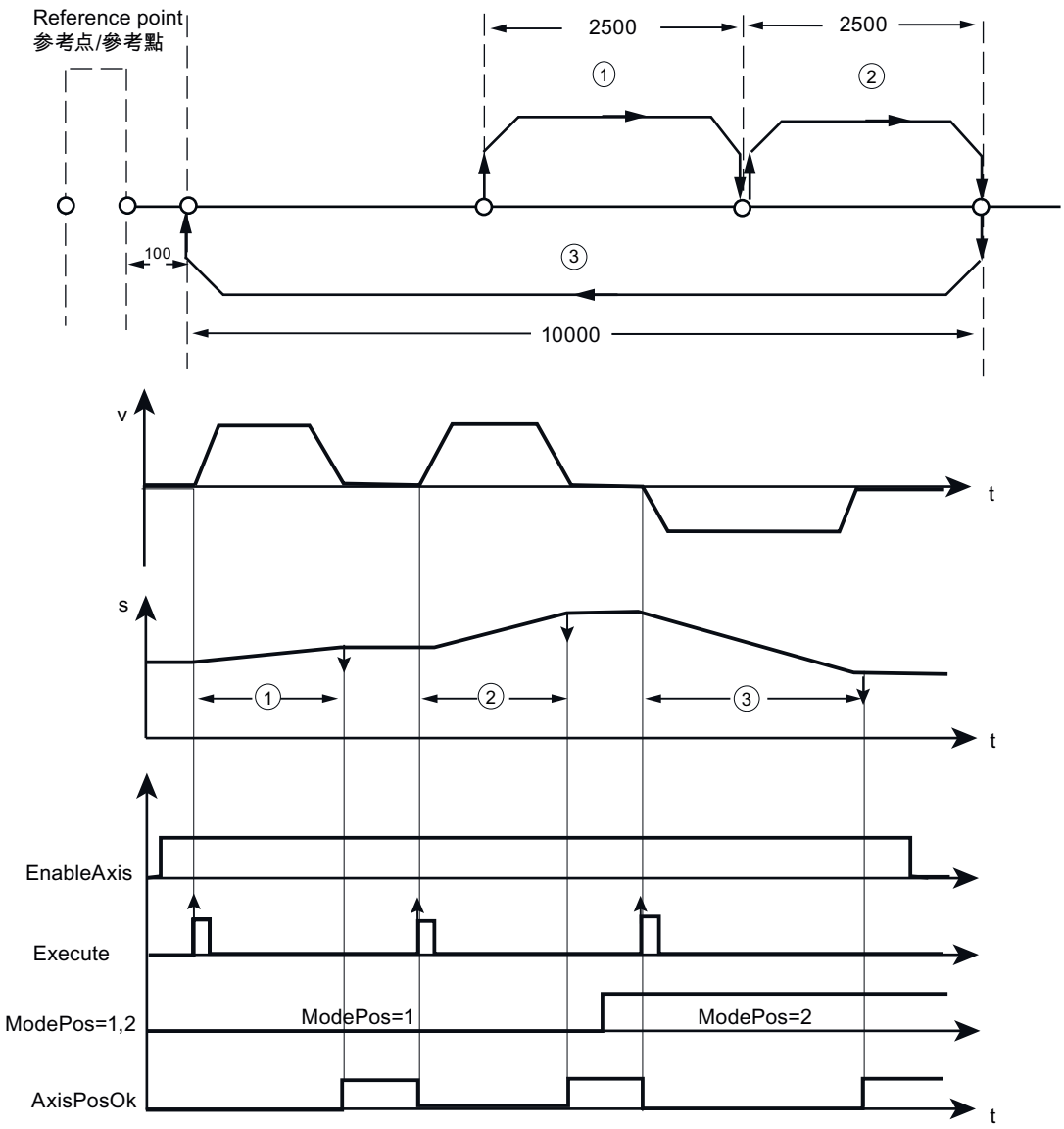
For the input operand "St_I_add" and "St_Q_add", keep the offset consistent with that in PROFINET wizard.

Result: Then the drive stays at the position of 100.

Note

For the operating mode "absolute positioning", you must set a valid reference position by configuring it in the mode "Referencing (active referencing)" or the "Referencing (set reference point)".

The following illustration displays the movement trajectory and dynamic parameters:



① Relative positioning
② Relative positioning
③ Absolute positioning

In this illustration, "v" refers to velocity, "s" refers to position, and "t" refers to time.

12.9.2.7 Setup mode

The Setup mode enables the position-controlled traversing of the axis in the positive or negative direction with constant velocity without specification of a target position with the "MDI set up" function of SINAMICS V90 PN.

Requirements

- The mode is selected with "ModePos" = 3.
- The device is switched on using "EnableAxis".
- The axis does not have to be referenced or the encoder adjusted.
- The axis is at a standstill if the operating mode is greater than 3. A change within the MDI operating modes (1, 2 or 3) is possible at any time.

Sequence

The traversing path and dynamic responses are specified with the following inputs:

- "Positive" or "Negative"
- "Velocity"
- "OverV" (velocity override)
- "OverAcc" (acceleration override)
- "OverDec" (deceleration override)

You must set the input signal "CancelTraversing" and "IntermediateStop" to "1". "Jog1" and "Jog2" have no effect and you must set them to "0".

The travel direction is determined via "Positive" and "Negative". Simultaneous selection stops the axis without further alarms or faults.

Traversing motion is started with a positive edge at "Execute". You can track the current state of the active command with "EPos_zsw1 / EPos_zsw2".

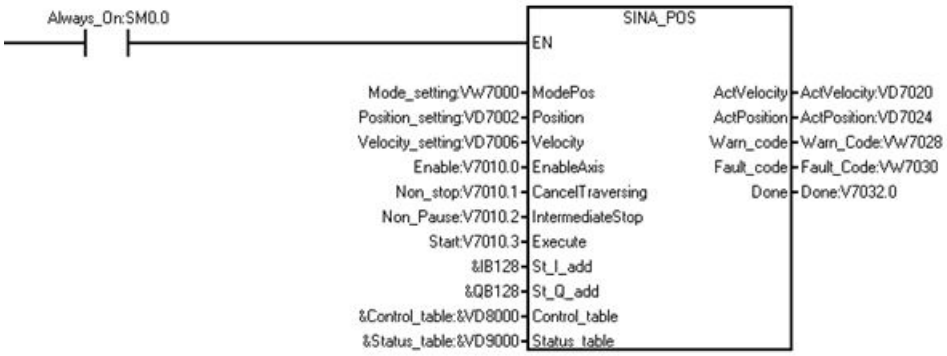
The output signal "AxisPosOk" is set when the setup mode is terminated with reject traversing task and the axis has stopped.

If an error occurs during the traversing motion, the bit "AxisError" in the output signal "Status_table" is issued.

Note

The current command can be replaced on-the-fly by a new command via "Execute". This is only possible for the "ModePos" 1, 2, 3 modes.

Example of the Setup mode



Set the Setup mode as follows:

1. Create the following variables and write the variables to the corresponding input operands.

Symbol	Address	Comment	Corresponding input operand	Data type	Value		
Mode_setting	VW7000	Mode selection	ModePos	WORD	3		
Velocity_setting	VD7006	Velocity	Velocity	DWORD	500		
Enable	V7010.0	Enable the drive.	EnableAxis	BOOL	1		
Non_stop	V7010.1	The status is non-stop.	CancelTraversing	BOOL	1		
Non_Pause	V7010.2	No pausing.	IntermediateStop	BOOL	1		
Start	V7010.3	Start the drive.	Execute	BOOL	1		
Control_table	VD8000	The control parameters.	Control_table	DWORD	VW8002	OverV	100
					VW8004	OverAcc	100
					VW8006	OverDec	100
					VD8008	ConfigEpos	3
					V8000.0 [1]	Positive	1
					V8000.1 [1]	Negative	0

[1] The value of V8000.0 and V8000.1 cannot be 1 or 0 at the same time.

Note

The variable value in this example is for your reference, and you need to create the variable according to your actual situation.

2. Enter the data in the input "St_I_add" and "St_Q_add".

Note

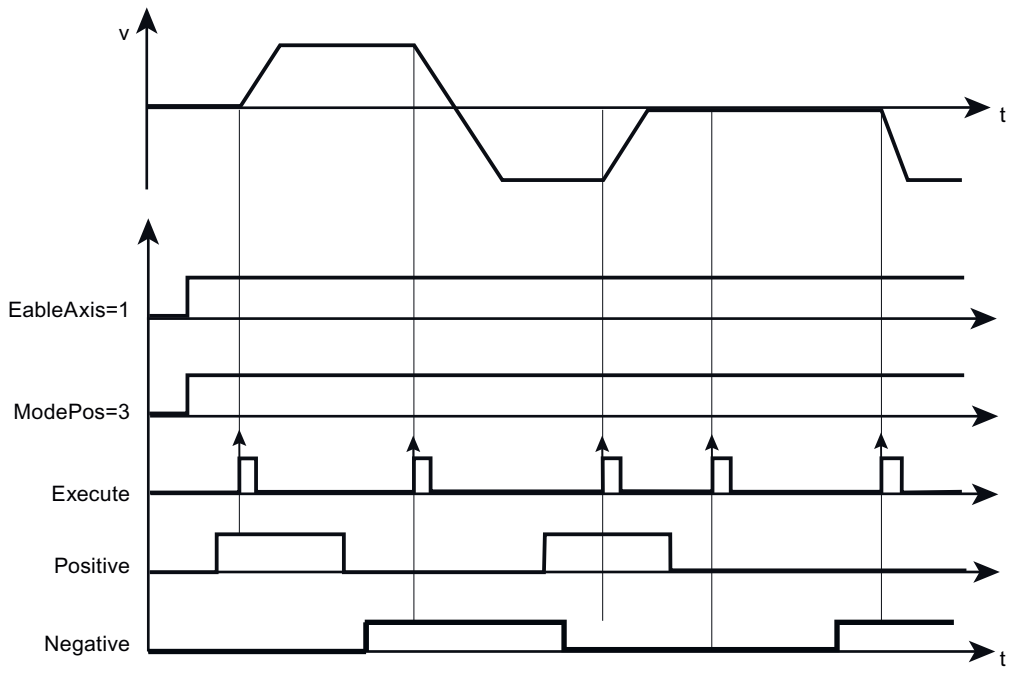
For the four inputs "St_I_add", "St_Q_add", "Control_table", and "Status_table", the mode of addressing instruction operands is the indirect addressing. You must enter an ampersand (&) at the beginning of the input operand.

For the input operands "St_I_add" and "St_Q_add", keep the offset consistent with that in the PROFINET wizard.

You have the following options for the variables:

- In the input signal "Control_table" :
  - To make the drive move toward the positive direction, set V8000.0 as 1 and V8000.1 as 0.
  - To make the drive moves toward the negative direction, set V8000.1 as 1 and V8000.0 as 0.
- To stop the drive, set the variable "Non_stop" to 0.
- To pause the drive, set the variable "Non_Pause" to 0.

The following picture displays the movement trajectory and dynamic parameters:



In this illustration, "v" refers to velocity and "t" refers to time.

12.9.2.8 Referencing (active referencing)

The Referencing (active referencing) mode enables the reference point approach of the axis in the positive or negative direction with predefined velocity and reference mode with the "Active referencing" function of SINAMICS V90 PN.

Requirements

- The mode is selected with "ModePos"=4.
- The device is switched on using "EnableAxis".
- The axis is at a standstill.

Sequence

The required velocity is saved as a velocity profile in the SINAMICS V90. Further, the preset acceleration and deceleration values are active in the traversing profile of the axis. The "OverV" velocity override affects the preconfigured traversing velocity. For example, the velocity that takes effect is 500LU/min and the OverV is 120%, the velocity in SINAMICS V90 is 600LU/min.

You must set the input signal "CancelTraversing" and "IntermediateStop" to "1". "Jog1" and "Jog2" have no effect and you must set them to "0".

The travel direction is determined via "Positive" and "Negative". Simultaneous selection is not permitted and results in a fault.

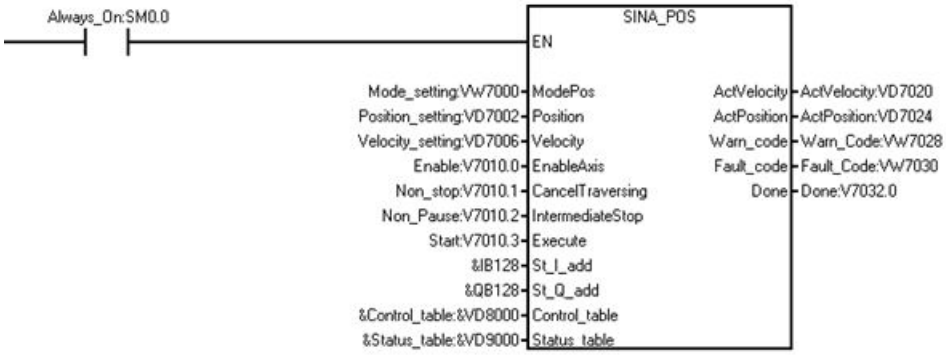
The reference point approach is started with a positive edge at "Execute".

Traversing motion is started with a positive edge at "Execute". You can track the current state of the active command with "EPos_zsw1 / EPos_zsw2".

The bit "AxisRef" in the output signal "Status_table" is set if the reference cam is appropriately found and evaluated.

If an error occurs during traversing motion, the bit "AxisError" in the output signal "Status_table" is issued.

Example of the Referencing (active referencing) mode



Set the operating mode "Referencing (active referencing)" as follows:

1. Create the following variables and write the variables to the corresponding input operands.

Symbol	Address	Comment	Corresponding input operand	Data type	Value		
Mode_setting	VW7000	Mode selection	ModePos	WORD	4		
Position_setting	VD7002	Position length	Position	DWORD	2500		
Velocity_setting	VD7006	Velocity	Velocity	DWORD	500		
Enable	V7010.0	Enable the drive.	EnableAxis	BOOL	1		
Non_stop	V7010.1	The status is non-stop.	CancelTraversing	BOOL	1		
Non_Pause	V7010.2	No pausing.	IntermediateStop	BOOL	1		
Start	V7010.3	Start the drive.	Execute	BOOL	1		
Control_table	VD8000	The control parameters.	Control_table	DWORD	V8000.0 [1]	Positive	1
					V8000.1 [1]	Negative	0
					VD8008	ConfigEpos	3
						V8011.6 (Bit 7)	1

[1] The value of V8000.0 and V8000.1 cannot be 1 or 0 at the same time.

Note

The variable value in this example is for your reference, and you need to create the variable according to your actual situation.

2. Enter the data in the input "St_I_add" and "St_Q_add".

Note

For the four inputs "St_I_add", "St_Q_add", "Control_table", and "Status_table", the mode of addressing instruction operands is the indirect addressing. You must enter an ampersand (&) at the beginning of the input operand.

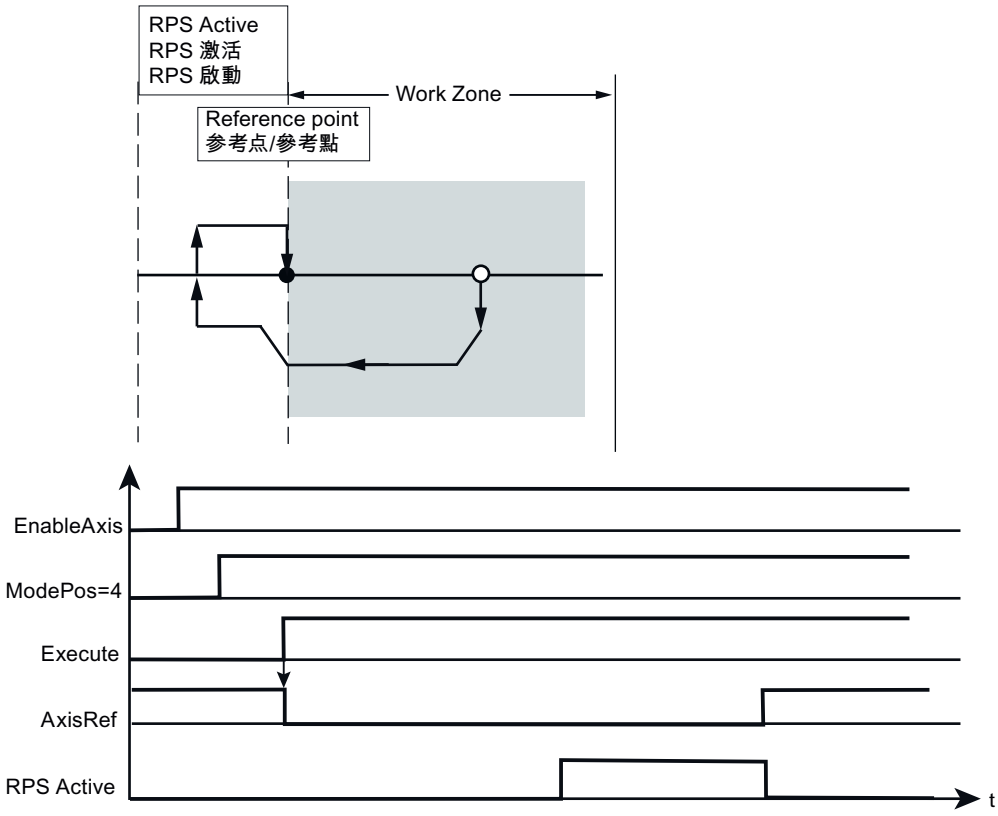
For the input operands "St_I_add" and "St_Q_add", keep the offset consistent with that in the PROFINET wizard.

3. In the variable "Control_table", if you set V8000.0 as 1 and V8000.1 as 0, the drive moves toward the positive direction to find the reference point. If you set V8000.1 as 1 and V8000.0 as 0, the drive moves toward the negative direction to find the reference point.

Note

When the external reference signal connects to PLC directly, you can set V8011.6 as 1 through digital inputs. The drive stops and you set the reference point successfully.

The following picture displays the movement trajectory and dynamic parameters:



In this illustration, "t" refers to time.

12.9.2.9 Referencing (set reference point)

The Referencing (set reference point) mode enables the referencing of the axis at an arbitrary position and is performed through the "Set reference point" function of SINAMICS V90 PN.

Requirements

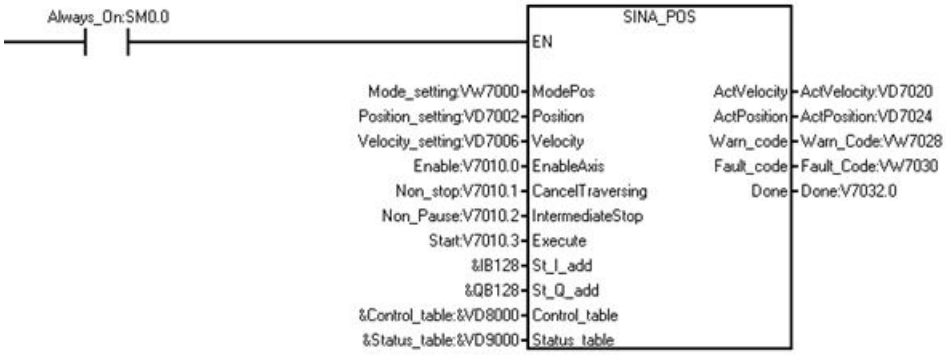
- The mode is selected with "ModePos"=5.
- The axis can be in closed-loop control, but must be at a standstill.

Sequence

The axis is at a standstill and the reference point is set with a positive edge at "Execute".

If an error occurs while setting the reference point, the bit "AxisError" in the output signal "Status_table" is issued.

Example of the Referencing (set reference point) mode



Set the operating mode as "Referencing (set reference point)" as follows:

1. Create the following variables and write the variables to the corresponding input operands.

Symbol	Address	Comment	Corresponding input operand	Data type	Value		
Mode_setting	VW7000	Mode selection	ModePos	WORD	5		
Position_setting	VD7002	Position length	Position	DWORD	2500		
Velocity_setting	VD7006	Velocity	Velocity	DWORD	500		
Enable	V7010.0	Enable the drive.	EnableAxis	BOOL	1		
Non_stop	V7010.1	The status is non-stop.	CancelTraversing	BOOL	1		
Non_Pause	V7010.2	No pausing.	IntermediateStop	BOOL	1		
Start	V7010.3	Start the drive.	Execute	BOOL	1		
Control_table	VD8000	The control parameters.	Control_table	DWORD	VD8008	ConfigEpos	3
Status_table	VD9000	The status parameters.	Status_table	DWORD	V9000.2	AxisRef	1

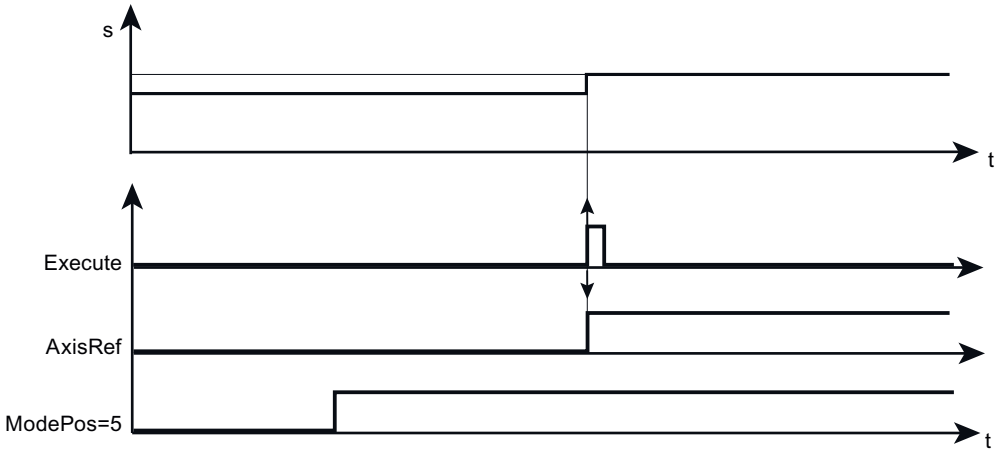
2. Enter the data in the input "St_I_add" and "St_Q_add".

Note

For the four inputs "St_I_add", "St_Q_add", "Control_table", and "Status_table", the mode of addressing instruction operands is the indirect addressing. You must enter an ampersand (&) at the beginning of the input operand.

For the input operands "St_I_add" and "St_Q_add", keep the offset consistent with that in PROFINET wizard.

The following picture displays the movement trajectory and dynamic parameters:



In this illustration, "s" refers to position, and "t" refers to time.

12.9.2.10 Traversing blocks

The Traversing blocks mode is implemented through the "Traversing blocks" function of SINAMICS V90 PN.

Requirements

- The mode is selected with "ModePos"=6.
- The device is switched on using "EnableAxis".
- The axis is at a standstill.
- The axis must be referenced or the encoder adjusted.

Sequence

Note

You can use the "Position" input to select the traversing task to start. The value for telegram 111 is from 0 to 15, and for telegram 112 is from 0 to 31.

If the value is out of the range, the error code 5 (Page 689)displays in the "Status_table" bit "Overrange_Error".

The traversing block parameters in the SINAMICS V90 specify the task modes, the target positions and dynamic responses. The velocity override "OverV" refers to the velocity setpoint stored in the traversing block. You must set the operating conditions "CancelTraversing" and "IntermediateStop" to "1". "Jog1" and "Jog2" have no effect and you must set them to "0".

The travel direction results from the task mode and the set position setpoint. The "Positive" and "Negative" are not relevant in this case and you must set them to "0".

Note

You can use the parameters "Positive" or "Negative" to specify a preferred direction to approach the target position for an axis.

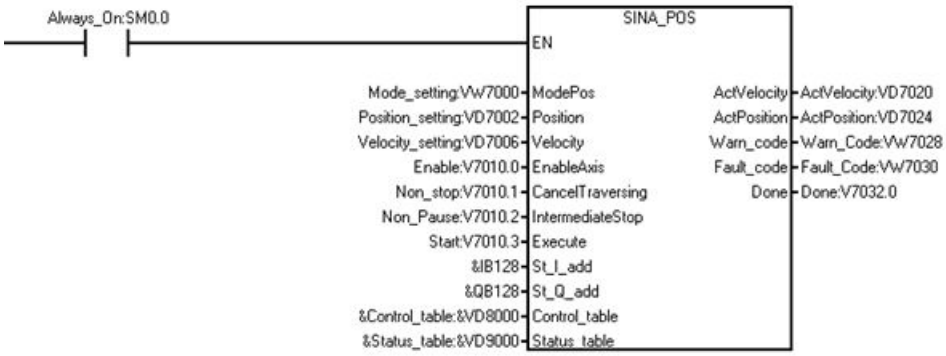
Traversing motion is started with a positive edge at "ExecuteMode". You can track the current state of the active command with "EPos_zsw1 / EPos_zsw2".

The SINA_POS instruction acknowledges when the end of the traversing path is reached successfully with "AxisPosOk". If an error occurs during the traversing motion, the bit "AxisError" in the output signal "Status_table" is issued.

Note

The current command can be replaced on-the-fly by a new command via "Execute". This is only possible for the same operating mode.

Example of the Traversing blocks mode



Set the operating mode as "Traversing blocks" as follows:

1. Create the following variables and write the variables to the corresponding input operands.

Symbol	Address	Comment	Corresponding input operand	Data type	Value		
Mode_setting	VW7000	Mode selection	ModePos	WORD	6		
Position_setting	VD7002	Position length	Position	DWORD	Enter the index of the desired blocks. The maximum supported blocks are 16, so the value in this variable is from 0 to 15.		
Enable	V7010.0	Enable the drive.	EnableAxis	BOOL	1		
Non_stop	V7010.1	The status is non-stop.	CancelTraversing	BOOL	1		
Non_Pause	V7010.2	No pausing.	IntermediateStop	BOOL	1		
Start	V7010.3	Start the drive.	Execute	BOOL	1		
Control_table	VD8000	The control parameters.	Control_table	DWORD	VD8008	Config-Epos	3

Note

The variable value in this example is for your reference, and you need to create the variable according to your actual situation.

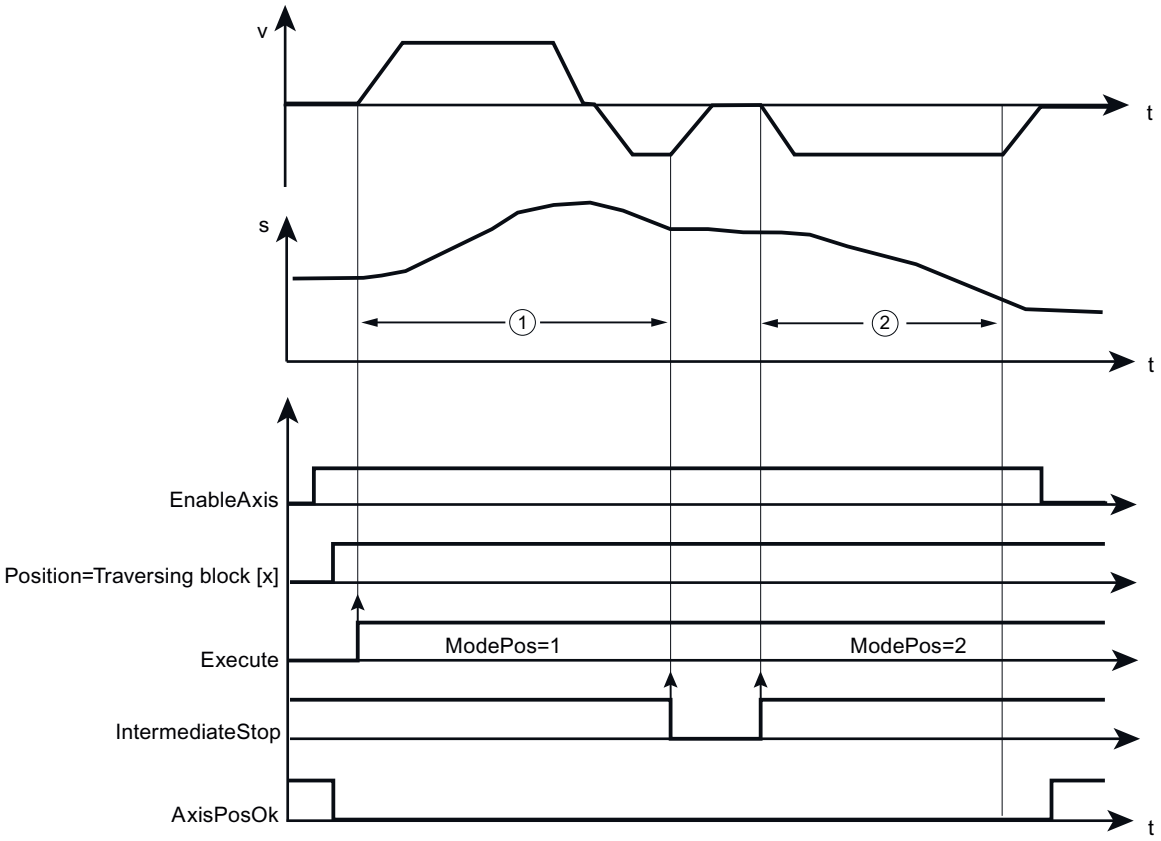
2. Enter the data in the input "St_I_add" and "St_Q_add".

Note

For the four inputs "St_I_add", "St_Q_add", "Control_table", and "Status_table", the mode of addressing instruction operands is the indirect addressing. You must enter an ampersand (&) at the beginning of the input operand.

For the input operands "St_I_add" and "St_Q_add", keep the offset consistent with that in the PROFINET wizard.

The following picture displays the movement trajectory and dynamic parameters:



In this illustration, "v" refers to velocity, "s" refers to position, and "t" refers to time.

12.9.2.11 Jog

The Jog mode is implemented using the "Jog" function of SINAMICS V90 PN. It enables the position-controlled, velocity-dependent traversing of axes using the integrated position controller of the SINAMICS V90 PN.

Requirements

- The mode is selected with "ModePos" = 7.
- The device is switched on using "EnableAxis".
- The axis is at standstill.
- The axis does not have to be referenced or adjusted.

Sequence

The specification of the jog velocity is performed via the V-assistant form or the SINA_PARA_S instruction for the configuration of the operating mode in the SINAMICS V90. The SINAMICS V90 uses the acceleration and deceleration for the dynamic responses of the axis.

The velocity override also applies in this operating mode and is set through "OverV". For example, the velocity that takes effect is 500LU/min and the OverV is 120%, the velocity in SINAMICS V90 is 600LU/min.

The input signal "CancelTraversing" and "IntermediateStop" are not relevant for the operating mode and you must set them to "1".

Note

"Jog1" and "Jog2" are the signal sources for the jog mode in SINA_POS. "Jog1" is the signal source for negative, and "Jog2" is the signal source for positive.

You can configure the distance of traversing motion in the SINAMICS V90 PN.

The velocity setpoint sets the travel direction for jogging.

The inputs "Positive" and "Negative" are not relevant for the operating mode and you can set them to "0".

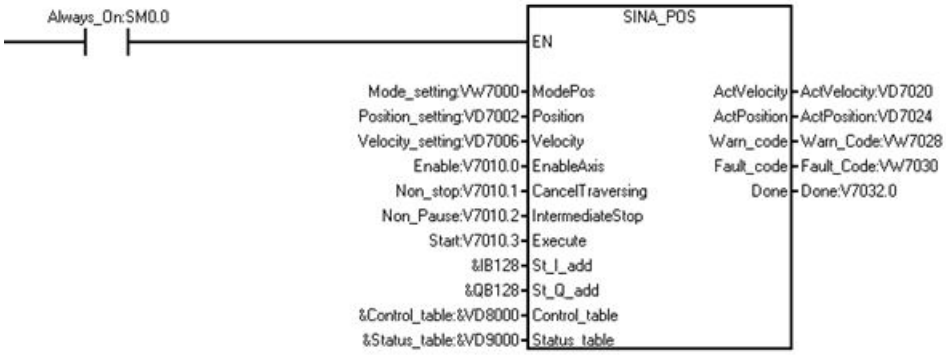
You can track the current state of the active command with "EPos_zsw1 / EPos_zsw2".

The SINA_POS instruction displays the current command processing with "AxisEnabled" and acknowledges the termination of the jog function ("Jog1" or "Jog2" = 0) when the axis is at standstill with "AxisPosOk". If an error occurs during the traversing motion, the bit "AxisError" in the output signal "Status_table" is issued.

Note

The current command can be replaced on-the-fly by a new command via "Jog1" or "Jog2". This is only possible for the "ModePos" 1, 2 or 3 mode.

Example of the Jog mode



Set the Jog mode as follows:

1. Create the following variables and write the variables to the corresponding input operands.

Symbol	Address	Comment	Corresponding input operand	Data type	Value		
Mode_setting	VW7000	Mode selection	ModePos	WORD	7		
Enable	V7010.0	Enable the drive.	EnableAxis	BOOL	1		
Non_stop	V7010.1	The status is non-stop.	CancelTraversing	BOOL	1		
Non_Pause	V7010.2	No pausing.	IntermediateStop	BOOL	1		
Control_table	VD8000	The control parameters.	Control_table	DWORD	V8000.2 [1]	Jog1	0
					V8000.3 [1]	Jog2	1
					VD8008	Config-Epos	3

[1] The value of V8000.2 and V8000.3 cannot be 1 or 0 at the same time.

Note

The variable value in this example is for your reference, and you need to create the variable according to your actual situation.

2. Enter the data in the input "St_I_add" and "St_Q_add".

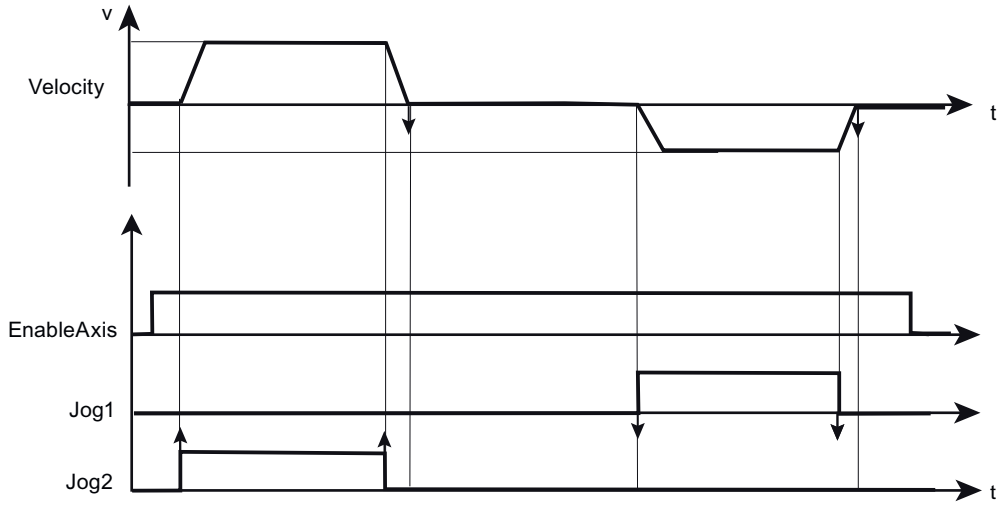
Note

For the four inputs "St_I_add", "St_Q_add", "Control_table", and "Status_table", the mode of addressing instruction operands is the indirect addressing. You must enter an ampersand (&) at the beginning of the input operand.

For the input operands "St_I_add" and "St_Q_add", keep the offset consistent with that in the PROFINET wizard.

3. In the variable "Control_table", if you set V8000.2 as 1 and V8000.3 as 0, the drive moves toward the negative direction. If you set V8000.3 as 1 and V8000.2 as 0 , the drive moves toward the positive direction.
4. To pause the drive, set the variable V8000.2 or V8000.3 as 0.

The following illustration displays the movement trajectory and dynamic parameters:



In this illustration, "v" refers to velocity and "t" refers to time.

12.9.2.12 Incremental jog

The Incremental jog mode is implemented through the "Jog" function of SINAMICS V90 PN. It enables the position-controlled, distance-dependent traversing of axes using the integrated position controller of the SINAMICS V90 PN.

Requirements

- The mode is selected with "ModePos" = 8.
- The device is switched on via "EnableAxis".
- The axis is at standstill.
- The axis does **not** have to be referenced or adjusted.

Sequence

The specification of the jog velocity is performed via the V-assistant or the SINA_PARA_S instruction for the configuration of the operating mode in the SINAMICS V90. The SINAMICS V90 uses the acceleration and deceleration for the dynamic responses of the axis.

The velocity override also applies in this operating mode and is set through "OverV". For example, the velocity that takes effect is 500LU/min and the OverV is 120%, the velocity in SINAMICS V90 is 600LU/min.

The operating conditions "CancelTraversing" and "IntermediateStop" are not relevant for the operating mode and can be set to "1".

Note

"Jog1" and "Jog2" are the signal sources for the jog mode in SINA_POS. Jog 1 is the signal source for negative, and Jog 2 is the signal source for positive.

You can configure the distance of incremental traversing motion in the SINAMICS V90 PN.

The travel direction when jogging results from the set velocity setpoint. The inputs "Positive" and "Negative" are not relevant for the operating mode and can be set to "0" as standard.

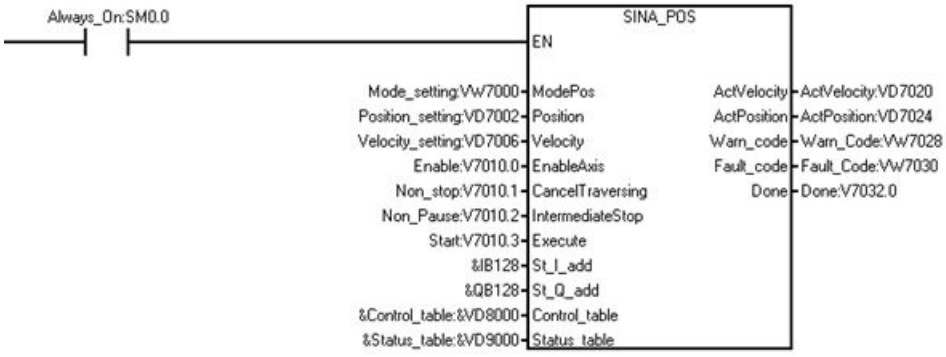
The current state of the active command can be tracked via "EPosZSW1 / EPosZSW2".

The block displays the current command processing with "AxisEnabled" and acknowledges the termination of the jog function ("Jog1" or "Jog2" = 0) when the axis is at standstill with bit "AxisPosOk". If an error occurs during the traversing motion, the bit "AxisError" in the output signal "Status_table" is issued.

Note

The current command can be replaced on-the-fly by a new command via "Jog1" or "Jog2". This is only possible when you remain in one of the jog modes.

Example of the Incremental jog mode



Set the operating mode as "Incremental Jog" as follows:

1. Create the following variables and write the variables to the corresponding input operands.

Symbol	Address	Comment	Corresponding input operand	Data type	Value		
Mode_setting	VW7000	Mode selection	ModePos	WORD	8		
Enable	V7010.0	Enable the drive.	EnableAxis	BOOL	1		
Non_stop	V7010.1	The status is non-stop.	CancelTraversing	BOOL	1		
Non_Pause	V7010.2	No pausing.	IntermediateStop	BOOL	1		
Control_table	VD8000	The control parameters.	Control_table	DWORD	V8000.2 [1]	Jog1	0
					V8000.3 [1]	Jog2	1
					VD8008	ConfigEpos	3

[1] The value of V8000.2 and V8000.3 cannot be 1 or 0 at the same time.

Note

The variable value in this example is for your reference, and you need to create the variable according to your actual situation.

2. Enter the data in the input "St_I_add" and "St_Q_add".

Note

For the four inputs "St_I_add", "St_Q_add", "Control_table", and "Status_table", the mode of addressing instruction operands is the indirect addressing. You must enter an ampersand (&) at the beginning of the input operand.

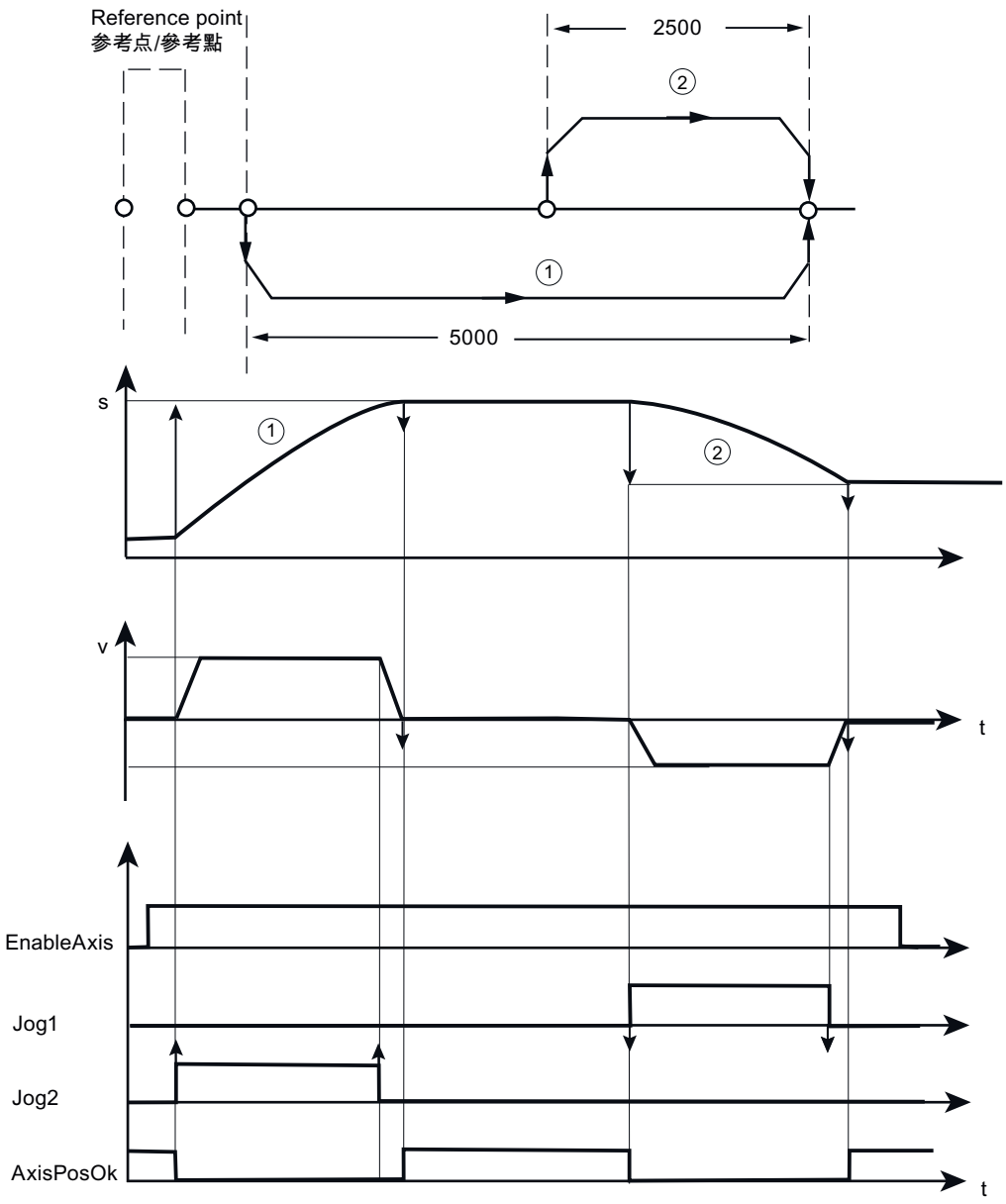
For the input operand "St_I_add" and "St_Q_add", keep the offset consistent with that in PROFINET wizard.

You have the following options for the variable:

- To make the drive rotate in the negative position, set V8000.2 as 1, V8000.3 as 0.
- To make the drive rotate in the positive position, set V8000.3 as 1, V8000.2 as 0.

Result: The drive runs at the incremental jog speed set in V-assistant. When the incremental distance is reached, the motor stops, no matter whether "Jog1" or "Jog2" is set as 1.

The following picture displays the movement trajectory and dynamic parameters:



In this illustration, "v" refers to velocity, "s" refers to position, and "t" refers to time.

12.9.2.13 Encoder adjustment (for telegram 112 only)

This function is only available when using a servo motor with the absolute encoder. It’s only needed to adjust once to the mechanics and then the adjustment can be retained even after the machine is switched off. For the absolute encoder adjustment, the machine axis must be moved to the desired position first, this can be done in jog mode or manually. A position marking, a fixed stop, or measuring the distance between the axis and the fixed home position can be used as a reference. The encoder adjustment is performed via the "encoder adjustment" drive function.

Note

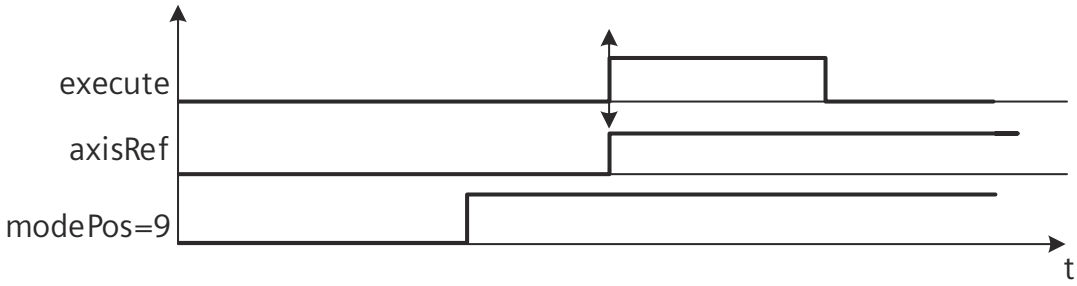
After the absolute encoder is adjusted, the offset should be saved into the drive.

Requirements

- The operating mode is selected with "modePos"=9.
- The axis can be in closed loop control but must be at a standstill.

Sequence

- Axis is at a standstill and the position is moved to the desired position, then set by means of a positive edge for "execute".
- If a fault occurs while setting the home position, the "error" output signal is output. The following figure shows how the encoder adjustment mode works.



12.9.3 SINA_SPEED instruction

12.9.3.1 Prerequisite of using the SINA_SPEED instruction

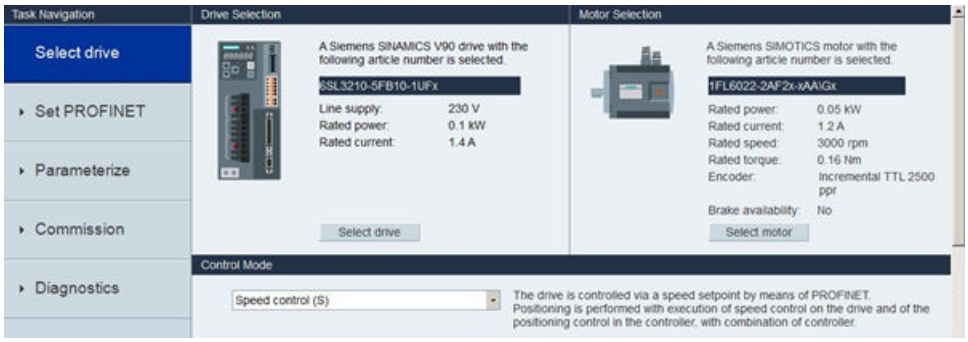
The prerequisite for using the Speed_control instruction is as follows:

- The SINAMICS V90 PN drive and the servo motor are ready.
- The PROFINET network is connected between the drive and the S7-200 Smart CPU.
- The software V-assistant is connected with V90 PN. For the detailed information, refer to SINAMICS V-ASSISTANT Online Help (https://support.industry.siemens.com/cs/ww/en/view/109738316). You can download the SINAMICS V-ASSISTANT software (https://support.industry.siemens.com/cs/ww/en/view/109738387) and the SINAMICS V90: PROFINET GSD file (https://support.industry.siemens.com/cs/ww/en/view/109737269) in the SIEMENS Industry Online Support Website.

Procedure of configuring SINAMICS V90 PN parameters with V-assistant

Configure the SINAMICS V90 PN parameters as follows:

1. Start the software V-assistant.
2. Click "Online" to select the working mode.
3. Click the connected drive and click the "OK" button.
4. Click "Select drive" from the navigation tree and select "Speed control (s)" in the "Control Mode" field.

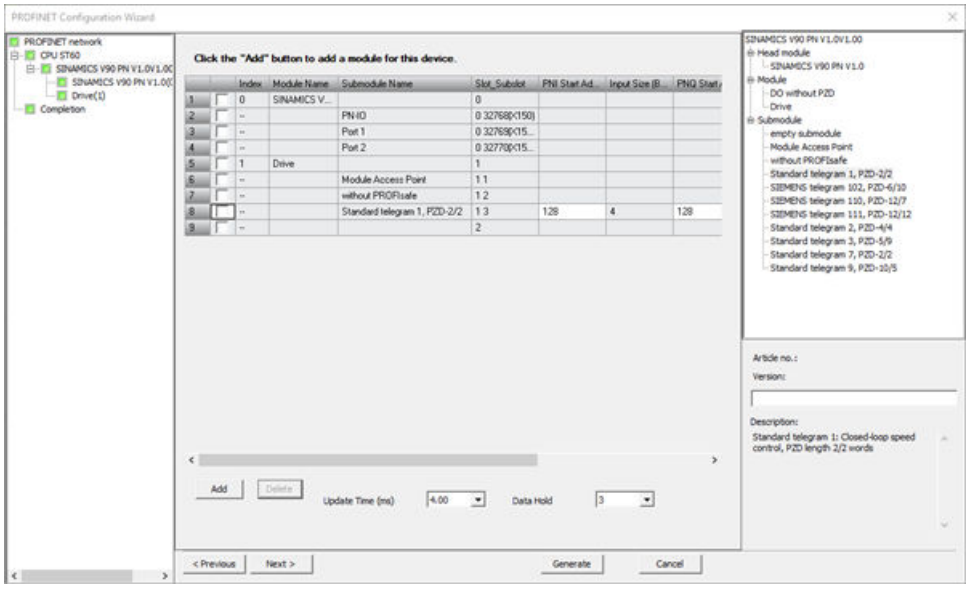


5. Click "Set PROFINET" from the navigation tree and click "Select telegram".

6. In the "Selection of telegram" field, select "Standard telegram 1" as the current telegram. The SINA_SPEED instruction only supports Standard telegram 1.

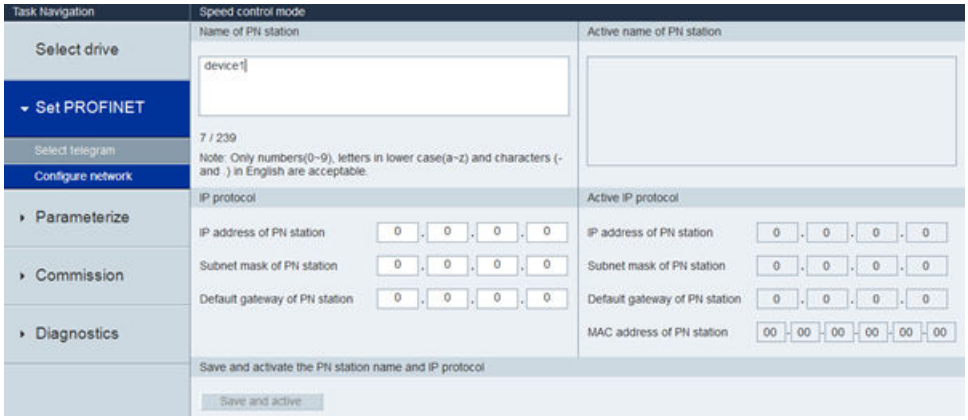
Note

When you configure the PROFINET network in PROFINET wizard, keep the telegram consistent with that in this step.



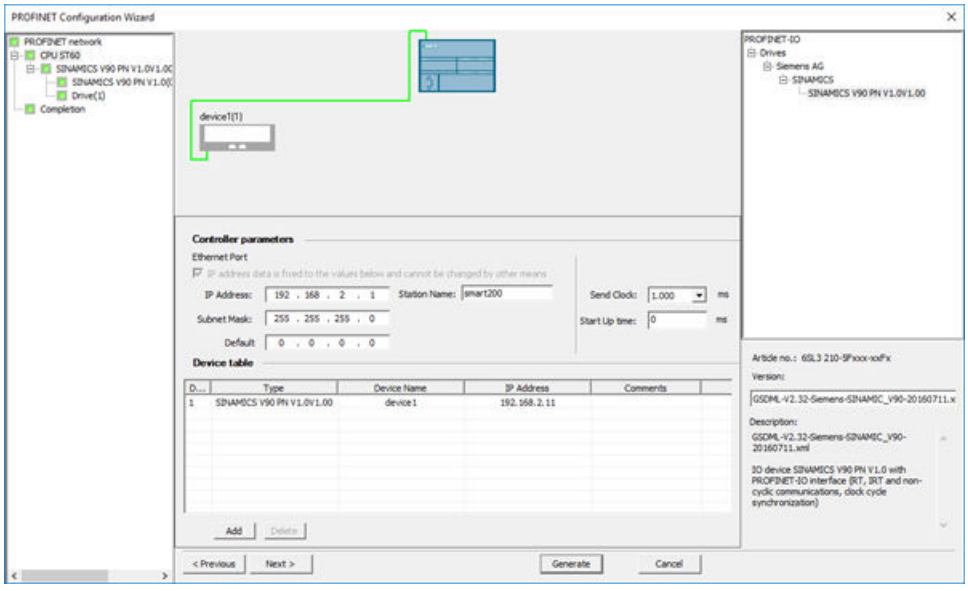
7. Click "Configure network" from the navigation tree.

8. Define the PN station name in the "Name of PN station".



Note

When you configure the PROFINET network in PROFINET wizard, keep the device name consistent with the PN station name in this step.



9. Click the "Save and active" button.

Result: The drive restarts and the configured parameters take effect.

12.9.3.2 Input and output interface of SINA_SPEED instruction

The SINA_SPEED instruction is used for setting the drive speed.

Table 12-56 SINA_SPEED instruction

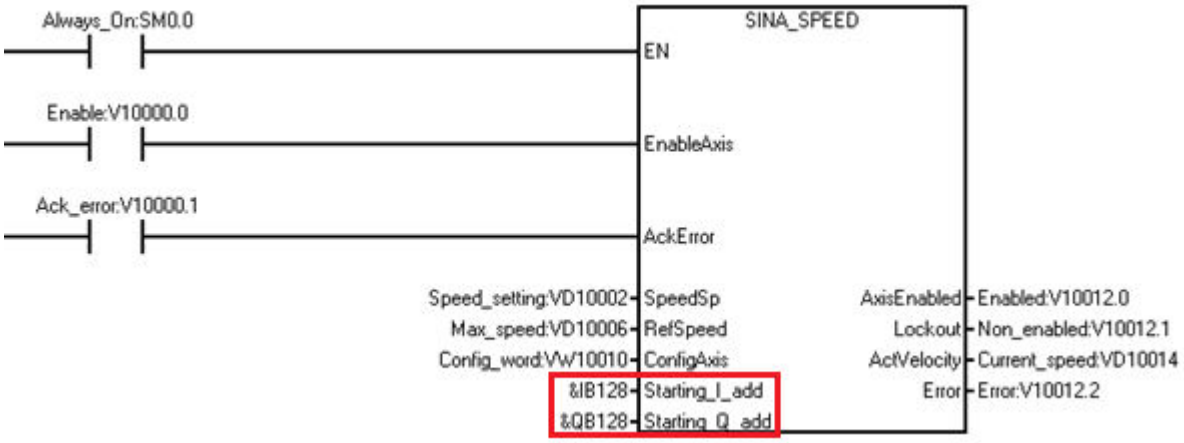
LAD/ FBD	STL	Description
SINA_SPEED EN EnableAxis AckError SpeedSp — AxisEnabled RefSpeed — Lockout ConfigAxis — ActVelocity Starting_I_add — Error Starting_Q_add	CALL SINA_SPEED,EnableAxis, AckError, SpeedSp, RefSpeed, ConfigAxis Starting_I_add,Starting_Q_add, AxisEnabled, Lockout, ActVelocity, Error	The SINA_SPEED instruction enables the speed control of drive movement.

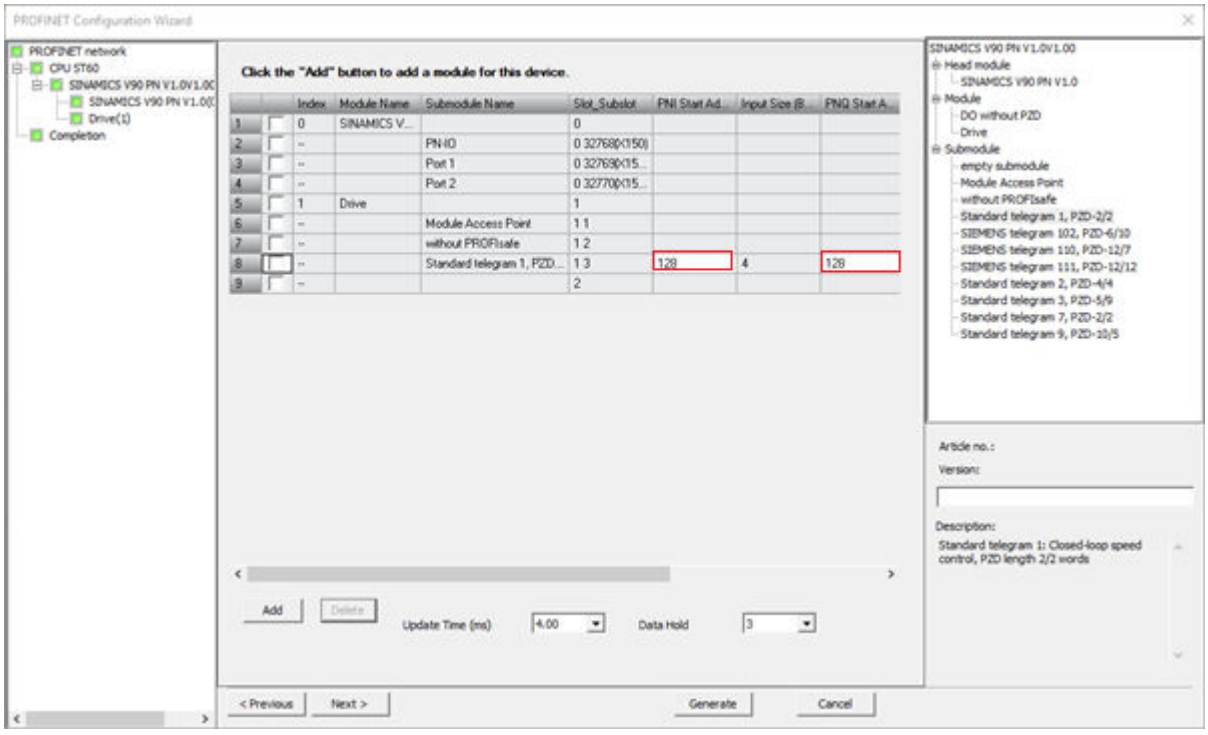
Note

For the two inputs "St_I_add" and "St_Q_add", the mode of addressing instruction operands is indirect addressing.

You must enter an ampersand (&) at the beginning of the input operand and keep the offset consistent with that in the PROFINET wizard.

You can take the following pictures for reference:





The parameters of SINA_SPEED instruction are as follows:

Table 12-57 Parameters of SINA_SPEED instruction

Parameter and type		Data type	Description
EnableAxis	IN	BOOL	"EnableAxis" = 1 -> switches on the drive
AckError	IN	BOOL	Acknowledges axis faults -> "AckFlt"=1
SpeedSp	IN	REAL	Speed setpoint. SpeedSp value changes with Refspeed. For example, if you set Refspeed as 1000 rpm, the SpeedSp value range is (0, 1000 rpm).
Refspeed	IN	REAL	Rated speed of the drive. The value range is (6, 210000 rpm).
ConfigAxis	IN	WORD	One input to control the SINA_SPEED functions that are not directly specified at the block. For the detailed information, refer to Definition of "ConfigAxis" parameters (Page 725). (Default value = 0)
Starting_I_add	IN	DWORD	Pointer of I memory area start address for PROFINET IO.
Starting_Q_add	IN	DWORD	Pointer of Q memory area start address for PROFINET IO.
AxisEnabled	OUT	BOOL	The drive is being executed or enabled.
Lockout	OUT	BOOL	1 = switching-on inhibited active.
ActVelocity	OUT	REAL	Actual velocity ->dependent on scaling factor RefSpeed.
Error	OUT	BOOL	1 = group fault active.

12.9.3.3 Definition of "ConfigAxis" parameters

The following table lists the definition of "ConfigAxis" parameters:

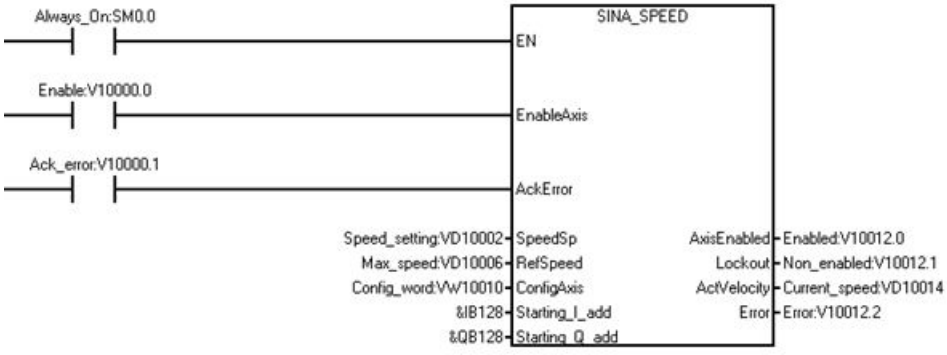
Table 12-58 ConfigAxis

ConfigAxis	Meaning
Bit0	OFF2
Bit1	OFF3
Bit2	Inverter enable
Bit3	Enable ramp-function generator
Bit4	Continue ramp-function generator
Bit5	Enable speed setpoint
Bit6	Direction of rotation
Bit7	Unconditionally open holding brake
Bit8	Motorized potentiometer increase setpoint
Bit9	Motorized potentiometer, decrease setpoint
Bit10	Reserved
Bit11	Reserved
Bit12	Reserved
Bit13	Reserved
Bit14	Reserved
Bit15	Reserved

12.9.3.4 Example of SINA_SPEED instruction

The SINA_SPEED instruction is used for controlling the drive cyclically with standard telegram 1.

Example



Insert and configure the SINA_SPEED instruction as follows:

1. Create the following variables and write the variables to the corresponding input operands.

Symbol	Address	Comment	Corresponding input operand	Data type	Value
Speed_setting	VD10002	Speed	SpeedSp	REAL	100
Max_speed	VD10006	Maximum speed	RefSpeed	REAL	3000
Config_word	VW10010	Parameter table of ConfigAxis	ConfigAxis	WORD	63 **Note:** To enable the drive, you must set this variable as 63 (Decimal).
Enable	V10000.0	Enable the drive.	EnableAxis	BOOL	1
Ack_error	V10000.1	Acknowledge the fault.	AckError	BOOL	
Enabled	V10012.0	The axis is enabled.	AxisEnabled	BOOL	
Non_enabled	V10012.1	The drive is not enabled.	Lockout	BOOL	
Current_speed	VD10014	Actual velocity	ActVelocity	REAL	
Error	V10012.2	Error from the drive	Error	REAL	

2. Enter the data in the input "St_I_add" and "St_Q_add".

Note

For the inputs "St_I_add" and "St_Q_add", the mode of addressing instruction operands is indirect addressing. You must enter an ampersand (&) at the beginning of the input operand.

For the input operands "St_I_add" and "St_Q_add", keep the offset consistent with that in the PROFINET wizard.

12.9.4 SINA_PARA_S instruction

12.9.4.1 Prerequisite of using the SINA_PARA_S instruction

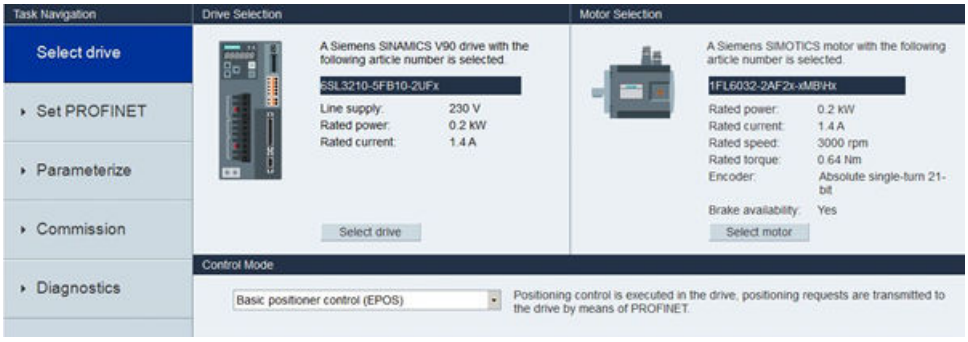
The prerequisite for using the SINA_PARA_S instruction is as follows:

- The SINAMICS V90 PN drive and the servo motor are ready.
- The PROFINET network is connected between the drive and the S7-200 Smart CPU.
- The software V-assistant is connected with SINAMICS V90 PN. For the detailed information, refer to SINAMICS V-ASSISTANT Online Help (https://support.industry.siemens.com/cs/ww/en/view/109738316). You can download the SINAMICS V-ASSISTANT software (https://support.industry.siemens.com/cs/ww/en/view/109738387) and the SINAMICS V90: PROFINET GSD file (https://support.industry.siemens.com/cs/ww/en/view/109737269) in the SIEMENS Industry Online Support Website.

Procedure of configuring SINAMICS V90 PN parameters with V-assistant

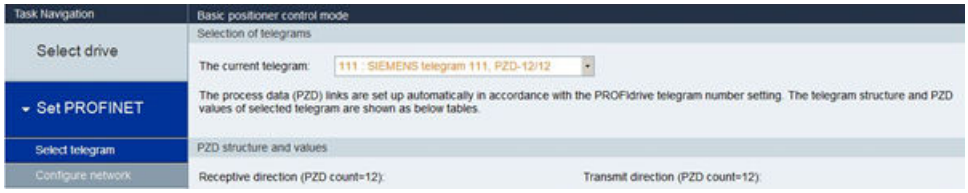
Configure the V90 PN parameters with V-assistant as follows:

1. Start the software V-assistant.
2. Click "Online" to select the working mode.
3. Click the connected drive and click the "OK" button.
4. Click "Select drive" from the navigation tree, and select "Speed control (s)" or "Basic positioner control (EPOS)" in the "Control Mode" field.



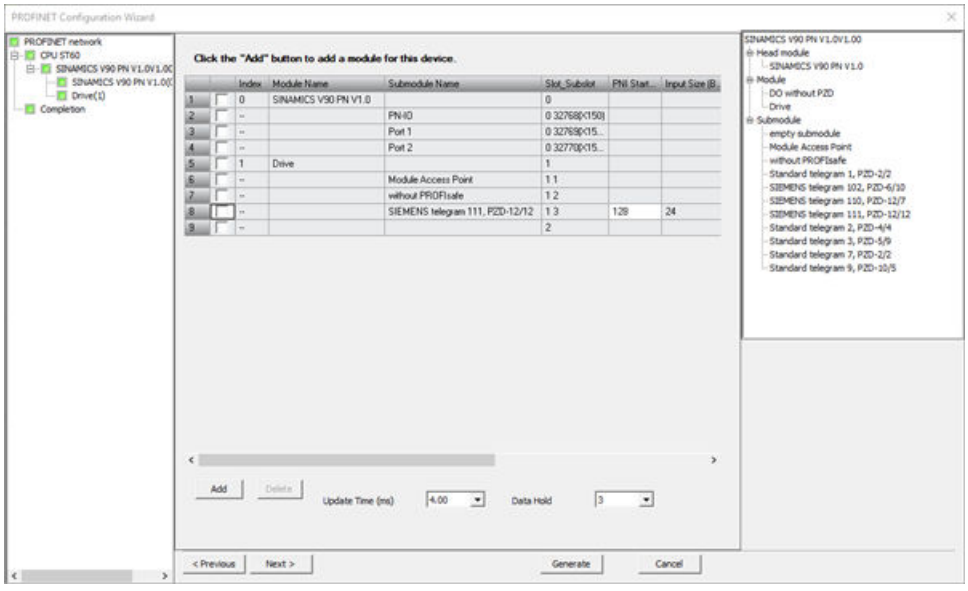
5. Click "Set PROFINET" from the navigation tree and click "Select telegram".

6. In the "Selection of telegram" field, select your desired telegram as the current telegram. The SINA_PARA_S instruction only supports SIEMENS telegram 111 or Standard telegram 1.



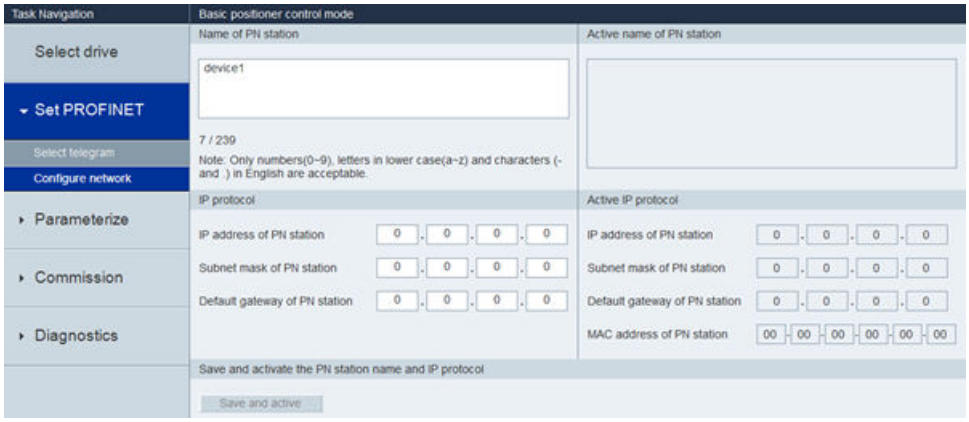
Note

When you configure the PROFINET network in PROFINET wizard, keep the telegram consistent with that in this step.



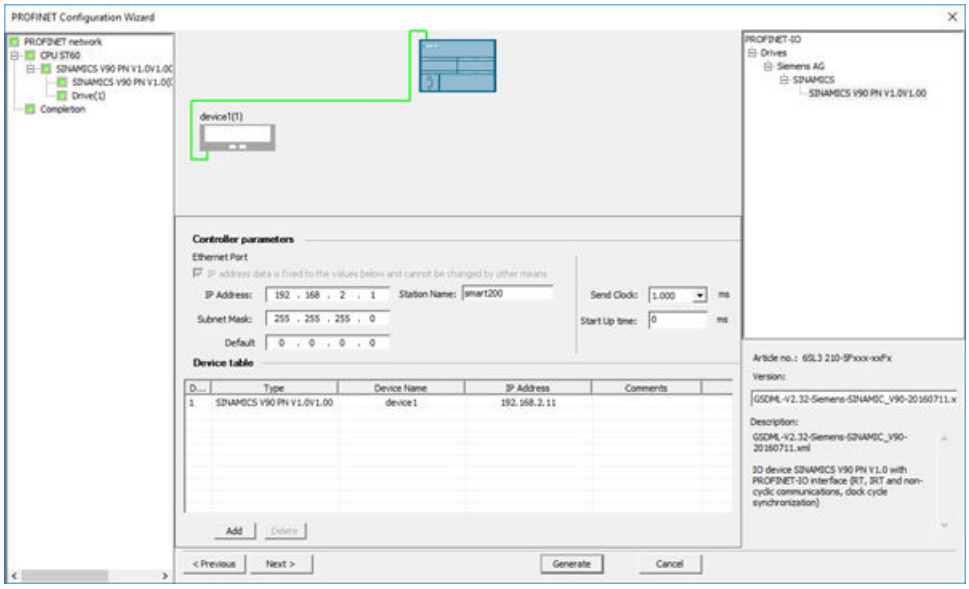
7. Click "Configure network" from the navigation tree.

8. Define the PN station name in the "Name of PN station".



Note

When you configure the PROFINET network in PROFINET wizard, keep the device name consistent with the PN station name that in this step.



9. Click the "Save and active" button.

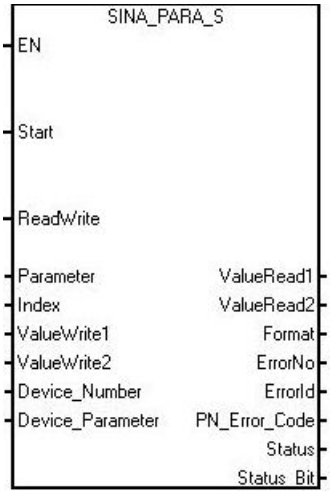
Result: The drive restarts and the configured parameters take effect.

12.9.4.2 Input and output interface of SINA_PARA_S instruction

The SINA_PARA_S instruction is used for managing the acyclic parameters.

The SINA_PARA_S instruction is as follows:

Table 12-59 SINA_PARA_S instruction

LAD/FBD	STL	Description
	CALL SINA_PARA_S,Start, ReadWrite, Parameter, Index, ValueWrite1, ValueWrite2, Device_Number, Device_Parameter, ValueRead1, ValueRead2, Format, ErrorNo, ErrorId, PN_Error_Code, Status, Status_Bit	Use the SINA_PARA_S instruction to set the drive parameters or to read the parameter from drive.

The following table lists the input and output signals of SINA_PARA_S insrtruction:

Table 12-60 Parameters of SINA_PARA_S instruction

Parameter and type		Data type	Description
Start	IN	BOOL	Start of the task (0= no task or cancel task; 1= start and execute task)
ReadWrite	IN	BOOL	Task type selection 0 = read, 1= write
Parameter	IN	INT	Parameter number
Index	IN	INT	Index of the parameter
ValueWrite1	IN	REAL	Value of the parameter in REAL format
ValueWrite2	IN	DINT	Value of the parameter in DINT format
DeviceNo	IN	WORD	Device number
Device_Parameter	IN	DWORD	Pointer of the start address of "Device_Parameter" (Page 731). "Device_Parameter" refers to the parameter of the PROFINET slave.
ValueRead1	OUT	REAL	Value of the parameter read from the drive (REAL format)
ValueRead2	OUT	DINT	Value of the parameter read from the drive (DINT format)
Format	OUT	BYTE	Format of the read parameter (Page 731)
ErrorNo	OUT	WORD	Error number according to PROFIdrive profile (Page 732)
ErrorID	OUT	DWORD	Error ID (Page 733). First word: binary-coded indicating which parameter access is faulted Second word: type of fault
PN_Error_Code	OUT	DINT	Error code according to the PROFINET protocol. For detailed information, refer to *Technical Specification for PROFINET IO (Version 2.3)*.
Status	OUT	BYTE	The status of the current operation (Page 733).
Status_bit	OUT	BYTE	Status table (Page 734)

Definition of "Device_parameter"

The following table list the bit definitions of Device_parameters:

Table 12-61 Device_parameters

Byte Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Axis number: Select 2 for the V90 drive. For other drives, refer to the related manual.							
1	Reserved							
2	API number[1]							
3								
4								
5								
6	Slot number[1]							
7								
8	Subslot number[1]							
9								

Note

[1] For the API number, Slot number and Subslot number, move to the PROFINET wizard (Page 505)to check the information.

Definition of "Format" parameters

The format of parameter read from the drive is as follows:

Table 12-62 Format

ID	Description
02	Integer 8
03	Integer 16
04	Integer 32
05	Unsigned 8
06	Unsigned 16
07	Unsigned 32
08	Floating Point
10	Octet String 8 (16 bit)
13	Time Difference (32 bit)
41	Byte
42	Word
43	Double word
44	Error

Definition of "ErrorNo" parameters

ErrorNo refers to the error code defined by the PROFIdrive protocol.

Table 12-63 ErrorNo

Error ID	Description
00 hex	Illegal parameter number (access to a parameter that does not exist)
01 hex	Parameter value cannot be changed (change request for a parameter value that cannot be changed)
02 hex	Lower or upper value limit exceeded (change request with a value outside the value limits)
03 hex	Incorrect subindex (access to a parameter index that does not exist)
04 hex	No array (access with a subindex to non-indexed parameters)
05 hex	Incorrect data type (change request with a value that does not match the data type of the parameter)
06 hex	Setting not permitted, only resetting (change request with a value not equal to 0 without permission)
07 hex	Descriptive element cannot be changed (change request to a descriptive element that cannot be changed)
09 hex	Description data not available (access to a description that does not exist, parameter value is available)
0B hex	No master control (change request but with no master control)
0F hex	Text array does not exist (although the parameter value is available, the request is made to a text array that does not exist)
11 hex	Request cannot be executed due to the operating state (access is not possible for temporary reasons that are not specified)
14 hex	Inadmissible value (change request with a value that is within the limits but which is illegal for other permanent reasons, for example, a parameter with defined individual values)
15 hex	Response too long (the length of the actual response exceeds the maximum transfer length)
16 hex	Illegal parameter address (illegal or unsupported value for attribute, number of elements, parameter number, subindex or a combination of these)
17 hex	Illegal format (change request for an illegal or unsupported format)
18 hex	Number of values not consistent (number of values of the parameter data to not match the number of elements in the parameter address)
19 hex	Drive object does not exist (access to a drive object that does not exist)
20 hex	Parameter text cannot be changed
21 hex	Service is not supported (illegal or not support request ID)
6B hex	A change request for a controller that has been enabled is not possible. (The drive rejects the change request because the motor is switched on. Please observe the "Can be changed" parameter attribute (U, T) as given in the parameter list relevant section in the SINAMICS V90, SIMOTICS S-1FL6 Operating Instructions
6C hex	Unknown unit.
77 hex	Change request is not possible during download
81 hex	Change request is not possible during download
82 hex	Accepting the master control is inhibited
83 hex	Desired interconnection is not possible (the connector output does not supply a float value although the connector input requires a float value)
84 hex	Inverter does not accept a change request (inverter is busy with internal calculations)
85 hex	No access methods defined
87 hex	Know-how protection active, access locked
C8 hex	Change request below the currently valid limit (change request to a value that lies within the "absolute" limits, but is however below the currently valid lower limit)

Error ID	Description
C9 hex	Change request above the currently valid limit (example: a parameter value is too large for the drive power)
CC hex	Change request not permitted (change is not permitted as the access code is not available)

Definition of "ErrorID" parameters

There are two types of error:

- ErrorID[1]
- ErrorID[2]

Table 12-64 ErrorID[1]

ID	Description
0×000	No fault active
0×001	Internal telegram error active
0×002	Parameterization error active
0×003	RDREC and WRREC call error active
0×004	Cancelation of the job during the active data transfer by resetting the Start input to "0"
0×005	Unknown data type detected; evaluation of the ErrorID[2] shows the parameter with unknown data type in the highest value bit

Table 12-65 ErrorID[2]

ID	Description
0×01	Unknown error

Definition of "STATUS" parameter

The parameters of STATUS are as follows:

Table 12-66 STATUS

Byte Offset	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	A [1]	E [2]	Error code [3]				

[1] A: 1 = a request is in process

[2] E: 1= an error occurs

[3] Error code: The system error code. For detailed information, refer to System-defined error code of the instructions RDREC and WRREC (Page 472).

Definition of "Status_bit" parameters

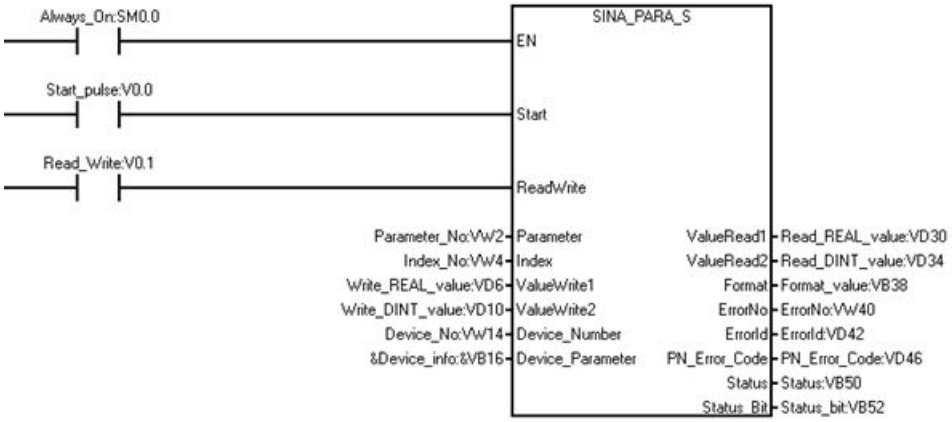
Table 12-67 Status_bit

Byte Offset	Bit 3	Bit 2	Bit 1	Bit 0
0	Error	Done	Busy	Ready

12.9.4.3 Example of the SINA_PARA_S instruction

The SINA_PARA_S instruction can read and write the drive parameters in acyclic communication.

Example



Modify the drive parameters with SINA_PARA_S instruction as follows:

1. Create the following variables and write the variables to the corresponding input operands.

Symbol	Address	Corresponding input operand	Data type	Value		
Start_pulse	V0.0	Start	BOOL	1		
Read_Write	V0.1	ReadWrite	BOOL	0 or 1		
Parameter_No	VW2	Parameter	INT	29070		
Index_No	VW4	Index	INT	1		
Write_REAL_value	VD6	ValueWrite1	REAL			
Write_DINT_value	VD10	ValueWrite2	DINT			
Device_No	VW14	Device_number	WORD	1		
Device_info	VB16	Device_Parameter	DWORD	VB16	Axis number	2
				VD18	API number	14848
				VW22	Slot number	1
				VW24	Subslot number	3
Read_REAL_value	VD30	ValueRead1	REAL			
Read_DINT_value	VD34	ValueRead2	DINT			
Format_value	VB38	Format	BYTE			
ErrorNo	VW40	ErrorNo	WORD			
ErrorId	VD42	ErrorID	DWORD			
PN_Error_Code	VD46	PN_Error_Code	DINT			
Status	VB50	Status	BYTE			
Status_bit	VB52	Status bit	BYTE	V52.0	Ready	
				V52.1	Busy	
				V52.2	Done	
				V52.3	Error	

2. Read the parameters from the drive to get the parameter data type information.
   - Set the variable "Read_Write" to 0 to initiate a task of reading drive parameters.
   - Enter the device parameter information in the variable "Device_info".
   - Enter the axis number in VB16 "Axis number".
   - Enter the parameter number in the variable "Parameter_No". Enter the index in the variable "Index_No".
   - Set the variable "Start_pulse" to 1 to start the task.

   Result:
   If the parameter data type is REAL, the variable "Read_REAL_value" displays the value.

If the parameter data type is DINT, the variable "Read_DINT_value" displays the value.

3. Set the variable "Read_Write" to 1 to initiate a task of writing the parameters to the drive.
4. Modify the parameter in the variable "Write_REAL_value" or "Write_DINT_value":
   If in step 2, the variable "Format_value" shows the following data: 16#02, 16#05, 16#41, 16#42, 16#03, 16#06, 16#0A or 16#08, you modify the parameter in the varaible "Write_REAL_value" .
   If in step 2, the variable "Format_value" shows the following data: 16#43, 16#04, 16#07 or 16#0D, you modify the parameter in the varaible "Write_DINT_value" .
5. Set the variable "Start_pulse" to 1.

Result:

In the software V-assistant, check whether you modify the parameter successfully.

Debugging and troubleshooting 13

STEP 7-Micro/WIN SMART V3 provides software tools to help you debug and test your program. These features include viewing the status of the program as it is executed by the CPU, selecting to run the CPU for a specified number of scans, and forcing values.

Use the hardware troubleshooting guide (Page 748) as a guide for determining the cause and possible solution when troubleshooting problems with the hardware.

13.1 Debugging your program

13.1.1 Bookmarks

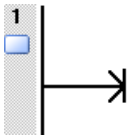
You can set bookmarks in your program to make it easy to move back and forth between designated networks in a long program:

Toggle Bookmark: Click this button to set or remove a bookmark at the program network designated by the current cursor location.

Next Bookmark: Click this button to move to the next bookmarked network of your program.

Previous Bookmark: Click this button to move to the previous bookmarked network of your program.

Remove All Bookmarks: Click this button to remove all bookmarks in your program.



13.1.2 Cross reference table

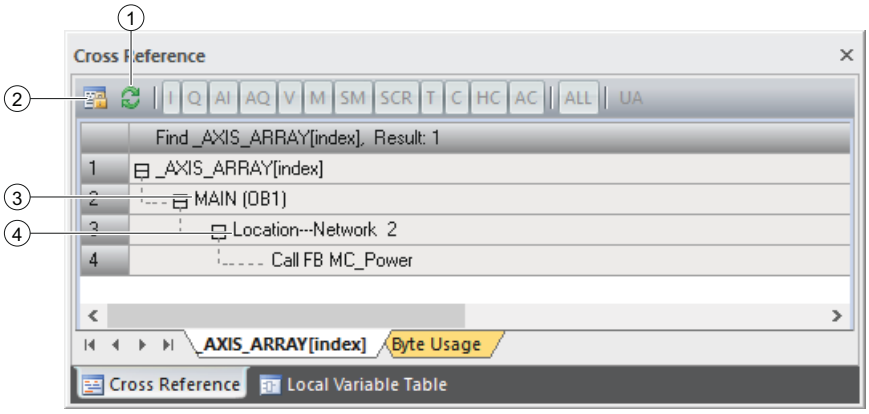
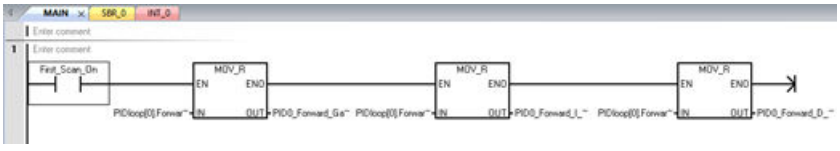
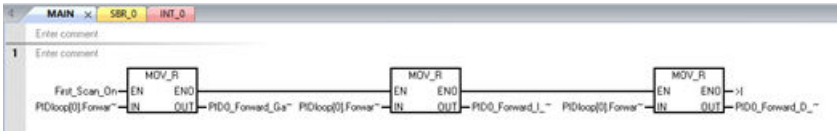
Use the cross reference table when you want to know whether a variable name or memory assignment is already in use in your program, and where it is used. The cross reference table identifies all operands used in the program, and identifies the POU, network or line location, and instruction context of the operand each time it is used. Double-clicking an element in the cross-reference table displays that part of your POU.

Checking the cross reference of an Element

1. Open the cross reference table
2. Select the operands you want to query in your program. You can use the hold button to hold the element you currently check.

Examples

The following examples show the cross-reference table for a simple program in all three languages: LAD, FBD, and STL.

Language	Program
Cross reference	 ① Refresh ② Hold the refrence result ③ The POU where the operand is used ④ The line or network where the operand is used
LAD	
FBD	
STL	

13.2 Displaying program status

13.2.1 Displaying status in the program editor

To display current data values and I/O status in the program editor, click the Program Status ON/OFF button from either the program editor toolbar or from the Debug menu ribbon strip Program Status.

Status data collection begins and shows the results of all logic operations during the execution of the program. You can also pause and resume program status collection with the Pause Status ON/OFF button from either the program editor toolbar or from the Debug menu ribbon strip Pause Status.

A status chart (Page 742) displays values at the end of scan.

Execution status coloring

- The power rail (LAD) is colored when the program is being scanned.
- Power flow or logic flow in the diagrams is indicated by coloring.
- Contacts and coils (LAD) that have power or are logically true are colored blue.
  You can assign your own color choice from the Options settings of the Tool menu ribbon strip and selecting the Colors tab.
- Box instructions – The box instructions are colored when the instruction has power and the instruction successfully executes without an error.
- Green timers and counters indicate that the Timer or Counter has valid data.
- Red indicates an instruction executed with an error.
- Jump and Label instructions display in the powerflow color when active. If not active, they display in gray.
- Gray color (default assignment) indicates no power flow - the instruction not scanned (jumped over or not called), or the PLC is in STOP mode.
- Blue color for Boolean Powerflow bits (FBD only).
- LAD, FBD, and STL program editors display the value of operands and indicate powerflow as each instruction executes during the execute program phase of a scan cycle. Execution status can display intermediate data values that may be overwritten by executing subsequent program instructions. All displayed PLC data values are collected from a single program scan cycle.

Example program status in an STL program

When you start program status in STL, the program editor window is divided into a code region and a status region. You can customize the status region according to the types of values that you want to monitor.

There are three categories of values that you can monitor in STL Status:

Operands	You can monitor up to 64 operands per instruction.
Logic stack	You can monitor up to the four most recent values from the logic stack.
Instruction status bits	You can monitor up to eleven status bits.

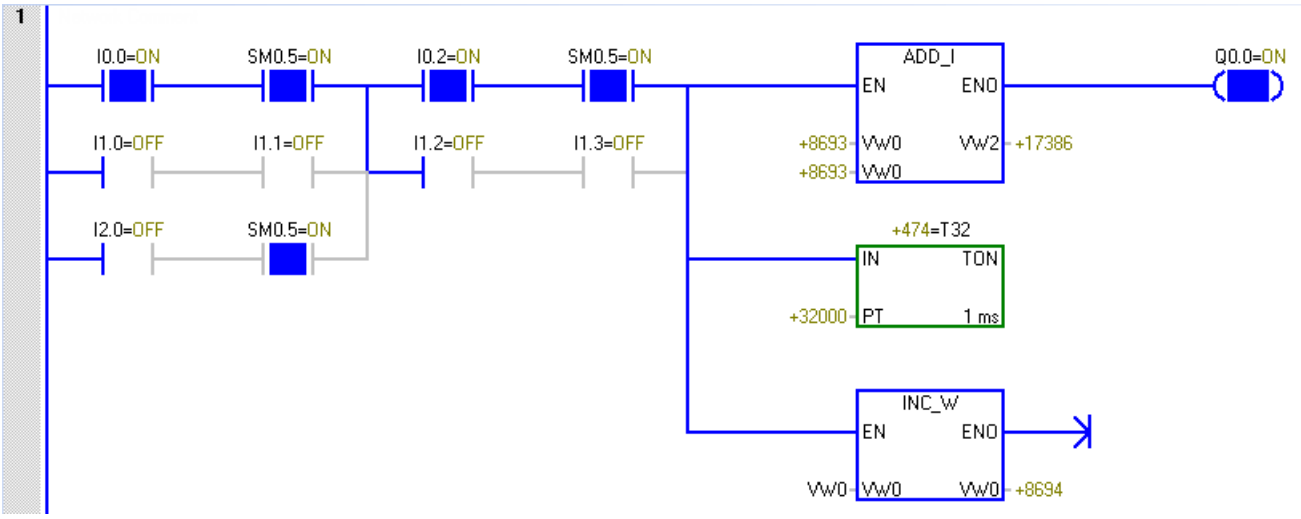
The STL Status tab of the Program Editor options (Page 742) in the Options settings of the Tools menu ribbon strip allows you to select or deselect any of these categories of values. If you deselect an item, it does not appear in the status display.

1

		Op 1	Op 2	Op 3	0123	
LD	I0.0	ON			1000	1
A	SM0.5	ON			1000	1
LD	I1.0	OFF			0100	0
A	I1.1	OFF			0100	0
OLD					1000	1
LD	I2.0	OFF			0100	0
A	SM0.5	ON			0100	1
OLD					1000	1
LD	I0.2	ON			1100	1
A	SM0.5	ON			1100	1
LD	I1.2	OFF			0110	0
A	I1.3	OFF			0110	0
OLD					1100	1
ALD					1000	1
LPS					1100	1
MOVW	VW0, VW2	+8525	+8525		1100	1
AENO					1100	1
+I	VW0, VW2	+8525	+17050		1100	1
AENO					1100	1
=	Q0.0	ON			1100	1
LRD					1100	1
TON	T32, +32000	+294	+32000		1100	1
LRD					1100	1
INCW	VW0	+8526			1100	1
LRD					1100	1
INCW	VW1000	+8526			1100	1
					–	–

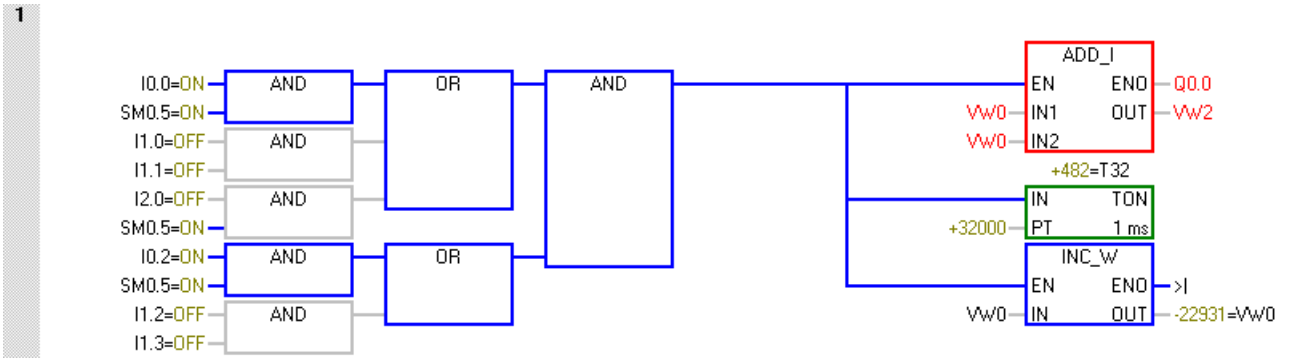
Example program status in the LAD program editor

The example below shows status in the LAD program editor. The program editor displays the values of operands and indicates powerflow as each instruction executes during the execute program phase of a scan cycle.



Example program status in the FBD program editor

The example below shows status in the FBD program editor. The red color of the ADD_I instruction box indicates errors in the operands.



13.2.2 Configuring the STL status options

To configure the STL program status display options, follow these steps:

1. Click the Options button from the Settings area of the Tools menu ribbon strip.

2. Under Options, click Program Editor > STL > Status.
3. Configure the following STL program status options:

Type	Select the font type for STL program status text.
Style	Select Regular, Italic, Bold, or Bold Italic for the text style.
Size	Select the point size for the font.
Watch Values	The check boxes and selection boxes allow you to include or remove operands, stack values, and instruction status bits (that is, flags) from the Program Status display.
Number of operands	If you chose to include operands in the Program Status display, you can edit the Operands list box to display more or fewer operands. The maximum number possible is 64.
Number of Stack Bits	If you chose to include logic stack values in the Program Status display, you can edit the Logic Stack list box to display more or fewer stack values. The maximum number possible is four.
Instruction Status Bits	If you chose to include instruction status bits in the program status display, select the Instruction Status Bits that you want to be shown and which (if any) should be omitted. A check mark indicates that you are choosing to watch a particular status bit in the program status display; if you deselect the checkbox, STEP 7-Micro/WIN SMART does not display that status bit in the Program Status.

See also

How to display status in the program editor (Page 739)

13.3 Using a status chart to monitor your program

In a status chart, you can enter addresses or defined variables to monitor or modify the status of program inputs, outputs, or variables by displaying the current values. The status chart also allows you to force or change the values of the process variables. You can create multiple status charts in order to view elements from different portions of your program.

You can display timer and counter values as either bits or words. If you display a timer or counter value as a bit, you see the output status of the instruction (0 or 1). If you display a timer or counter value as a word, you see the current value of the timer or counter.

Creating a new chart

To create a new status chart, make sure that Chart Status and Program Status is off and use one of these methods to create a new chart:

- From the project tree, right-click the Status Chart folder and select the context menu command **Insert > Chart**.
- From the Insert area of the Edit menu ribbon strip, click the down arrow beneath "Object" and select "Chart" from the drop-down menu.
- From a status chart tab in the status chart editor, or from any cell in an existing status chart, right-click and select the context menu command **Insert > Chart**.
- From the status chart toolbar, click the insert button.

After you successfully insert a new status chart, the new chart appears under "Status Chart" in the project tree and a new tab appears at the bottom of the Status Chart window.

Opening an existing chart

If the status chart editor is not open, you can open an existing status chart from the project tree, navigation bar, or from the Component drop-down list in the Windows area of the View menu. If the status chart editor is open, you can click a status chart tab in the editor to switch to that status chart.

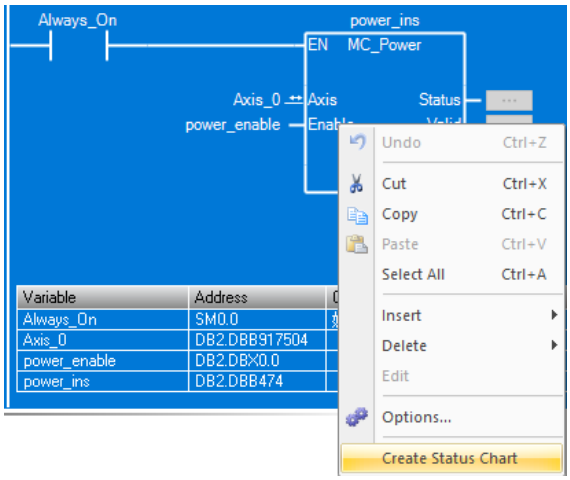
Building a status chart

To build a status chart, follow these steps:

1. Enter the address (or variable name) for each desired value in the Address field. Variable names must be names that you have already defined in the global variable table.
2. If the element is a bit (I, Q, or M, for example), the format is set as bit in the Format column. If the element is a byte, word, or double word, select the drop-down list in the Format column and select the valid format from the available options.
3. To insert an additional row, use one of the following methods:
   - From the Insert area of the Edit menu ribbon strip, click the "Row" button.
   - Right-click a cell in the status chart to bring up a context menu, and select the menu command **Insert > Row**.
     The new row is inserted above the current location of the cursor in the status chart. You can also place the cursor in the last cell of the last row and press the DOWN ARROW key to insert a row at the bottom of the status chart.

Building a status chart from a section of program code

Highlight a selection of networks in the program editor, right-click, and select "**Create Status Chart"** from the context menu. The new chart contains an entry for each unique operand in the selected region for which status can be gathered. STEP 7-Micro/WIN SMART places the entries in the order of their occurrence in the program, gives the chart a default name, and adds this chart after the last tab in the status chart editor.



When creating a chart from the program editor, note that only the first 4096 addresses can be added each time you select **"Create Status Chart"**. After STEP 7-Micro/WIN SMART creates the status chart, you can edit the chart entries.

You can also add an entry to a status chart by pressing the Ctrl key and dragging an operand from the LAD or FBD program editor to the status chart. From STL, you can select an address and drag it to a status chart.

Note

A project can store a maximum of 32 status charts.

Copying global variables from the global variable table to a status chart

You can copy addresses or global variable names from the global variable table and paste them into the status chart to build your chart more quickly.

13.4 Forcing specific values

The rule of forcing specific values is as follows:

- The CPU allows you to force any or all of the I/O points (I and Q bits).
- You can force up to 16 memory values (V or M) or analog I/O values (AI or AQ).
- You can force up to 100 bytes for PROFINET I/O values.

- V memory, M memory or PROFINET I/O values can be forced in bytes, words, or double words.
- Analog values are forced as words only, on even-numbered byte boundaries, such as AIW6 or AQW14. All forced values are stored in the non-volatile memory of the CPU.

Note

You cannot transfer force information for PROFINET ports with a SD card.

Because the forced data might be changed during the scan cycle (either by the program, by the I/O update cycle, or by the communication-processing cycle), the CPU reapplies the forced values at various times in the scan cycle.

- *Reading the inputs:* The CPU applies the forced values to the inputs as they are read.
- *Executing the control logic in the program:* The CPU applies the forced values to all immediate I/O accesses. Forced values are applied for up to 16 memory values after the program has been executed.
- *Processing any communications requests:* The CPU applies the forced values to all read/write communications accesses.
- *Writing to the outputs:* The CPU applies the forced values to the outputs as they are written.

  Note

  The Force function overrides a Read Immediate or Write Immediate instruction. The Force function also overrides the STOP mode values that you configured in the system block (Page 181). If the CPU goes to STOP mode, the output reflects the forced value and not the STOP mode value that you configured for the output in the system block.

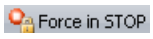
You can use the status chart to force values.

1. To force a new value, enter the value in the New Value column of the Status Chart, then click the Force button on the status chart toolbar, or right-click in the New Value column and select "Force" from the context menu.
2. To force an existing value, select the value in the Current Value column and click the Force button on the status chart toolbar, or right-click the value in the Current Value column and select "Force" from the context menu.

The status LEDs (Page 136) indicate whether the CPU has any forced data.

13.5 Writing and forcing outputs in STOP mode

To enable Write and Force functions while in STOP mode, click the "Force in Stop" button from the Settings area of the Debug menu ribbon strip.



The S7-200 SMART PLCs support writing and forcing outputs (both analog and digital) while the PLC is in STOP mode. However, as a safety precaution, you must specifically enable this functionality in STEP 7-Micro/WIN SMART V3 with the "Force in Stop" setting.

WARNING

Effect on process equipment of writing or forcing outputs

If you have connected the S7-200 SMART PLC to process equipment when you write or force an output, the PLC can transmit these changes to the equipment. This could result in unanticipated activity in the equipment, which could cause death or serious injury to personnel, and/ or property damage.

Only write or force outputs when your process equipment can safely accept those changes.

By default, STEP 7-Micro/WIN SMART V3 does not enable "Force in STOP". STEP 7-Micro/WIN SMART V3 prevents you from writing or forcing outputs while the PLC is in STOP mode. Clicking the "Force in STOP" button from the Debug menu enables writing and forcing for the current editing session with the current project. When you open a different project, "Force in STOP" returns to its default state and STEP 7-Micro/WIN SMART V3 prevents you from either writing or forcing output addresses while the PLC is in STOP mode.

The status LEDs (Page 136) indicate whether the CPU has any forced data in STOP mode.

13.6 How to execute a limited number of scans

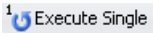
You can specify that the PLC execute your program for a limited number of scans (from 1 scan to 65,535 scans). By selecting the number of scans for the PLC to run, you can monitor the program as it changes the process variables.

On the first scan, the value of SM0.1 is one (ON).

Before executing a single scan or multiple scans, change the PLC to STOP mode (Page 82) if the PLC is not already in STOP mode.

Executing a single scan

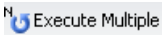
To execute a single scan, click the "Execute Single" button from the Scan area of the Debug menu ribbon strip.



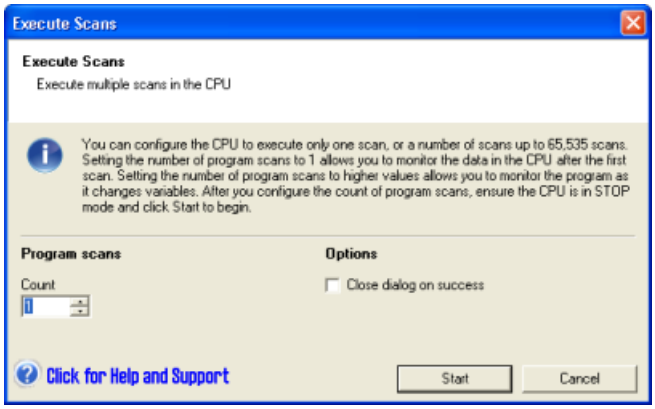
Executing multiple scans

To execute multiple scans, follow these steps:

1. Click the "Execute Multiple" button from the Scan area of the Debug menu ribbon strip.

   The Execute Scans dialog box appears.

2. Enter a value for the desired number of scans, and click "Start" to execute the number of scans you entered.

   Note

   When you are ready to resume normal program operation, change the PLC back to RUN mode (Page 82).

See also

Overview of debugging and monitoring features (Page 737)

How to display status in the editor windows (Page 739)

How to display status in a status chart (Page 742)

How to download a program (Page 124)

Timestamp mismatch error (Page 1061) (ensuring that project in programming device matches project in PLC)

Cross reference and element usage (Page 737) (ensuring that program edits do not cause duplicate assignments)

Forcing values (Page 744)

Forcing outputs in STOP mode (Page 745)

13.7 Hardware troubleshooting guide

Table 13-1 Troubleshooting guide for the S7-200 SMART hardware

Symptom	Possible cause	Possible solution
Outputs stop working	The device being controlled has caused an electrical surge that damaged the output	When connecting to an inductive load (such as a motor or relay), a proper suppression circuit should be used. Refer to the wiring guidelines in Chapter 3.
	Wiring loose or incorrect	Check wiring and correct
	Excessive load	Check load against contact ratings
	Output point is forced	Check the CPU for forced I/O
ERROR light on the CPU turns on (Red)	Electrical noise	Refer to the wiring guidelines in Chapter 3. It is very important that the control panel is connected to a good ground and that high voltage wiring is not run in parallel with low voltage wiring. Connect the M terminal on the 24 V DC Sensor Power Supply to ground
	Component damage	Send in hardware for repair or replacement
None of the CPU LEDs turn on	Blown fuse	Use a line analyzer and monitor the input power to check the magnitude and duration of the over-voltage spikes. Based on this information, add the proper type surge arrestor device to your power wiring.
	Reversed 24 V power wires	Refer to the wiring guidelines in Chapter 3 for information about installing the field wiring.
	Incorrect voltage	
Intermittent operation associated with high energy devices	Improper grounding	Refer to the wiring guidelines in Chapter 3.
	Routing of wiring within the control cabinet	It is very important that the control panel is connected to a good ground and that high voltage wiring is not run in parallel with low voltage wiring. Connect the M terminal on the 24 V DC Sensor Power Supply to ground.
	Time delay on input filters too short	Increase the input filter delay in the system data block.

Symptom	Possible cause	Possible solution
Serial communications (RS-485 or RS-232) are damaged when connecting to an external device. Either the port on the external device or the port on the CPU is damaged.	The communications cable can provide a path for unwanted currents if all non-isolated devices, such as PLCs, computers, or other devices do not share the same network circuit common reference. The unwanted currents can cause communications errors or damage to electric circuits.	• Refer to the wiring guidelines in Chapter 3 and to the network guidelines in Chapter 8. • Purchase network isolators or isolated RS485-to-RS485 repeaters when you connect devices that do not have a common electrical reference. Refer to Appendix F for information about article numbers for S7-200 SMART equipment.
Other communications problems (STEP 7-Micro/WIN SMART V3)		Refer to Communication (Page 475) for information about network communications.
Error handling		Refer to Appendix C for information about error codes.

PID loops and tuning

14

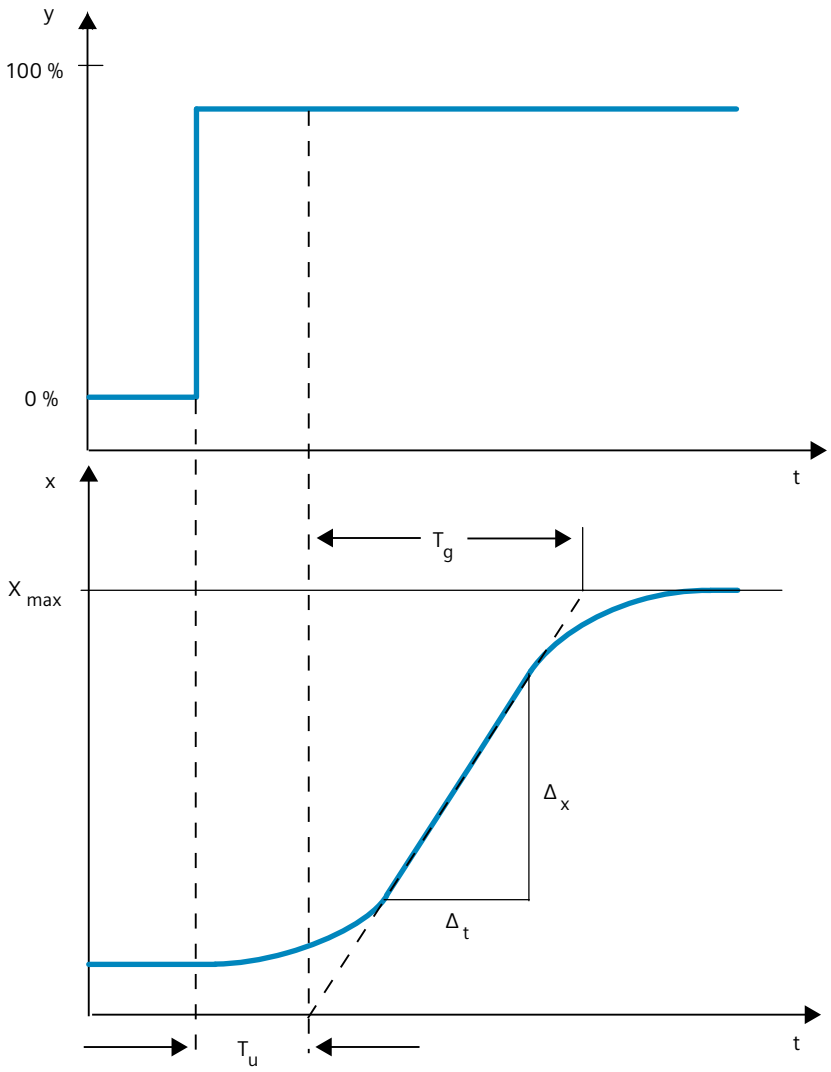
PID auto-tune capability is incorporated into the CPU, and STEP 7-Micro/WIN SMART V3 adds a PID Tune control panel. Together, these two features greatly enhance the utility and ease-of-use of the PID function provided.

You can initiate auto-tune in the user program, from an operator panel, or by the PID Tune control panel. The PID Auto-Tuner computes suggested (near optimum) values for the gain, integral time (reset), and derivative time (rate) tuning values. You can select tuning for fast, medium, slow, or very slow response of your loop.

With the PID Tune control panel, you can initiate the auto-tuning process, abort the auto-tuning process, and monitor the results in a graphical form. The control panel displays any error conditions or warnings that might be generated. You can apply the gain, reset, and rate values computed by auto-tune.

14.1 Pretuning

The pretuning determines the process response to a jump change of the output value. The tuned PID parameters are calculated as a function of the maximum slope and dead time of the controlled system.



The time response can be determined by approximation using the variables Delay time Tu, Recovery time Tg and Maximum value Xmax. The variables are determined by applying tangents to the maximum value and the inflection point of the step response. In many situations, it is not possible to record the response characteristic up to the maximum value because the process value cannot exceed the specific values. In this case, the rate of rise vmax is used to identify the controlled system (vmax = Δx/Δt).

The more stable the process value is, the easier it is to calculate the PID parameters and the more precise the result will be. So it is recommended that you enable pretuning in "inactive" mode. And check the "inactive PID" in the parameter setting. Noise on the process value can be tolerated as long as the rate of rise of the process value is significantly higher compared to the noise.

STEP 7-Micro/WIN SMART offers different pretuning types depending on the configuration:

Note: It is recommend pretuning when PIDx_CTRL is in the "Inactive" modes.

- Pretuning forward/heating
- Pretuning reverse/cooling
- Pretuning forward/heating and reverse/cooling

If you want to tune the PID parameters for forward/heating and reverse/cooling, you can expect a better control response with "Pretuning forward/heating" followed by "Pretuning reverse/cooling" rather than with "Pretuning forward/heating and reverse/cooling". However, carrying out pretuning in two steps takes more time.

Requirements for unidirection pretuning

Forward/Heating

- | setpoint - process value | > 0.3 × |upper process value - lower process value |
- | setpoint - process value | > 0.5 × setpoint
- setpoint > process value

Reverse/Cooling

- | setpoint - process value | > 0.3 × |upper process value - lower process value |
- | setpoint - process value | > 0.5 × setpoint
- setpoint < process value

Requirements for bidirection pretuning

Forward/Heating

- | setpoint - process value | > 0.3 × |upper process value - lower process value |
- | setpoint - process value | > 0.5 × setpoint
- setpoint > process value

Reverse/Cooling

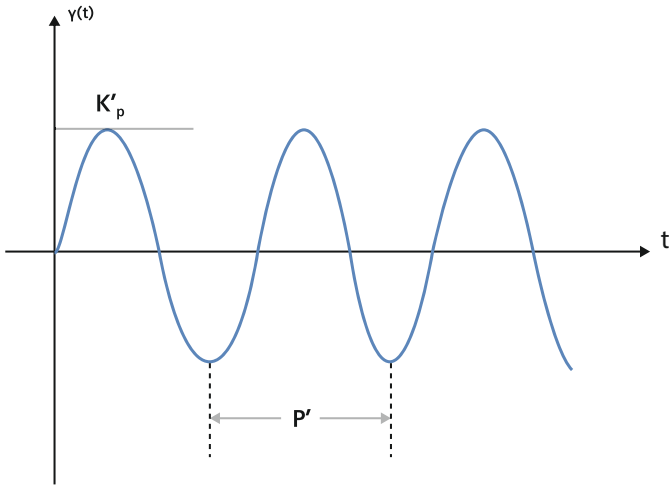
- Forward/heating pretuning successfully performed
- | setpoint - process value | < 0.05 × |upper process value - lower process value |

Forward/Heating and reverse/Cooling

- | setpoint - process value | > 0.3 × |upper process value - lower process value |
- | setpoint - process value | > 0.5 × setpoint
- setpoint > process value

14.2 Fine tuning

Fine tuning generates a constant, limited oscillation of the process value. The PID parameters are tuned for the operating point from the amplitude and frequency of this oscillation. All PID parameters are recalculated from the results. PID parameters from fine tuning usually have better master control and disturbance characteristics than PID parameters from pretuning.



K'_P amplitude

P' oscillation period

In the PID control panel, the fine tuning types varies according to different configurations:

- If you don't enable bidirectional output in the PID loop wizard, only one fine tuning type is available:
  - Fine tuning
- If you enable bidirectional output and select "Normal" as the "Controller Type" in the PID loop wizard, the following fine tuning types are available:
  - Fine tuning forward
  - Fine tuning reverse
- If you enable bidirectional output and select "Temperature" as the "Controller Type" in the PID loop wizard, the following fine tuning types are available:
  - Fine tuning heating
  - Fine tuning cooling

Tuning Offset

You can specify a temporary offset for fine tuning and the temporary offset works as an output for the opposite direction:

- For the fine tuning type "Fine tuning forward/heating", you can set the reverse offset parameter.
  Before starting tuning, enter a reverse tuning offset that can make the process value decrease to a value smaller than the set point value. The reverse tuning offset works as an output to generate a reverse oscillation of the process value.
- For the fine tuning type "Fine tuning reverse/cooling", you can set the forward offset parameter.

Note

- The value for tuning offset is from 0% to 100%.
- If you set tuning offset to 0%, the tuning offset is disabled.

Note

- If you configure a tuning offset, the tuning offset works as an output to generate a forward or reverse oscillation for the process value.
- If you don't configure a tuning offset, the PID output at the start of fine tuning plus the "initial output step" value works as an output to generate a forward oscillation for the process value, or the PID output at the start of fine tuning minus the "initial output step" value works as an output to generate a reverse oscillation for the process value.
- If the process value cannot cross the set point, or it takes long time to make the process value cross the set point, adjust the tuning offset or the initial output step.

Two-step control

If the process value is close to the set point, you can perform the fine tuning directly.

If the process value is far apart from the set point value, and you have the requirement to perform the fine tuning, the PLC will use the two-step control method automatically to make the process value close to the set point value.

14.3 PID loop definition table

The loop table allocates memory from the starting address you enter for Table (TBL) in the PID instruction box as follows. The PID instruction for the S7-200 SMART CPU V3.0 references this loop table that contains the loop parameters.

- Unidirectional: 120 bytes
- Bidirectional: 240 bytes

If you use the PID Tune control panel, all interaction with the PID loop table is handled for you by the control panel. If you need to provide auto-tuning capability from an operator panel, your program must provide the interaction between the operator and the PID loop

table to initiate and monitor the auto-tuning process, and then apply the suggested tuning values.

Table 14-1 Loop table for unidirectional PID

Offset	Field	Format	Type	Description
0	Process variable (PV_n)	REAL	In	Contains the process variable, which must be scaled between 0.0 and 1.0.
4	Setpoint (SP_n)	REAL	In	Contains the setpoint, which must be scaled between 0.0 and 1.0.
8	Output (M_n)	REAL	In/Out	Contains the calculated output, scaled between 0.0 and 1.0.
12	Gain (K_C)	REAL	In	Contains the gain, which is a proportional constant. Can be a positive or negative number.
16	Sample time (T_S)	REAL	In	Contains the sample time, in seconds. Must be a positive number.
20	Integral time or reset (T_I)	REAL	In	Contains the integral time or reset, in minutes.
24	Derivative time or rate (T_D)	REAL	In	Contains the derivative time or rate, in minutes.
28	Bias (MX)	REAL	In/Out	Contains the bias or integral sum value between 0.0 and 1.0.
32	Previous process variable (PV_{n-1})	REAL	In/Out	Contains the value of the process variable stored from the last execution of the PID instruction.
36	PID Extended Table ID	ASCII	Constant	'PIDA' (PID Extended Table for unidirectional PID, Version A): ASCII constant 'PIDB' (PID Extended Table for bidirectional PID, Version B): ASCII constant
40	AT Control (ACNTL)	BYTE	In	See the table "Expanded description of control and status fields".
41	AT Status (ASTAT)	BYTE	Out	See the table "Expanded description of control and status fields".
42	AT Result (ARES)	BYTE	In/Out	See the table "Expanded description of control and status fields".
43	AT Config (ACNFG)	BYTE	In	See the table "Expanded description of control and status fields".
44	Reserved	REAL	In	Reserved
48	Reserved	REAL	In	Reserved
52	Initial Output Step (STEP)	REAL	In	Normalized size of the step change in the output value used to induce oscillations in the PV (suggested range: 0.1 to 0.4).
56	Watchdog Time (WDOG)	REAL	In	Maximum time allowed between zero crossings in seconds (range: 60 to 7200).
60	Suggested Gain (AT_K_C)	REAL	Out	Suggested loop gain as determined by the auto-tune process.
64	Suggested Integral Time (AT_T_I)	REAL	Out	Suggested integral time as determined by the auto-tune process.
68	Suggested Derivative Time (AT_T_D)	REAL	Out	Suggested derivative time as determined by the auto-tune process.
72	Actual Step size (ASTEP)	REAL	Out	Normalized output step size value as determined by the auto-tune process.
76	Actual Hysteresis (AHYS)	REAL	Out	Normalized PV hysteresis value as determined by the auto-tune process.
80	Reserved	WORD	In	Reserved
82	Reserved	BYTE	In	Reserved

Offset	Field	Format	Type	Description
83	Reserved	DWORD	In	Reserved
87	Reserved	DWORD	In	Reserved
91	Loop Setpoint Range	REAL	In	The default value is 100.0.
95	Offsets for fine-tuning	REAL	In	Offsets for fine-tunin
99	Control spec	BYTE	In	Control spec:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Fast Response System	Reserved	CNTL_TYPE	Reserved		INTEGRAL_LIM_EN	DZ_EN	RG_EN

Fast Response System: 0 - Non-fast Response System; 1 - Fast Response System

DZ_EN: 0 - Dead zone not enable; 1- enable

RG_EN : 0 - Control range not enable; 1- enable

CNTL_TYPE: 0 - Normal, 1- Temperature

INTEGRAL_LIM_EN: 0 - Integral Limit not enable; 1- enable

Offset	Field	Format	Type	Description
100	Dead Zone	REAL	In	Dead Zone: Range 0.0-1.0
104	Control Range	REAL	In	Control range: Range 0.0-1.0
108	Reserved	BYTE	In	Reserved
109	Integral Upper Limit	BYTE	In	Integral Upper Limit: Range 0-100 • 0: Integral is disabled. • 100: Integral limit is disabled.
110	Reserved	WORD	In	Reserved
112	Reserved	REAL	In	Reserved
116	Reserved	REAL	In	Reserved

Note

The table above describes the unidirectional PID loop definition.

If you configure bidirectional PID, the table above describes the PID loop definition for forward/heating side.

Table 14-2 Loop table (Offset 120-240 used for reverse/cooling side)

Offset	Field	Format	Type	Description
120	Reserved	REAL	In	Reserved
124	Reserved	REAL	In	Reserved
128	Output (M_n)	REAL	In/Out	Contains the calculated output, scaled between 0.0 and 1.0.
132	Gain (K_C)	REAL	In	Contains the gain, which is a proportional constant. Can be a positive or negative number.
136	Reserved	REAL	In	Reserved
140	Integral time or reset (T_I)	REAL	In	Contains the integral time or reset, in minutes.

Offset	Field	Format	Type	Description
144	Derivative time or rate (T_D)	REAL	In	Contains the derivative time or rate, in minutes.
148	Bias (MX)	REAL	In/Out	Contains the bias or integral sum value between 0.0 and 1.0.
152	Reserved	REAL	In	Reserved
156	PID Extended Table ID	ASCII	Constant	'PIDB' (PID Extended Table for bidirectional PID, Version B): ASCII constant
160	AT Control (ACNTL)	BYTE	In	See the table "Expanded description of control and status fields".
161	AT Status (ASTAT)	BYTE	Out	See the table "Expanded description of control and status fields".
162	AT Result (ARES)	BYTE	In/Out	See the table "Expanded description of control and status fields".
163	AT Config (ACNFG)	BYTE	In	See the table "Expanded description of control and status fields".
164	Reserved	REAL	In	Reserved
168	Reserved	REAL	In	Reserved
172	Initial Output Step (STEP)	REAL	In	Normalized size of the step change in the output value used to induce oscillations in the PV (suggested range: 0.0 to 0.4).
176	Watchdog Time (WDOG)	REAL	In	Maximum time allowed between zero crossings in seconds (range: 60 to 7200).
180	Suggested Gain (AT_K_C)	REAL	Out	Suggested loop gain as determined by the auto-tune process.
184	Suggested Integral Time (AT_T_I)	REAL	Out	Suggested integral time as determined by the auto-tune process.
188	Suggested Derivative Time (AT_T_D)	REAL	Out	Suggested derivative time as determined by the auto-tune process.
192	Actual Step size (ASTEP)	REAL	Out	Normalized output step size value as determined by the auto-tune process.
196	Actual Hysteresis (AHYS)	REAL	Out	Normalized PV hysteresis value as determined by the auto-tune process.
200	Reserved	WORD	In	Reserved
202	Reserved	BYTE	In	Reserved
203	Reserved	DWORD	In	Reserved
207	Reserved	DWORD	In	Reserved
211	Loop Setpoint Range	REAL	In	The default value is 100.0.
215	Offsets for fine-tuning	REAL	In	Offsets for fine-tuning
219	Control spec	BYTE	In	Control spec:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Fast Response System	Reserved	CNTL_TYPE	Reserved		INTEGRAL_LIM_EN	DZ_EN	RG_EN

Fast Response System: 0 - Non-fast Response System; 1 - Fast Response System

DZ_EN: 0 - Dead zone not enable; 1- enable

RG_EN : 0 - Control range not enable; 1 - enable

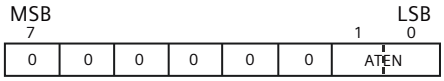
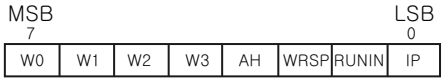
CNTL_TYPE: 0 - Normal, 1- Temperature

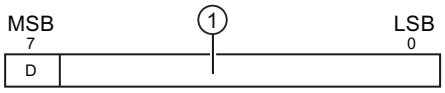
INTEGRAL_LIM_EN: 0 - Integral Limit not enable; 1- enable

220	Dead Zone	REAL	In	Dead Zone: Range 0.0-1.0
224	Control Range	REAL	In	Control range: Range 0.0-1.0

Offset	Field	Format	Type	Description
228	Reserved	BYTE	In	Reserved
229	Integral Upper Limit	BYTE	In	Integral Upper Limit: Range 0-100 • 0: Integral is disabled. • 100: Integral limit is disabled.
230	Reserved	BYTE	In	Reserved
231	Reserved	BYTE	In	Reserved
232	Reserved	REAL	In	Reserved
236	Reserved	REAL	In	Reserved

Table 14-3 Expanded description of control and status fields

Field	Description	
AT Control (ACNTL) Input - Byte		
	ATEN	Set to 1 to start fine tuning; set to 2 to start pretuning; set to 0 to abort auto-tune.
AT Status (ASTAT) Output - Byte		
	W0	Warning: The deviation setting is not four times greater than the hysteresis setting.
	W1	Warning: Inconsistent process deviations may result in the incorrect adjustment of output step value.
	W2	Warning: The actual average deviation is not four times greater than the hysteresis setting.
	W3	Warning: No inflection point detected or poor fit of sampling data. Pretuning parameters may be inaccurate.
	AH	Auto-hysteresis calculation in progress: 0 - not in progress 1 - in progress
	WRSP	Forward/heating side pretuning is successful. Wait until the process value reaches the setpoint value for reverse/cooling side pretuning.
	RUNIN	Two-step control tuning is ongoing.
	IP	Pretuning or fine tuning is in progress: 0 - not in progress 1 - in progress
	Each time a pretuning or fine tuning sequence is started the CPU clears the warning bits and sets the in progress bit. Upon completion of auto-tune, the CPU clears the in progress bit.	

Field	Description	
AT Result (ARES) Input/Output - Byte	 1 Result code	
	D	Done bit: 0 - auto-tune not complete 1 - auto-tune complete Must be set to 0 before auto-tune can start
	Result Code	00 - completed normally (suggested tuning values available) 01 - aborted by the user 02 - aborted, watchdog timed out waiting for a zero crossing 03 - aborted, process (PV) out-of-range 04 - aborted, maximum hysteresis value exceeded 05 - aborted, illegal configuration value detected 06 - aborted, numeric error detected 07 - aborted, PID instruction executed without having power flow (loop in manual mode) 08 - aborted, auto-tuning allowed only for P, PI, PD, or PID loops 09 - forward/heating fine tuning aborted, process value (PV) out-of-range 10 - reverse/cooling fine tuning aborted, process value (PV) out-of-range 11 - fine tuning aborted, output direction not supported 12 to 15 - reserved 16 - reverse/cooling pretuning aborted due to unfinished or unsuccessful forward/heating pretuning 17 - pretuning fails. Curve does not fit due to lack of data. 18 - pretuning fails. Aborted due to calculated parameter is invalid. 19 - Cannot start pretuning. The difference between SP and PV must exceed 30% of the difference between process (PV) high and low limits. 20 - Cannot start pretuning. The difference between SP and PV must exceed 50% of the SP. 21 - Cannot start pretuning. The difference between SP and PV must less than 5% of the difference between process (PV) high and low limits. 22 - Only support ZN and CHR methods for pretuning. 23 - CHR method does not support the PD control type.

Field	Description		
AT Config (ACNFG) Input - Byte	MSB 7 / LSB 0 0 0 0 PTM R1 R0 DS HS		
	PTM		Pretuning method: • 0 - ZN method; **Note:** You need to combine both the bit value of R0 and R1 to determine which ZN method is used. • 1 - CHR method;
	R1	R0	Dynamic response
	0	0	Fast response
	0	1	Medium response
	1	0	Slow response
	1	1	Very slow response
	DS	Reserved	
	HS	Reserved	

Note

The standard PID instruction is not used directly by projects with PID wizard configurations. If you use a PID wizard configuration, then your program must use "PIDx_CTRL", to activate the PID wizard subroutine.

To simplify the use of PID loop control in your application, STEP 7-Micro/WIN SMART provides a PID wizard to configure your PID loops.

14.4 Prerequisites

The loop that you want to auto-tune must be in automatic mode. The loop output must be controlled by the execution of the PID instruction. Auto-tune will fail if the loop is in manual mode.

Ideally, the loop output value needs to be near the center of the control range when auto-tuning is started. The auto-tune procedure sets up an oscillation in the process by making small step changes in the loop output. If the loop output is close to either extreme of its control range, the step changes introduced in the auto-tune procedure can cause the output value to attempt to exceed the minimum or the maximum range limit.

If this happens, the generation of an auto-tune error condition results, and the determination of less than near optimal suggested values certainly results.

14.5 Auto-tune sequence

The auto-tuning sequence begins after the hysteresis and deviation values have been determined. The tuning process begins when the initial output step is applied to the loop output.

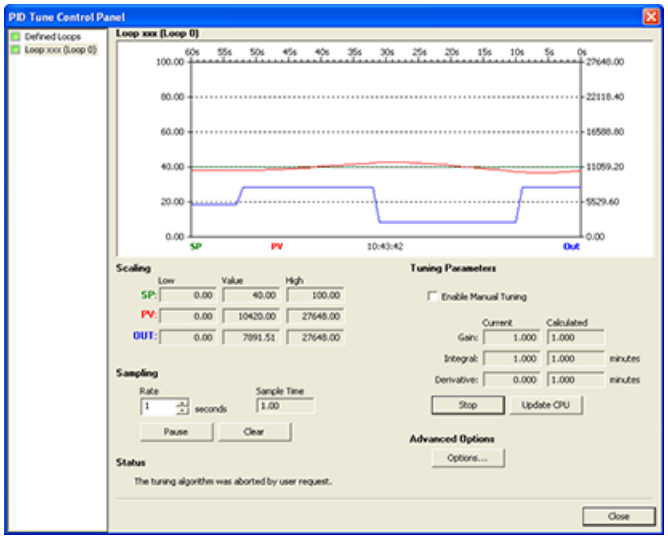
This change in output value causes a corresponding change in the value of the process variable. When the output change drives the PV away from setpoint far enough to exceed the hysteresis boundary, a zero-crossing event is detected by the auto-tuner. Upon each zero-crossing event, the auto-tuner drives the output in the opposite direction.

The tuner continues to sample the PV and waits for the next zero-crossing event. In total, the tuner requires no less than twelve zero-crossings to complete the sequence. The magnitude of the observed peak-to-peak PV values (peak error) and the rate at which zero-crossings occur are directly related to the dynamics of the process.

Early in the auto-tuning process, the output step value is proportionally adjusted once to induce subsequent peak-to-peak swings of the PV to more closely match the desired deviation amount. Once the adjustment is made, the new output step amount is written into the Actual Step Size field (ASTEP) of the loop table.

If the time between zero-crossings exceeds the zero-crossing watchdog interval time, the auto-tuning sequence is terminated with an error. The default value for the zero-crossing watchdog interval time is two hours.

The following figure shows the output and process variable behaviors during an auto-tuning sequence on a direct acting loop. You use the PID Tune control panel to initiate and monitor the tuning sequence.



Notice how the auto-tuner switches the output to cause the process (as evidenced by the PV value) to undergo small oscillations. The frequency and the amplitude of the PV oscillations are indicative of the process gain and natural frequency.

Based upon the information collected about the frequency and gain of the process during the auto-tune process, the ultimate gain and ultimate frequency values are calculated. From

these values, the suggested values for gain (loop gain), reset (integral time), and rate (derivative time) are calculated.

Note

Your loop type determines which tuning values are calculated by the auto-tuner. For example, for a PI loop, the auto-tuner will calculate gain and integral time values, but the suggested derivative time will be 0.0 (no derivative action).

Once the auto-tune sequence has completed, the output of the loop is returned to its initial value. The next time the loop is executed, the normal PID calculation will be performed.

14.6 Exception conditions

Warning conditions

Tuning execution can generate three warning conditions. Tuning execution reports these warnings in three bits of the ASTAT field of the loop table, and, once set, these bits remain set until the next auto-tune sequence is initiated:

- Warning 0: Generated if the deviation value is not at least 4X greater than the hysteresis value. This check is performed when the hysteresis value is actually known, which depends upon the auto-hysteresis setting.
- Warning 1: Generated if there is more than an 8X difference between the two peak error values gathered during the first 2.5 cycles of the auto-tune procedure
- Warning 2: Generated if the measured average peak error is not at least 4X greater than the hysteresis value
- Warning 3: No inflection point detected or poor fit of sampling data. Pretuning parameters may be inaccurate.

Error conditions

In addition to the warning conditions, several error conditions are possible. The following table lists the error conditions, along with a description of the cause of each error.

Table 14-4 Error conditions during tuning execution

Result code (in ARES)		Condition
01	aborted by user	EN bit cleared while tuning is in progress
02	aborted due to a zero-crossing watchdog time-out	Half-cycle elapsed time exceeds zero-crossing watchdog interval
03	aborted due to the process out-of-range	PV goes out-of-range: • during the auto-hysteresis sequence, or • twice before the fourth zero-crossing, or • after the fourth zero crossing

Result code (in ARES)		Condition
04	aborted due to hysteresis value exceeding maximum	User-specified hysteresis value, or automatically determined hysteresis value > maximum
05	aborted due to illegal configuration value	The following range checking errors: • Initial loop output value is < 0.0 or > 1.0 • User-specified deviation value is <= hysteresis value, or is > maximum • Initial output step is <= 0.0 or is > maximum • Zero-crossing watchdog interval time is < minimum • Sample time value in loop table is negative
06	aborted due to a numeric error	Illegal floating point number or divide by zero encountered
07	PID instruction was executed with no power flow (manual mode)	PID instruction executed with no power flow while auto-tuning is in progress or is requested
08	auto-tuning allowed only for P, PI, PD, or PID loops	Loop type is not P, PI, PD, or PID
09	forward/heating fine tuning aborted	Process value (PV) out-of-range
10	reverse/cooling fine tuning aborted	Process value (PV) out-of-range
11	fine tuning aborted	Output direction not supported
12 to 15	reserved	
16	reverse/cooling pretuning aborted	unfinished or unsuccessful forward/heating pretuning
17	pretuning fails	Curve does not fit due to lack of data
18	pretuning fails	Calculated parameter is invalid
19	cannot start pretuning	The difference between SP and PV must exceed 30% of the difference between process (PV) high and low limits.
20	cannot start pretuning	The difference between SP and PV must exceed 50% of the SP.
21	cannot start pretuning	The difference between SP and PV must less than 5% of the difference between process (PV) high and low limits.
22	invalid pretuning configuration	Only support ZN and CHR methods for pretuning
23	invalid pretuning configuration	CHR method does not support the PD control type

14.7 Notes concerning PV out-of-range (result code 3)

The process variable is considered to be in-range by the auto-tuner if its value is greater than 0.0 and less than 1.0.

If the PV is detected to be out-of-range during the auto-hysteresis sequence, then the tuning is immediately aborted with a process out-of-range error result.

If the PV is detected to be out-of-range between the starting point of the tuning sequence and the fourth zero-crossing, then the output step value is cut in half and the tuning sequence is restarted from the beginning. If a second PV out-of-range event is detected after the first zero-crossing following the restart, then the tuning is aborted with a process out-of-range error result.

Any PV out-of-range event occurring after the fourth zero-crossing results in an immediate abort of the tuning and a generation of a process out-of-range error result.

14.8 PID tune control panel introduction

STEP 7-Micro/WIN SMART V3 includes a PID Tune control panel that allows you to graphically monitor the behavior of your PID loops. In addition, the control panel allows you to initiate the auto-tune sequence, abort the sequence, and apply the suggested tuning values or your own tuning values.

Note

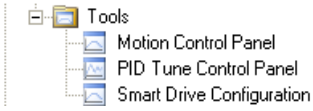
If the SP is modified in the process of control after the auto tuning is completed, Siemens recommends that you download the project again.

To use the control panel, you must be communicating with the CPU and a wizard-generated configuration for a PID loop must exist in the CPU. The CPU must be in RUN mode for the control panel to display the operation of a PID loop.

To open the PID control panel, use one of the following methods:

- Click the "PID Control Panel" button from the Tools area of the Tools menu ribbon strip.

- Open the Tools folder in the project tree, select the "PID Tune Control Panel" node and press Enter; or double-click the "PID Tune Control Panel" node.

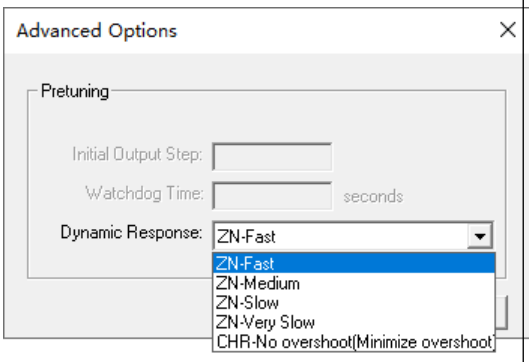
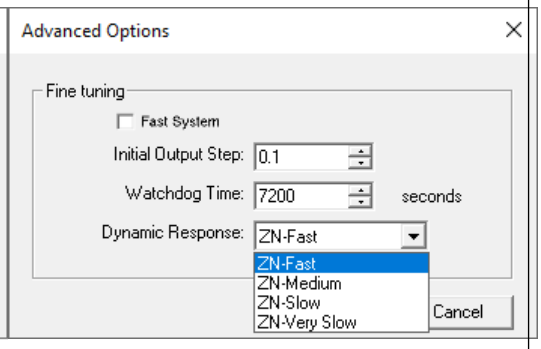


STEP 7-Micro/WIN SMART V3 opens the PID control panel if the connected CPU is in RUN mode:

The PID control panel includes the following fields:

- Current values: The values of the SP (Setpoint), PV (Process Variable), OUT (Output), Sample Time, Gain, Integral time, and Derivative time are displayed. The SP, PV, output forward/heating and output reverse/cooling are shown in green, red, blue and orange, respectively; the same color legend is used to plot the PV, SP, and OUT values.
- Graphical display: The graphical display shows color-coded plots of the PV, SP, and Output as a function of time. The PV and SP share the same vertical scale which is located at the left hand side of the graph while the vertical scale for the output is located on the right hand side of the graph.
- Tuning Parameters: At the bottom right-hand side of the screen are the Tuning Parameters (Minutes). Here, the Tuning type, Gain, Integral Time, and Derivative Time values are displayed. You click in the "Calculated" column to modify any one of the three sources for these values.
  Tuning type
  - Pretuing: pretuning forward/heating, pretuning reverse/cooling, pretuning forward/heating and reverse/cooling
  - Fine tuning: fine tuning forward/heating, fine tuning reverse/cooling
  - Manual tuning: manual tuning forward/heating, manual tuning reverse/cooling
- "Update CPU" button: You can use the" Update CPU" button to transfer the displayed Gain, Integral Time, and Derivative Time values to the CPU for the PID loop that is being monitored. You can use the "Start tuning" button to initiate an auto-tuning sequence. Once an auto-tuning sequence has started, the "Start tuning" button becomes a "Stop tuning" button.

- Sampling: In the "Sampling" area, you can select the graphical display sampling rate from 1 to 240 seconds per sample.
  You can freeze the graph by clicking the "Pause" button. Click the "Resume" button to resume sampling data at the selected rate. To clear the graph, select "Clear" from the right-mouse button within the graph.
- Advanced Options: You can use the "Options" button to further configure parameters for the tuning process. (See the figure below.)

Pretuing	Fine tuning
	
You don't need to set Initial step and Watchdog time for pretuing.	

Fast system
For some systems that change quickly, it is recommended to check the option.
Initial Output Step
The initial output step means the initial output value of PID tuning. The initial step value is represented with a percentage of the maximum power for the actual output.
After the PID fine tuning starts, the controller will actively change PID output value to observe the reaction of the entire system.
If you don't configure a tuning offset, the PID output at the start of fine tuning plus the "initial output step" value works as an output to generate a forward oscillation for the process value, or the PID output at the start of fine tuning minus the "initial output step" value works as an output to generate a reverse oscillation for the process value.

Note

- To generate a constant and limited oscillation of the process value, it is recommended that you set the "initial output step" parameter.
- The value for initial output step is between 0 and 0.4 (0 is excluded).

Watchdog Time
The process value must reach or cross the set point value one time within the watchdog time. Otherwise, a watchdog timeout error will occur.
If you know in advance that the system response is slow, you can increase the watchdog time.
Dynamic Response

– ZN-Fast: a fast response may have overshoot and would correspond to an underdamped tuning condition.

– ZN-Medium: a medium speed response may be on the verge of having overshoot and would correspond to a critically damped tuning condition.

- ZN-Slow: a slow response may not have any overshoot and would correspond to an overdamped tuning condition.
- ZN-Very Slow: a very slow response may not have overshoot and would correspond to a heavily overdamped tuning condition.
- CHR-No overshoot (Minimize overshoot): this response may have minimal or no overshoot.

In the "Dynamic Response" field, use the dropdown button to select the type of loop response that you wish to have for your process.

Note

If your PID parameters obtained by pretuning are not satisfactory, it is recommended to change the dynamic response method to obtain better PID parameters.

Loop monitoring

After you have completed the pretuning or fine tuning sequence and have transferred the suggested tuning parameters to the CPU, you can use the control panel to monitor your loop's response to a step change in the setpoint.

To eliminate the ringing at the expense of the speed of response, select a medium or a slow response and re-run the auto-tuning process.

After you have a good starting point for the tuning parameters for your loop, you can use the control panel to tweak the parameters. Then you can monitor the loop's response to a setpoint change. In this way, you can fine tune your process for an optimum response in your application.

To simplify the use of PID loop control in your application, STEP 7-Micro/WIN SMART V3 provides a PID wizard to configure your PID loops.

Open loop motion control(PLCopen) 15

The new Motion Control enhances your automation experience with the following new features:

- The upgraded configuration interface simplifies motion control parameter setup
- Fully embraces the PLCopen standard
- Support for variable programming

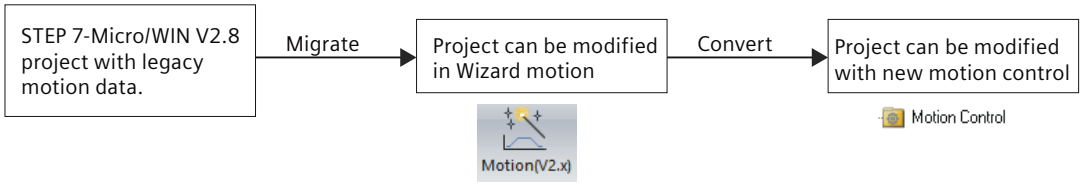
STEP 7-Micro/WIN SMART V3 Motion Control feature enables you to add and configure the following motion control components:

- Virtual Axis: virtual axis can execute motion control instructions but will not output a pulse train.
- PTO Axis : PTO (Pulse Train Output) axis uses the high-speed pulse ports integrated in the PLC to control servo motors or other actuators.
- Encoder Axis: incrementally record the number of input pulses, and display the corresponding position and speed in the axis information.
  The external encoder technology object detects a position and makes this available to the controller. The detected position can be evaluated in the user program.
- Cam: an electronic cam controls the slave axis in synchronization with the master axis, using pulse trains, according to a pre-defined electronic cam table.
  The cam technology object ("TO_Cam") describes the dependency of an output variable on an input variable in a unit-neutral manner. The cam defines a transfer function f(x) that is used to couple the master and slave axes during camming. The transfer function f(x) is defined using key points and interpolation function.
  The cam technology object of the type "TO_Cam" can contain up to 65535 points and 360 segments.
- Axes group: Cartesian Coordinate System 2D/3D Axes Group.

New motion control and legacy motion wizard are mutually exclusive

Siemens recommends that you use the new motion control, while also keeping the motion wizard to open and modify the projects you created with STEP 7-Micro/WIN SMART V2.8.

- Legacy motion wizard is disabled and gray out by default.
- STEP 7-Micro/WIN SMART V3 allows you migrate a V2.8 project with legacy motion data to STEP 7-Micro/WIN SMART V3 and modify it either with motion wizard or the new motion control.
  - After migration, the new motion control and instructions are disabled by default. You can use motion wizard to modify you legacy project.
  - If you want to use the new motion control to modify the migrated project, you need to covert it first. After converting, motion wizard and legacy instructions are disabled.



Using an axis

Perform these tasks to configure a axis:

- Add axis (Page 779)
- Configure the axis
- Writing programs with axis instructions

Using a cam

Perform these tasks to configure a cam:

- Add axis (Page 779)
- Configure the axis
- Add Cam
- Configuring the Cam (Page 800)
- Writing programs with cam instructions

Using Axes Group

Perform these tasks to configure axes group:

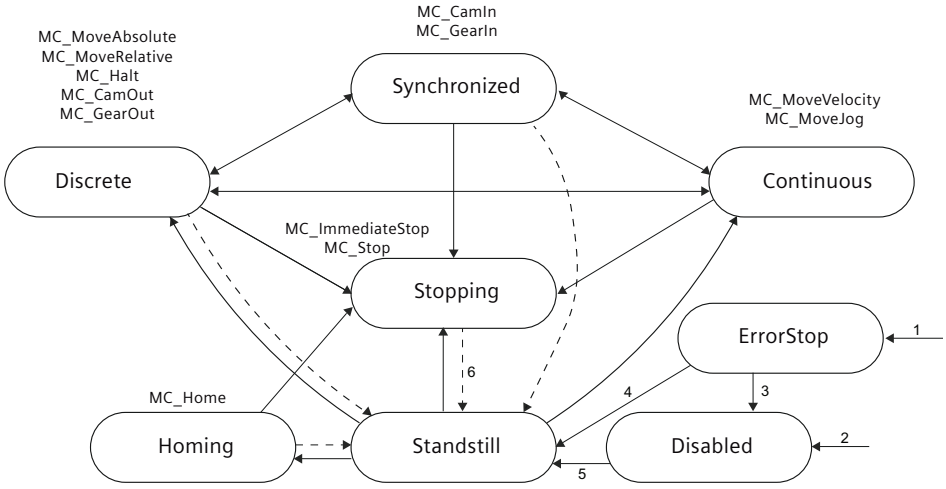
- Add axes (Page 779)
- Configure these axes
- Add axes group (Page 779)
- Configure the axes group (Page 807)
- Writing programs with axes group instructions

15.1 State changes

15.1.1 Single axis state machine

Single axis state

The movement of the axis after starting the corresponding motion control command for the axis is shown in the figure below.
The motion control instructions are executed sequentially, and the axis is in any of the states in the table below.



① Enter this state when the axis' fault detection logic detects a fault

② If axis OK and MC_Power.Enable=FALSE

③ When calling MC_Reset to reset an axis fault and MC_Power.Status=FASLE

④ When calling MC_Reset to reset an axis fault and MC_Power.Status=TRUE

⑤ When MC_Power.Enable=TRUE and MC_Power.Status=TRUE

⑥ When MC_Stop(MC_Immediate-Stop).Done=TRUE and MC_Stop(MC_Immediate-Stop).Execute=FALSE

Status		Description
Servo OFF	Disabled	Axis is in servo-off, stop in process or start ready.
	ErrorStop	Axis is in servo off, stop in progress when the axis' fault detection logic detects a fault.

Status		Description
Servo ON	Standstill	Axis is in servo-ON, stop state.
	Discrete	The state of executing positioning with a specified target position. This includes the state of waiting for the positioning to be completed and the state of setting the overshoot value to "0" and the speed to "0" during the positioning action.
	Continuous	The state of executing continuous action without a specified target position. Changes to this state during velocity control. This also includes the state in which the target velocity is set to “0” and the velocity is “0”, and the state in which the overshoot value is set to “0” and the velocity is “0” in continuous operation.
	Synchronized	Synchronized control of the state of the axes by synchronized control instructions. It also includes the synchronization waiting state after a synchronization control instruction switch.
	Stopping	The state in which the axis is stopped by the `MC_Stop` instruction or the `MC_ImmediateStop` instruction. This also includes the state in which Execute is TRUE after stopping with the MC_Stop instruction. In this state, no action can be executed by an axis instruction.
	ErrorStop	The axis is servo-ON and the axis' fault detection logic detects a fault. This includes the state in which the axis decelerates due to an abnormal situation. The axis motion instruction cannot be executed in ErrorStop state. You must use the `MC_Reset` instruction to reset the axis.
	Homing	The state when using `MC_Home` searches the home position.

Note

Only the following instructions can be called in an interrupt program. Other motion control instructions cannot be called in an interrupt program.

- MC_Stop
- MC_Halt
- MC_ImmediateStop

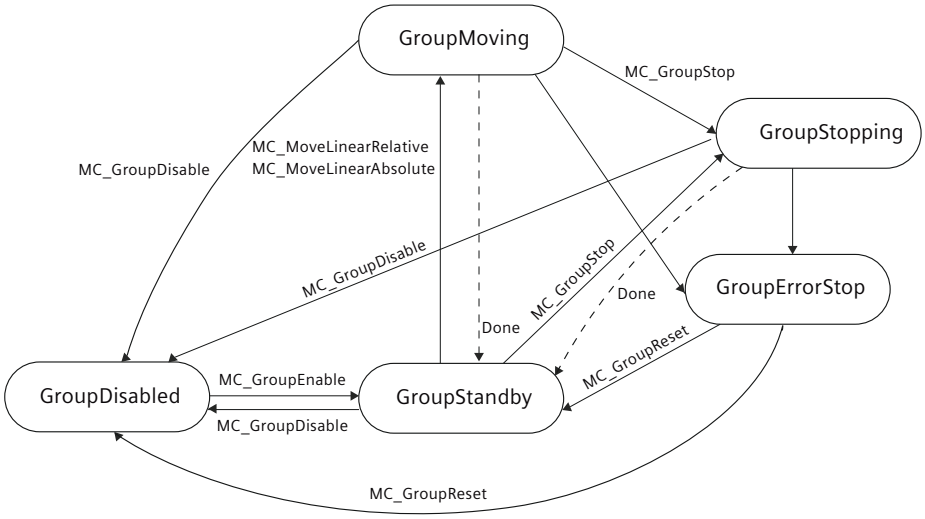
Note

The Jerk of the abort instruction does not take effect. For example, when MC_Stop is used to abort MC_MoveRelative, the Jerk of MC_Stop does not take effect.

15.1.2 Axes group state machine

Axes Group state machine

The state machine of the axes group after starting the corresponding motion control command for the axes group is shown in the figure below.



GroupError-Stop	Enter this state when the axes group' fault detection logic detects a fault
GroupStand-by	• When calling MC_GroupEnable while axes group is in the GroupDisabled state • When axes group GroupMoving completed • When axes group completed GroupStopping and the Execute=FALSE • When calling GroupReset while the axes group is stopped due to an error
GroupDisa-bled	• During an ErrorStop, if the MC_Power.Status of any axis is FALSE, the axes group can directly return to the GroupDisabled state after calling MC_Group-Reset • When calling MC_GroupDisable while axes group is moving, stopping or in standby, axes group enters the state GroupDisabled
GroupMov-ing	When calling MC_GroupLinearRelative or MC_GroupLinearAbsolute while axes group is in the GroupStandby state
GroupStop-ping	When calling MC_GroupStop while axes group is in the GroupStandby or Group-Moving state, axes group enters the state GroupStopping after excuting MC_GroupStop

Status	Description
axes group disabled	The state when the axes group is not in use.
axes group enabled	The state when the axes group is in use.

Status		Description
	GroupStandby	The state of action commands for axes group instructions that have not been executed.
	GroupMoving	The state of executing positioning with a specified target position. This includes the state of waiting for the positioning to be completed .
	GroupStopping	The state in which the axes are stopped by the `MC_GroupStop` instruction. This also includes the state when the axis is stopped, but the Execute of MC_GroupStop remains TRUE. In this state, no action can be executed by an axes group instruction.
	GroupErrorStop	The axes group' fault detection logic detects a fault. This includes the state in which the axis decelerates due to an abnormal situation. The axis motion instruction cannot be executed in this state.

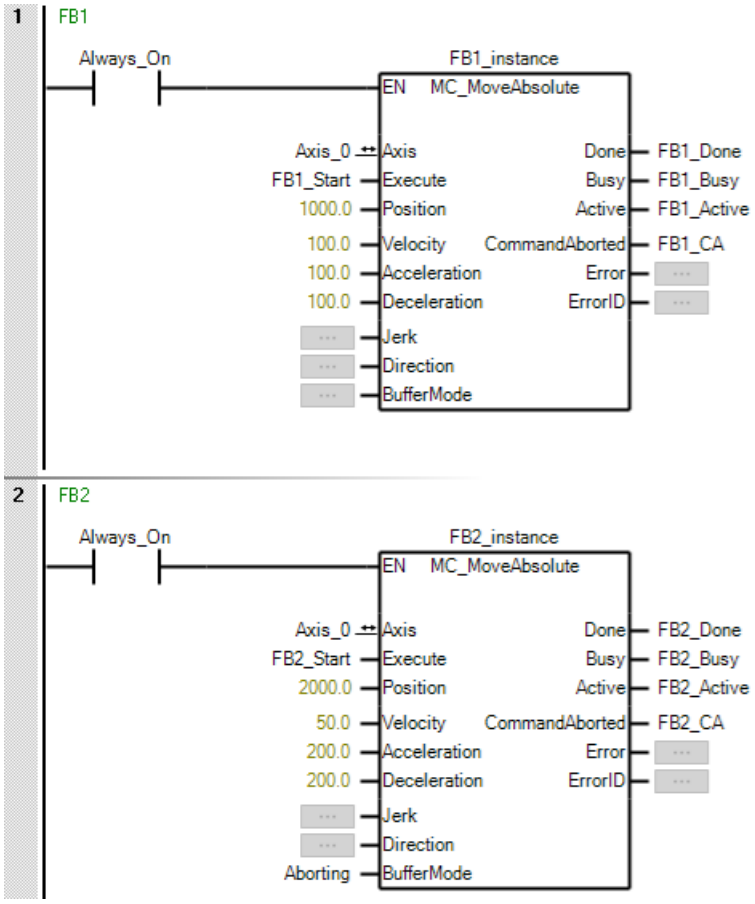
15.1.3 Single-axis command buffered mode

The following table gives an overview of single-axis command buffered mode.

S7-200 SMART CPU 3.0 can buffer up to 8 instructions.

Instruction	Can be specified as a buffered command	Can be followed by a buffered command	Relevant signal to activate the next buffered instruction
MC_Power	NO	YES	Status
MC_Home	NO	NO	--
MC_Stop	NO	NO	--
MC_Halt	NO	YES	Done
MC_MoveAbsolute	YES	YES	Done
MC_MoveRelative	YES	YES	Done
MC_MoveVelocity	YES	NO	--
MC_CamIn	YES	NO	--
MC_CamOut	NO	NO	--
MC_GearIn	YES	NO	--
MC_GearOut	NO	NO	--
MC_PhasingRelative	NO	NO	--
MC_ImmediateStop	NO	NO	--
MC_MoveJog	NO	NO	--

Example: Aborting motion



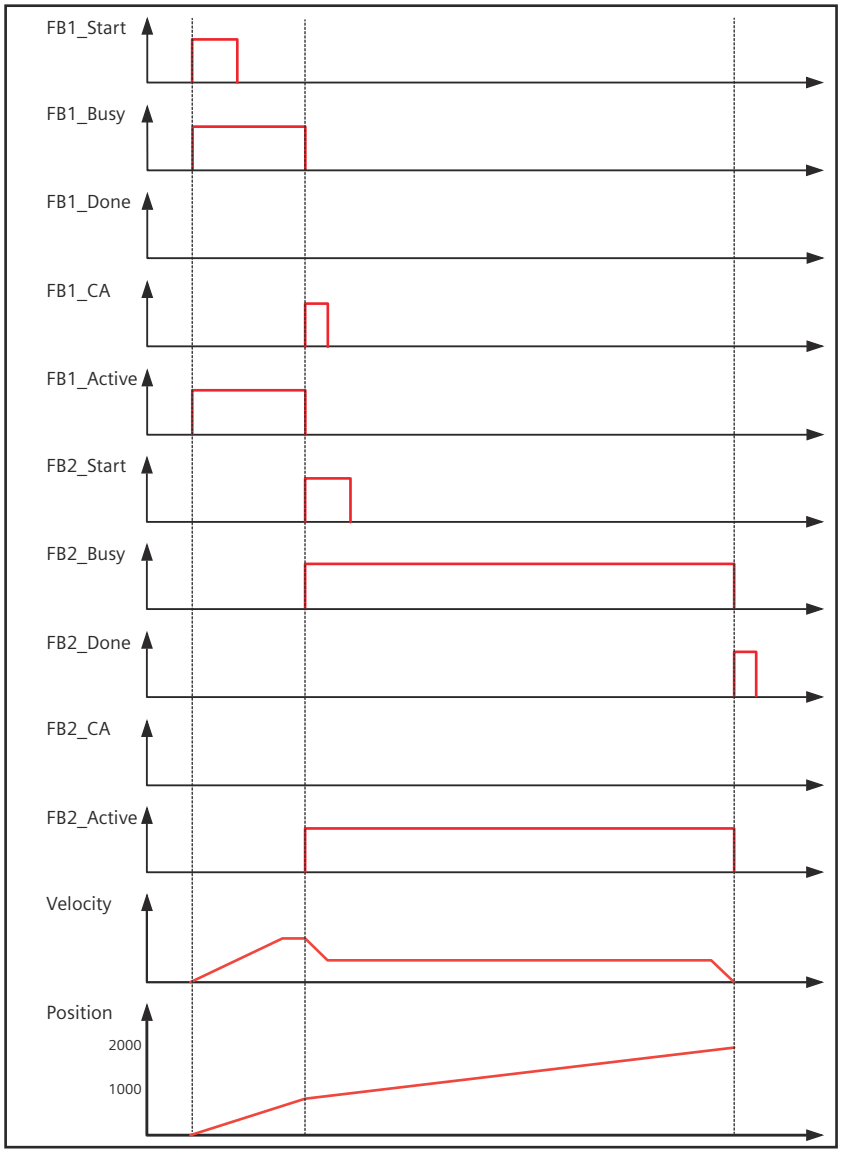
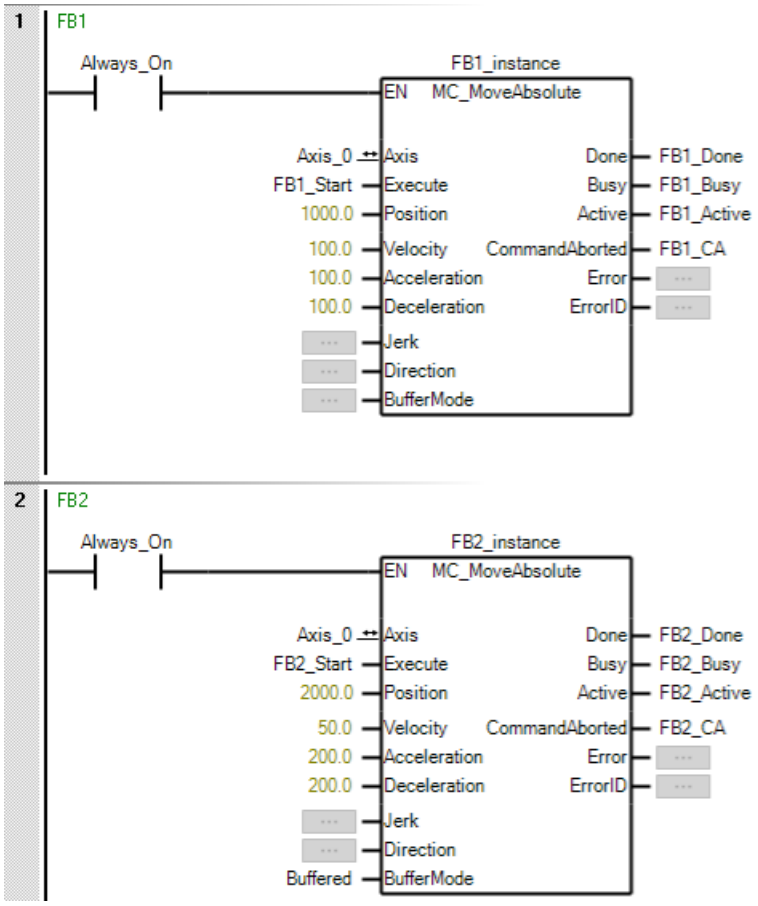


Figure 15-1 Timing diagram for example above with FB2 interrupting FB1 (Executing FB2 immediately from the breakpoint)

Example: Buffered motion



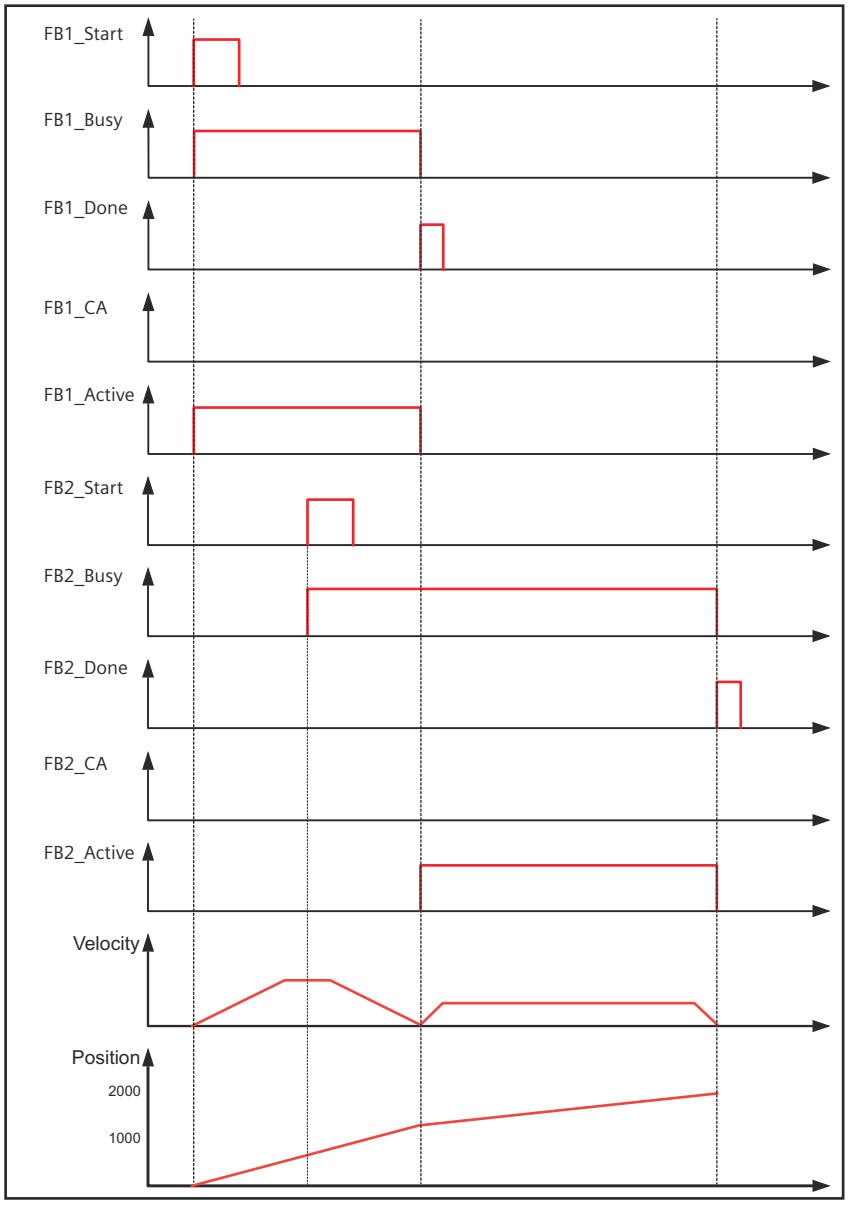


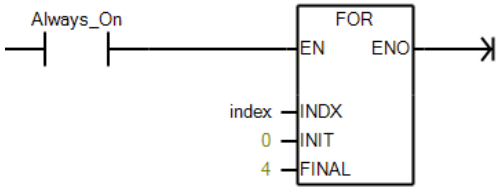
Figure 15-2 Timing diagram for example above in Buffered Mode (Stopping to velocity 0 and starting FB2 at that point without delay)

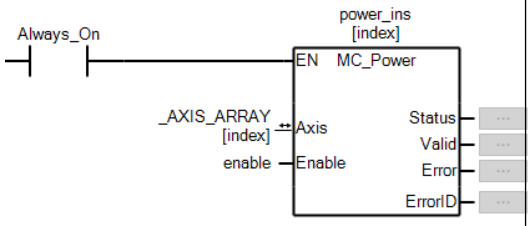
15.1.4 Achieve multi-axis control with arrays

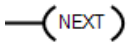
You can achieve multi-axis control by calling array.

In the example below, define five axes and enable these five axes using a FOR loop and an array. _AXIS_ARRAY[0...15] corresponds to axis IDs from 0 to 15.

Table 15-1 Example: enable axes with axis IDs from 0 to 4

LAD / FBD	Description	STL
Network 1:  	Use a for loop to control multiple axes	`LD Always_On` `FOR index, 0, 4`

Network 2:
 
 	
	Enable multiple axes	`LD Always_On`
		
		`FBCALL MC_Power,`
		`power_ins[index],`
		`_AXIS_ARRAY[index],`
		`enable,`
		`power_ins[index].Status,`
		`power_ins[index].Valid,`
		`power_ins[index].Error,`
		`power_ins[index].ErrorID`

Network 3:  		

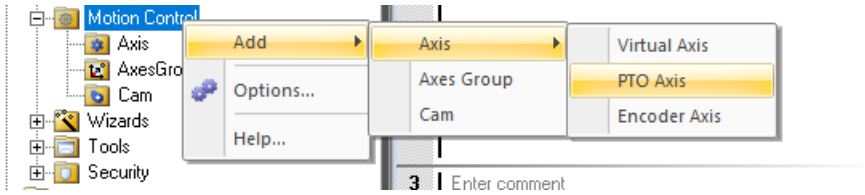
15.2 Adding and Configuring the motion control component

15.2.1 Adding a motion control component

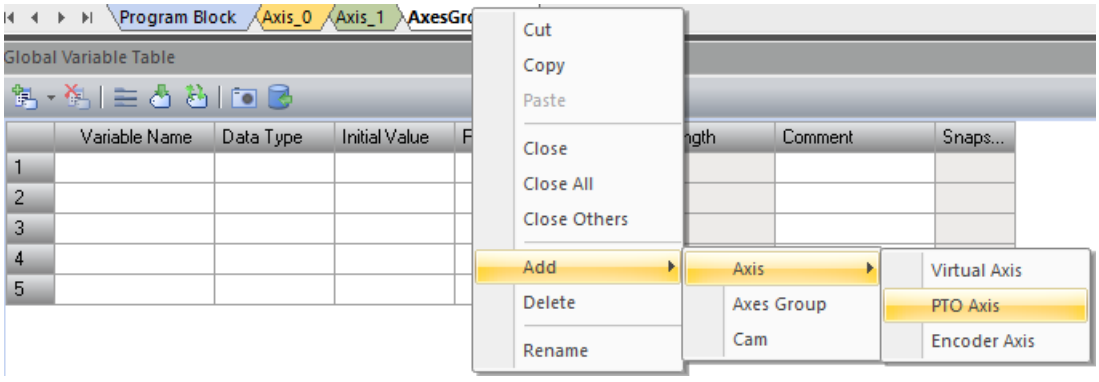
In motion control systems, an "axis" is an object that is under control.

To create a component, follow one of these steps:

- Double-click the Motion Control folder in the project tree; right-click on the component folder and click on the component name.



- Right-click within the axis editor window, select **Add** from the context menu, and select corresponding component name.



Note

Limit on the number of Axis

You can create a maximum of 16 Axis in Motion Control, with the following limitations on the number of each type of device:

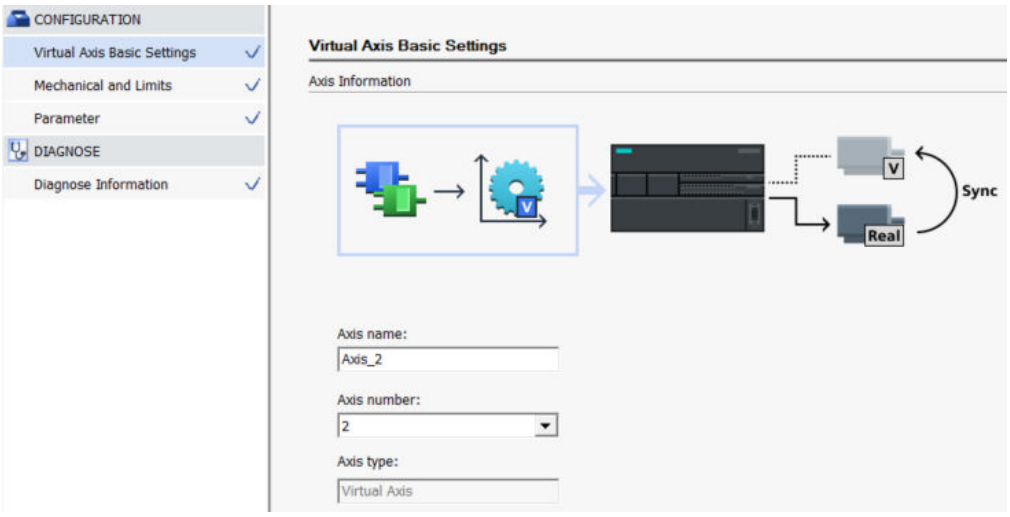
- Virtual Axis: 16
- PTO Axis: 5
- Encoder Axis: 2
  Note: The use of HSC6 and HSC7 is incompatible with the use of encoder axes. Therefore, if either HSC6 or HSC7 is enabled, only one encoder axis can be used. If both HSC6 and HSC7 are enabled at the same time, the encoder axis cannot be used, and vice versa.
- Cam: 8
- Axes Group: 2

15.2.2 Configuring the Virtual Axis

15.2.2.1 Virtual Axis Basic Setting

After a virtual axis is added, the "Virtual Axis Basic Setting" dialog displays as shown in the following figure.

You can configure a custom name for the newly added axis. The default name is "Axis_x", where "x" is equal to the axis created in the numeric order.



Note

The axis name can contain up to 23 characters, including letters, numbers, Chinese characters, underscores and extended characters (ASCII 128 to ASCII 255). The first character is restricted to alpha or underscore. It is illegal to use keywords as variable names, or to use names that begin with a number, contain characters that are not alphanumeric, or are same with existing Instruction name or keyword. Axis name cannot be duplicated or same with existing variable name.

15.2.2.2 Mechanical and Limits

Mechanical

When you click the "Mechanical and Limits" in the "CONFIGURATION" panel, the following dialog displays:

Configuration items	Mode	Description
Axis mode	Linear	In this mode, the axis can move along a straight line, typically guided by linear rails or tracks.
	Rotary	In this mode, the axis can rotate around a central point or axis of rotation.

Configuration items	Mode	Description
Measurement system	pulses	• All speeds in controlling the motion of the axis are expressed in units of pulses/s. • All distances in controlling the motion of the axis are expressed in units of pulses. Pulses mode does not support the use of "Reduction gear".
	cm, mm, inch, or degree	All speeds in controlling the motion of the axis are expressed in units of "Measurement system" per second. The mandatory parameters to configure are as follows: • Number of pulses per rotation of the motor/encoder (refer to your motor's/encoder's or drive's data sheet). • The movement of the work table in one rotation If you select "Use reduction" in "Reduction gear", you must configure the parameters "Gear numerator" and "Gear denominator".
Reduction gear[1]	No reduction	The output speed of the motor/encoder or axis is directly proportional to the input speed, with no additional speed reduction. In essence, this means that motion is transferred from input to output at a 1:1 ratio.
	Use reduction	By setting "Gear numerator" and "Gear denominator," you can determine the ratio of the reduction gear, which affects the changes in output speed and torque.

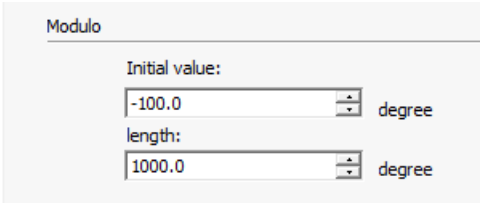
[1] **Note**: The range of the "Number of pulses per rotation of the motor/encoder" parameter is 1 to 9,999,999 pulses. The default value for this parameter is 5,000.

Modulo

The "Modulo" configurations can only be applied if you select "Rotary" for the **Axis mode**.

"Initial value" defines from which degree you want the axis to start moving, and "Length" defines how much degrees you want the axis to move. The module range is left-closed and right-open. For example, in the following example, the range of the modulo is [-100, 900).

- If you set the position to 900.0 through MC_Home, after executing the instruction, the axis position display as -100.0.
  If you set the position to 1500.0 through MC_Home, after executing the instruction, the axis position display as 400.0.



Software limit

The Software limit is disabled by default. When the input is disabled, all other related parameters are disabled. When the "Enabled Software limit" checkbox is selected, you can access the "Software negative/positive limit", and "Response" parameters.

Note

If MC_Home is not executed after power-on, the soft limits will not take effect. For the rotary axis, if the axis is outside the workspace after executing MC_Home, it is necessary to execute MC_Home or manually adjust the mechanical structure to position between Pos LMT and Neg LMT.

Note

When the maximum acceleration of an axis in the axis configuration is significantly greater than the maximum deceleration of that axis, the deceleration during a soft limit slowdown may exceed the maximum deceleration limit.

Software negative/positive limit

You can define the position at which the axis stops moving.

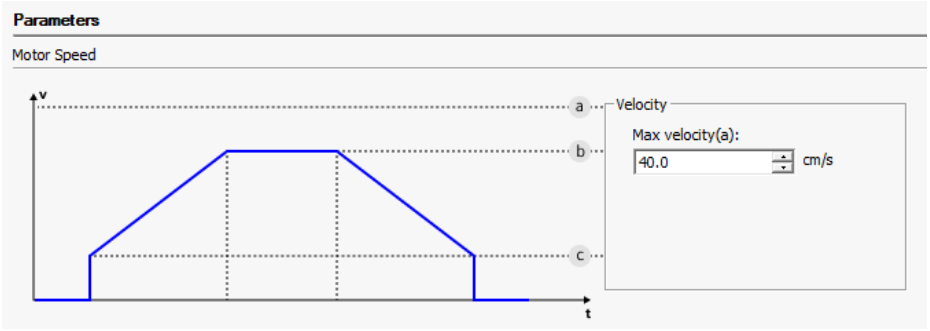
Response

The deceleration rate for "Decelerated Stop" is defined by the "Max acceleration/deceleration" in Parameters (Page 783).

15.2.2.3 Parameters

Motor Speed

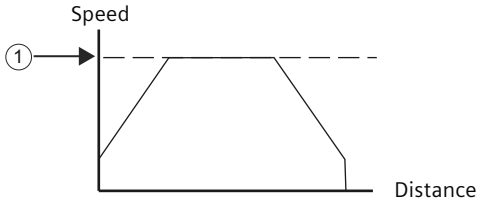
In the Motor Speeds dialog, you can define the maximum velocity for your application.



Max velocity

- Enter the velocity value for the optimum operating speed of your application within the torque capability of your motor. The torque required to drive the load is determined by friction, inertia, and the acceleration/deceleration times.
- The measurement unit of the velocity is configured in the Measurement System (Page 781); the default value is 40.0 engineering units/s or 40 pulses/s.

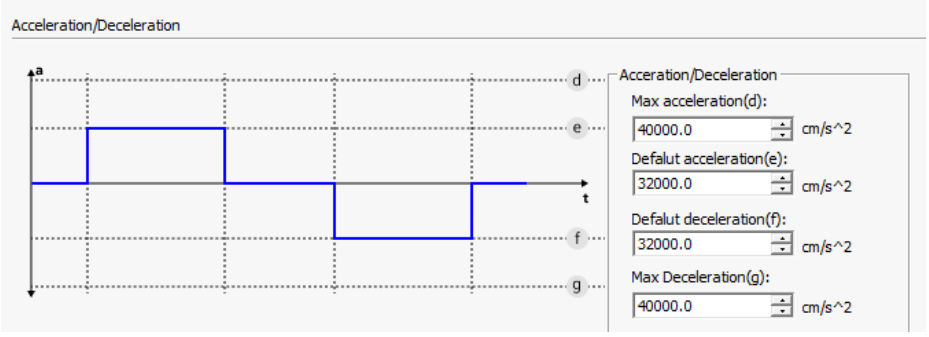
The maximum velocity define the ramping characteristics of the Axis of Motion as shown in the following figure:



Motor data sheets have different ways of specifying the Start/Stop velocity for a motor and given load.

Acceleration/Deceleration

In the Acceleration/Deceleration dialog, you can define the acceleration speed and deceleration speed for your application.



- **Max acceleration**: used to limit the Acceleration PIN for user-configured motion instructions.
- **Default acceleration**: the default value for the acceleration parameter in Motion Control Instructions.
- **Default deceleration**: the default value for the deceleration parameter in Motion Control Instructions.
- **Max deceleration**: used to limit the Deceleration PIN for user-configured motion instructions.
  and
  - For virtual axis, the maximum value used in Decelerated Stop for software limits.
  - For PTO axis, the maximum value used in Decelerated Stop for hardware limits/software limits, and Decelerated Stop for STP function in the Others configuration.
    Motor acceleration and deceleration settings are determined by trials and errors. You should start by entering a larger value. Optimize these settings for the application by gradually adjusting the input values until the motor starts to stall.

Jerk

In the Jerk dialog, you can select the "Enable Jerk" checkbox to apply a jerk compensation for acceleration and deceleration. Jerk compensation provides smoother position control by reducing the Jerk (rate of change) in acceleration and deceleration parts of the move profile. Reducing Jerk improves position tracking performance.

Jerk compensation can only be applied to simple one step profiles.

This compensation is applied equally to the beginning and ending portions of both the acceleration and deceleration curve. Assigning a value to Jerk will increase the total acceleration and deceleration time.

15.2.2.4 Diagnose Information

Command

The diagnose function can be enabled to monitor the most important status and error messages for the configured axis.

You can enable the motion control diagnose process by click on the "Start" button, and disable by click on the "Stop" button.

Axis status

The status of the axis is shown in the dialog. The details would be displayed in "ErrorCode (Page 866)", "Position", "Velocity", "Direction" dialogs.

Status message - Axis	Description
Error	An axis error occurred. Refer to the ErrorID and ErrorInfo parameters in the Motion Control instructions for details
Disabled	The axis is disabled and cannot be controlled via Motion Control instructions
Standstill	The axis is at a standstill state
Discrete	The axis is motion in Discrete motion state
Continuous	The axis is motion in Continuous motion state
Synchronized	The axis is motion in Synchronized motion state
Homed	The axis has returned to its home position
Stopping	The axis is Stopping
Error stop	The axis is experiencing an error condition and is currently in the ErrorStop state
SW LMT+	The software positive limit switch was reached or exceeded
SW LMT-	The software negative limit switch was reached or exceeded

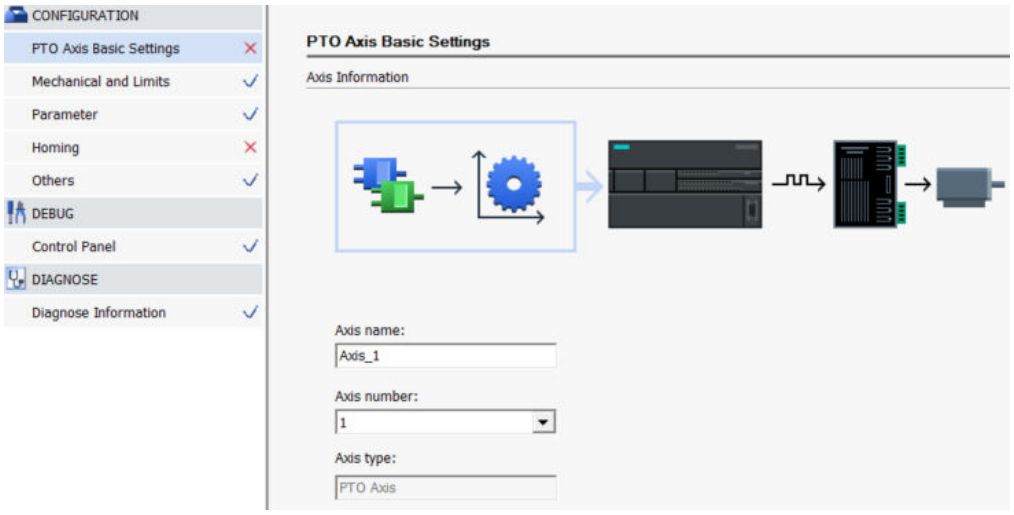
15.2.3 Configuring the PTO Axis

15.2.3.1 PTO Axis Basic Settings

Axis Information

After a PTO axis is added, the "PTO Axis Basic Settings" dialog displays as shown in the following figure.

You can configure a custom name for the newly added axis. The default name is "Axis_x", where "x" is equal to the axis created in the numeric order.



Note

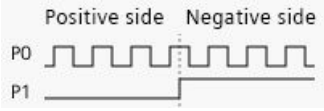
- The axis name can contain up to 23 characters, including letters, numbers, Chinese characters, underscores and extended characters (ASCII 128 to ASCII 255). The first character is restricted to alpha or underscore. It is illegal to use keywords as variable names, or to use names that begin with a number, contain characters that are not alphanumeric or in the extended character set, or are same with existing Instruction name or keyword. Axis name cannot be duplicated or same with existing variable name.
- You can create a maximum of 5 PTO Axes in Motion Control.

Output

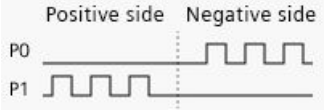
You can configure the pulse and direction output interfaces of the stepper/servo driver.

There are three options for the "Output mode" interface to the stepper/servo drive as follows:

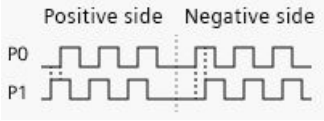
- Pulse+Direction (Single Phase 2 output): If you select the Pulse+Direction (Single Phase 2 output) option, then one output (Pulse) controls the pulsing, and one output (Direction) controls the pulse direction. Direction is high (active) if pulsing is in the positive direction. Direction is low (inactive) if pulsing is in the negative direction. Pluse+Direction (Single Phase 2 output) is shown in the figure below:



- Forward/Reverse (Two Phase 2 Output): If the Two Phase (2 Output) option is selected, one output (CW) pulse is used for the forward direction and the other output pulse is used for the negative direction. The forward/reverse (two-phase output) is shown in the following figure:



- A/B phase (2 output): If you select the A/B phase (2 output) option, then both outputs pulse at the speed specified, but 90 degrees out-of-phase. The A/B phase (2 output) is a 1X configuration, meaning 1 pulse is the amount of time between positive transitions of each input. In this case, which output transitions high first determines the direction. A leads B for the positive direction. B leads A for the negative direction. The figure below shows A/B phase (2 output):



You can switch positive and negative directions by selecting or unselecting the "Reverse" checkbox. For example, if the motor is wired in the wrong direction, you can avoid re-wiring the hardware by selecting the "Reverse" checkbox.

Note

The default setting for "Output mode" is "Pluse+Direction (Single Phase 2 output)" and the "Reverse" checkbox is unseleceted.

15.2.3.2 Mechanical and Limits

Mechanical

When you click the "Mechanical and Limits" in the "CONFIGURATION" panel, the following dialog displays:

Configuration items	Mode	Description
Axis mode	Linear	In this mode, the axis can move along a straight line, typically guided by linear rails or tracks.
	Rotary	In this mode, the axis can rotate around a central point or axis of rotation.
Measurement system	pulses	• All speeds in controlling the motion of the axis are expressed in units of pulses/s. • All distances in controlling the motion of the axis are expressed in units of pulses. Pulses mode does not support the use of "Reduction gear".
	cm, mm, inch, or degree	All speeds in controlling the motion of the axis are expressed in units of "Measurement system" per second. The mandatory parameters to configure are as follows: • Number of pulses per rotation of the motor/encoder (refer to your motor's/encoder's or drive's data sheet). • The movement of the work table in one rotation If you select "Use reduction" in "Reduction gear", you must configure the parameters "Gear numerator" and "Gear denominator".
Reduction gear[1]	No reduction	The output speed of the motor/encoder or axis is directly proportional to the input speed, with no additional speed reduction. In essence, this means that motion is transferred from input to output at a 1:1 ratio.
	Use reduction	By setting "Gear numerator" and "Gear denominator," you can determine the ratio of the reduction gear, which affects the changes in output speed and torque.

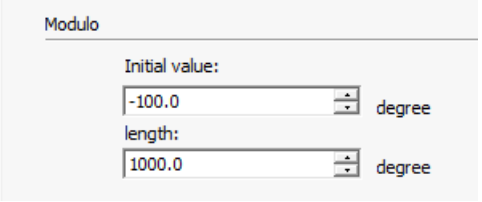
[1] **Note**: The range of the "Number of pulses per rotation of the motor/encoder" parameter is 1 to 9,999,999 pulses. The default value for this parameter is 5,000.

Modulo

The "Modulo" configurations can only be applied if you select "Rotary" for the **Axis mode**.

"Initial value" defines from which degree you want the axis to start moving, and "Length" defines how much degrees you want the axis to move. The module range is left-closed and right-open. For example, in the following example, the range of the modulo is [-100, 900).

- If you set the position to 900.0 through MC_Home, after executing the instruction, the axis position display as -100.0.
  If you set the position to 1500.0 through MC_Home, after executing the instruction, the axis position display as 400.0.



Hardware limits

You can define to which pins the Hardware negative limit (LMT-) and the Hardware positive limit (LMT+) input are assigned and the characteristics of the limit input, including Response and Trigger Level.

The LMT-/LMT+ input is disabled by default. When the input is disabled, all other related parameters are disabled. When the "Enabled Hardware negative/positive limit switches" checkbox is selected, you can access the "Input", "Response", and "Trigger Level" parameters.

Configuration items	Description
Input	You can assign LMT-/LMT+ to an input on the CPU. You can assign any of the Inputs from I0.0 through I1.7 of the CPU, but this Input (I) can no longer be used for motion control functions for this or other axis. Therefore, if another function in the motion control configuration uses an Input (I), this Input (I) is not shown in the selection drop-down list for this input. **Note**: If an input(I) is selected for a Motion function, this input(I) is not available for other functions in the CPU, including HSC and Edge Interrupt.
Trigger Level	• When the level is high, logic 1 is read when current is flowing in the input. • When the level is set to low, logic 1 is read when there is no current flow in the input. Logic 1 level is interpreted as meaning the limit has been reached. The LEDs corresponding to those inputs are illuminated when current flows in the input, regardless of the trigger level. The default level is "High".
Response	In this parameter, you can assign what should happen if the LMT-/LMT+ input goes active. You can select between the below two options: • Decelerated Stop • Immediate Stop The default setting is "Decelerated Stop".

WARNING

Risks with changes to filter time for digital input channel

If the filter time for a digital input channel is changed from a previous setting, a new "0" level input value may need to be presented for up to 12.8 ms accumulated duration before the filter becomes fully responsive to new inputs. During this time, short "0" pulse events of duration less than 12.8 ms may not be detected or counted.

This changing of filter times can result in unexpected machine or process operation, which may cause death or serious injury to personnel, and/or damage to equipment.

To ensure that a new filter time goes immediately into effect, power cycle the CPU.

Note

You must program the required filter time for your application's LMT-/LMT+ input in the system block. You can only assign the filter time of an input in the system block.

Software limit

The Software limit is disabled by default. When the input is disabled, all other related parameters are disabled. When the "Enabled Software limit" checkbox is selected, you can access the "Software negative/positive limit", and "Response" parameters.

Note

If MC_Home is not executed after power-on, the soft limits will not take effect. For the rotary axis, if the axis is outside the workspace after executing MC_Home, it is necessary to execute MC_Home or manually adjust the mechanical structure to position between Pos LMT and Neg LMT.

Note

When the maximum acceleration of an axis in the axis configuration is significantly greater than the maximum deceleration of that axis, the deceleration during a soft limit slowdown may exceed the maximum deceleration limit.

Software negative/positive limit

You can define the position at which the axis stops moving.

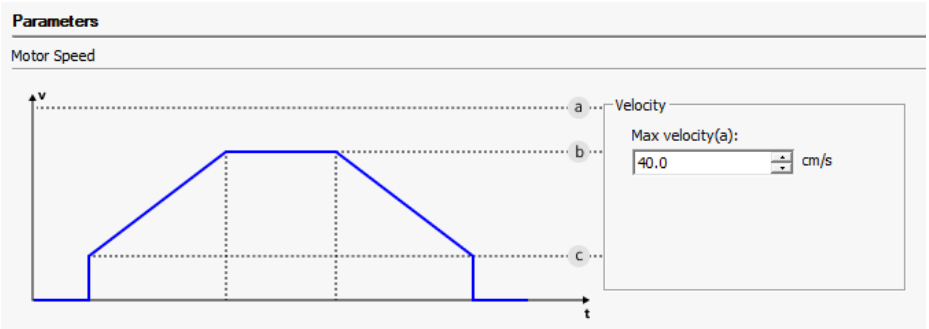
Response

The deceleration rate for "Decelerated Stop" is defined by the "Max acceleration/deceleration" in Mechanical and Limits (Page 788).

15.2.3.3 Parameters

Motor Speed

In the Motor Speeds dialog, you can define the maximum velocity for your application.



Max velocity

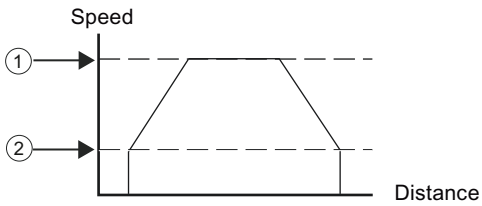
- Enter the velocity value for the optimum operating speed of your application within the torque capability of your motor. The torque required to drive the load is determined by friction, inertia, and the acceleration/deceleration times.
- The measurement unit of the velocity is configured in the Measurement System; the default value is 40.0 engineering units/s or 40 pulses/s. The velocity cannot be less than the start/stop velocity of STEP 7 Micro/WIN SMART V3.0 motion.

Start/Stop velocity

- The Start/Stop velocity cannot be configured, a fixed velocity is generated according to engineering unit. The system-defined minimum start-up velocity is 200 Hz.

$$\text{CPU pulse} = \frac{\text{Number of pulses per rotation of the motor(a)} * \text{Gear numerator (c)}}{\text{The movement of the work table in one rotation (b)} * \text{Gear denominator (d)}} * \text{move distance}$$

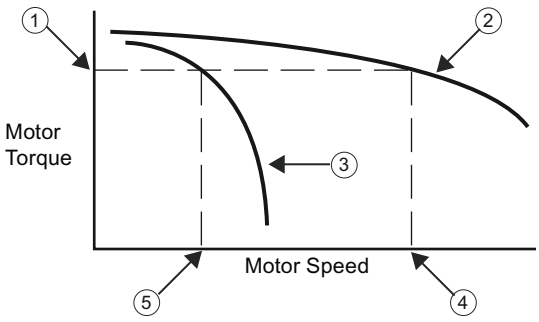
The maximum velocity and the start/stop velocity define the ramping characteristics of the Axis of Motion as shown in the following figure:



① Max velocity
② Start/Stop velocity

Motor data sheets have different ways of specifying the Start/Stop velocity for a motor and given load.

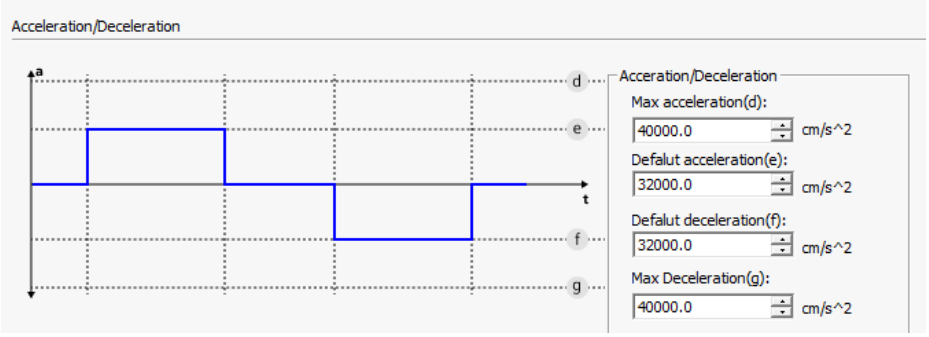
To help you select the correct speeds for your application, refer to your motor data sheet. The figure below shows a typical motor torque/speed curve:



① Torque required to drive the load
② Motor torque versus velocity characteristic
③ Start/Stop velocity versus torque: This curve moves towards lower speed as the load inertia increases.
④ Maximum velocity that the motor can drive the load: the configured Max velocity should not exceed this value.
⑤ Start/Stop velocity for this load

Acceleration/Deceleration

In the Acceleration/Deceleration dialog, you can define the acceleration speed and deceleration speed for your application.



- **Max acceleration**: used to limit the Acceleration PIN for user-configured motion instructions.
- **Default acceleration**: the default value for the acceleration parameter in Motion Control Instructions.
- **Default deceleration**: the default value for the deceleration parameter in Motion Control Instructions.
- **Max deceleration**: used to limit the Deceleration PIN for user-configured motion instructions.
  and
  - For virtual axis, the maximum value used in Decelerated Stop for software limits.
  - For PTO axis, the maximum value used in Decelerated Stop for hardware limits/software limits, and Decelerated Stop for STP function in the Others configuration.
    Motor acceleration and deceleration settings are determined by trials and errors. You should start by entering a larger value. Optimize these settings for the application by gradually adjusting the input values until the motor starts to stall.

Jerk

In the Jerk dialog, you can select the "Enable Jerk" checkbox to apply a jerk compensation for acceleration and deceleration. Jerk compensation provides smoother position control by reducing the Jerk (rate of change) in acceleration and deceleration parts of the move profile. Reducing Jerk improves position tracking performance.

Jerk compensation can only be applied to simple one step profiles.

This compensation is applied equally to the beginning and ending portions of both the acceleration and deceleration curve. Assigning a value to Jerk will increase the total acceleration and deceleration time.

15.2.3.4 Homing

RPS Signal

Enable RPS

In the RPS Signal dialog, you can define to which pin the reference point seek input is assigned and the characteristics of the RPS input, including "Trigger Level".

The RPS input defines the home position or the reference point when performing the reference point seek command.

The RPS input is disabled by default. When the input is disabled, all other parameters on the page are disabled. When the "Enabled" checkbox is checked, you can access the "Input" and "Trigger Level" parameters.

Input parameter

You can assign RPS to an input on the CPU. You can assign any of the inputs from I0.0 through I1.7 of the CPU, but this Input (I) can no longer be used for motion control functions for this or other axes. Therefore, if another function in the motion control configuration uses an Input (I), that Input (I) is not shown in the selection drop-down list for this input.

For this reason, if an input is used by another function in the Motion wizard, it does not appear in the drop-down list as a selection for this input. When the input is enabled, it is defaulted to the next available input in the list.

Level parameter

In this parameter, you can assign the active level of the RPS input. When the level is high, logic 1 is read when current is flowing in the input. When the level is set to low, logic 1 is read when there is no current flow in the input. Logic 1 level is interpreted as meaning the reference point or home position has been reached. The LEDs corresponding to those inputs are illuminated when current flows in the input, regardless of the active level. The default level is "High".

Enable ZP

You can define to which input pin the ZP input is assigned by selecting the "Enable ZP" checkbox. The Zero Pulse input (ZP) helps establish the reference point or home position used in the Reference Point Seek (RPS) command. Typically, the motor driver/amplifier pulses ZP once per revolution.

You can assign the ZP input on the CPU. You can assign from I0.0 through I1.7 of the CPU, but this Input (I) can no longer be used for motion control functions for this or other axes. Therefore, if another function in the motion control configuration uses an Input (I), this Input (I) is not shown in the selection drop-down list for this input.

Homing

Homing mark

Select whether to use the positive or negative side as the origin.

Approach direction / Homing direction

With the direction selection, you determine the approach direction used during active homing to search for the homing switch, as well as the homing direction. The homing direction specifies the travel direction the axis uses to approach the configured end of the homing switch to carry out the homing operation.

Reach the hardware limit

If this "Auto-return when hardware limit is reached" checkbox is selected, the axis is allowed to move in the reversed direction to the homing point.

Velocity

- Approach velocity: define the approaching velocity to the origin
- Homing start velocity: define the start velocity back to origin

---

Note

The limit for the input value: Start/stop velocity ≤ Approach velocity ≤ Homing velocity ≤ Max velocity

---

Origin Point Offset

If the desired home position deviates from the position of the homing switch, the home position offset can be specified in this field.
If the value does not equal 0, the axis executes the following actions following homing at the homing switch:

1. Move the axis at the homing velocity by the value of the home position offset
2. Upon reaching the "home position offset", the axis is at the home position that was specified in input parameter "Position" of the "MC_Home" Motion Control instruction.

15.2.3.5 Others

Emergency Stop

Enable STP

In the Emergency Stop dialog, you can define the pin for the stop input and the characteristics of the stop input by selecting the "Enable STP" checkbox.

The STP input has the following functions:

- Makes any active motion command ramp down to the Start-Stop Speed and stop pulsing immediately after the ramp down
- Prevents starting a new motion command when the "Trigger" is set to **Level**

- Can serve as a ready signal coming from your stepper/servo drive
- If active, you can start a new motion, once the previous motion stops, with the "Trigger" set to **Edge**

By default, the "Enable STP" checkbox is unselected, and all the other parameters on the STP dialog are disabled. When you select the "Enabled STP" check box, you can access the "Input", "Response", "Trigger", and "Level" parameters.

Input

You can assign STP to an input on the CPU. You can assign any of the Inputs from I0.0 through I1.7 of the CPU, but this Input (I) can no longer be used for motion control functions for this or other axes. Therefore, if another function in the motion control configuration uses an Input (I), this Input (I) is not shown in the selection drop-down list for this input.

Note

If you select an input for a Motion function, it is not available for other functions in the CPU, including HSC and Edge Interrupt.

Response parameter

In this parameter, you can assign what should happen if the STP input goes active. The default setting is "Decelerated Stop". You can select between the following two options:

- Decelerated Stop: Makes any active motion command ramp down to the Start-Stop velocity and stop pulsing
- Immediate Stop: Makes any active motion command stop pulsing immediately

Trigger parameter

In this parameter, you can assign the trigger of the STP input. The default setting is "Level". You can select between the following two options:

- Level: If you select a level trigger, the motion stops whenever the input is active. The level trigger setting prevents any new motion from starting until the STP input goes inactive.
- Edge: If you select an edge trigger, the motion stops only when the input goes from inactive to active. The edge trigger setting allows a new motion to start after the previous motion has been stopped, even if the STP input is still active.

When the motion stops, the inputs become inactive and the axis can begin another motion.

Probe

You can enable the probe control by selecting the "Enable probe 0" and "Enable probe 1".

Input

Select the input for the touch probe. You can assign any of the Inputs from I0.0 through I1.7 of the CPU.
You must configure the Probe before using the MC_TouchProbe instruction.

15.2.3.6 Diagnose Information

Command

The diagnose function can be enabled to monitor the most important status and error messages for the configured axis.

You can enable the motion control diagnose process by click on the "Start" button, and disable by click on the "Stop" button.

Axis status

The status of the axis is shown in the dialog. The details would be displayed in "ErrorCode", "Position", "Velocity", "Direction" dialogs.

Status message - Axis	Description
Error	An axis error occurred. Refer to the ErrorID and ErrorInfo parameters in the Motion Control instructions for details
Disabled	The axis is disabled and cannot be controlled via Motion Control
Standstill	The axis is at a standstill state
Discrete	The axis is moving in Discrete state
Continuous	The axis is moving in Continuous state
Synchronized	The axis is moving in Synchronized state
Homed	The axis has returned to its home position
Stopping	The axis is Stopping
Error stop	The axis is experiencing an error condition and is currently in the ErrorStop state
HW LMT+	The hardware positive limit switch is reached or exceeded
HW LMT-	The hardware negative limit switch is reached or exceeded
SW LMT+	The software positive limit switch is reached or exceeded
SW LMT-	The software negative limit switch is reached or exceeded
RPS	The RPS signal is triggered when the axis is homing
STP	The STP signal is triggered

15.2.4 Configuring the Encoder Axis

15.2.4.1 Encoder Axis Basic Settings

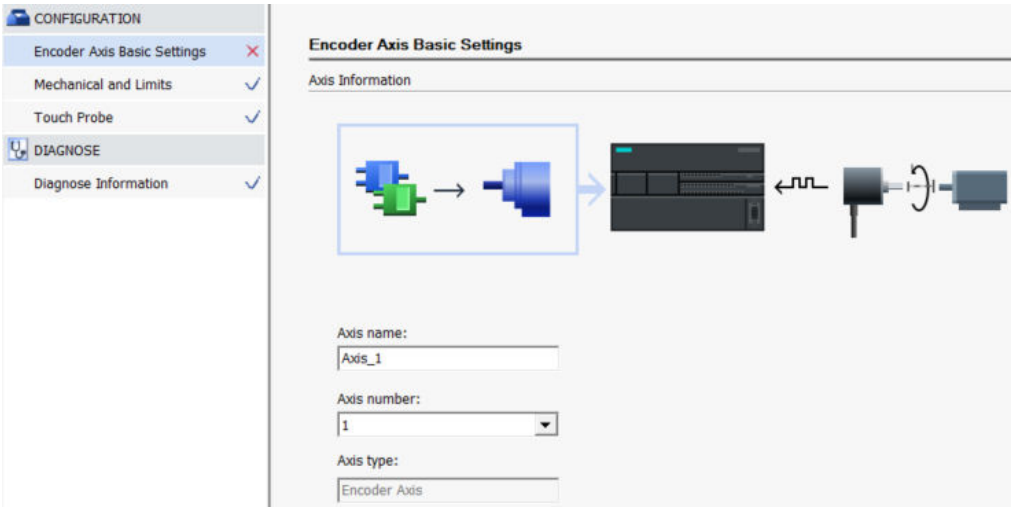
Axis Information

Note

When using encoder axis, ensure that the digital input filter configured in the System Block is ≤ 1.6μs for the selected digital input.

After an encoder axis is added, the "Encoder Axis Basic Settings" dialog displays as shown in the following figure.

You can configure a custom name for the newly added axis. The default name is "Axis_x", where "x" is equal to the axis created in the numeric order.



Note

- The axis name can contain up to 23 characters, including letters, numbers, Chinese characters, underscores and extended characters (ASCII 128 to ASCII 255). The first character is restricted to alpha or underscore. It is illegal to use keywords as variable names, or to use names that begin with a number, contain characters that are not alphanumeric or in the extended character set, or are same with existing Instruction name or keyword. Axis name cannot be duplicated or same with existing variable name.
- You can create a maximum of 2 Encoder Axis in Motion Control.
  Note: The use of HSC6 and HSC7 is incompatible with the use of encoder axes. Therefore, if either HSC6 or HSC7 is enabled, only one encoder axis can be used. If both HSC6 and HSC7 are enabled at the same time, the encoder axis cannot be used, and vice versa.

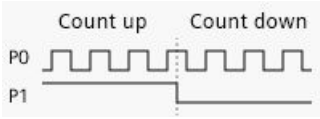
Input

You can configure the pulse and direction input interface to your stepper/servo drive.

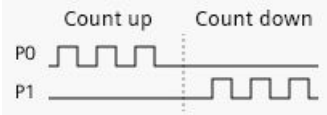
There are four options for the "Input mode" interface to the stepper/servo drive as follows:

- Pulse(A):If you select the Pulse(A) option, then the input (Pulse) receives the pulsing. The CPU accepts only positive motion commands in this mode. The figure below shows Pulse(A) (assuming positive polarity):

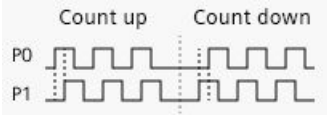
- Pluse(A) + Direction(B): If you select the Pluse(A) + Direction(B) option, then one input (Pulse) receives the pulsing, and one input (Direction) controls the pulse direction. Direction is high (active) if pulsing is in the positive direction. Direction is low (inactive) if pulsing is in the negative direction. Pluse(A) + Direction(B) is shown in the figure below (assuming the "reverse" checkbox is unselected):



- Count up(A) + Count down(B): If you select the Count up(A) + Count down(B) option, then one input pulses for positive directions, and a different output pulses for negative directions. Count up(A) + Count down(B) is shown in the figure below (assuming the "reverse" checkbox is unselected):



- Incremental encoder (AB phase shifting): If you select the Incremental encoder (AB phase shifting) option, then both inputs pulse at the speed specified, but 90 degrees out-of-phase. The Incremental encoder (AB phase shifting) is a 1X frequency, meaning 1 pulse is the amount of time between positive transitions of each input. A leads B for the positive direction. B leads A for the negative direction. The figure below shows Incremental encoder (AB phase shifting), assuming the "reverse" checkbox is unselected:



You can switch positive and negative directions by selecting or unselecting the "Reverse" checkbox. For example, if the motor is wired in the wrong direction, you can avoid re-wiring the hardware by selecting the "Reverse" checkbox.

You can set a time constant in ms, range (0-2000), for the actual velocity filter to ensure the output velocity to be more smooth.

Note

The default setting for "Input mode" is "Pluse(A)".

Note

When using the encoder axis as the master axis for the `MC_CamIn` or `MC_GearIn`, the velocity of the encoder axis is not affected by the "Time Constant of actual velocity filter" parameter in the "Encoder Axis Basic Settings" page.

15.2.4.2 Mechanical and Limits

Mechanical

When you click the "Mechanical and Limits" in the "CONFIGURATION" panel, the following dialog displays:

Configuration items	Mode	Description
Encoder count mode	Linear	In this mode, the axis can move along a straight line, typically guided by linear rails or tracks.
	Rotary	In this mode, the axis can rotate around a central point or axis of rotation.
Measurement system	pulses	• All speeds in controlling the motion of the axis are expressed in units of pulses/s. • All distances in controlling the motion of the axis are expressed in units of pulses. Pulses mode does not support the use of "Reduction gear".
	cm, mm, inch, or degree	All speeds in controlling the motion of the axis are expressed in units of "Measurement system" per second. The mandatory parameters to configure are as follows: • Number of pulses per rotation of the motor/encoder (refer to your motor's/encoder's or drive's data sheet). • The movement of the work table in one rotation If you select "Use reduction" in "Reduction gear", you must configure the parameters "Gear numerator" and "Gear denominator".
Reduction gear[1]	No reduction	The output speed of the motor/encoder or axis is directly proportional to the input speed, with no additional speed reduction. In essence, this means that motion is transferred from input to output at a 1:1 ratio.
	Use reduction	By setting "Gear numerator" and "Gear denominator," you can determine the ratio of the reduction gear, which affects the changes in output speed and torque.

[1] **Note**: The range of the "Number of pulses per rotation of the motor/encoder" parameter is 1 to 9,999,999 pulses. The default value for this parameter is 5,000.

Modulo

The "Modulo" configurations can only be applied if you select "Rotary" for the encoder count mode.

"Initial value" defines from which degree you want the axis to start moving, and "Length" defines how much degrees you want the axis to move.

15.2.4.3 Touch Probe

You can enable the probe control by selecting the "Enable probe 0" and "Enable probe 1".

Input

Select the input for the touch probe. You can assign any of the Inputs from I0.0 through I1.7 of the CPU. You must configure the Probe before using the MC_TouchProbe instruction.

15.2.4.4 Diagnose Information

Command

The diagnose function can be enabled to monitor the most important status and error messages for the configured axis.

You can enable the motion control diagnose process by click on the "Start" button, and disable by click on the "Stop" button.

Axis status

The status of the axis is shown in the dialog. The details would be displayed in "ErrorCode", "Position", "Velocity", "Direction" dialogs.

Status message - Axis	Description
Enable	The axis is enabled and ready to be controlled by Motion Control instructions
Error	An axis error occurred. Refer to the ErrorID and ErrorInfo parameters in the Motion Control instructions for details
Homed	When a system is Homed, it means that it moved to a known and predetermined position, usually at the start or end of its range of motion. This process allows the system to accurately track and control its position during subsequent operations

15.2.5 Configuring the Cam

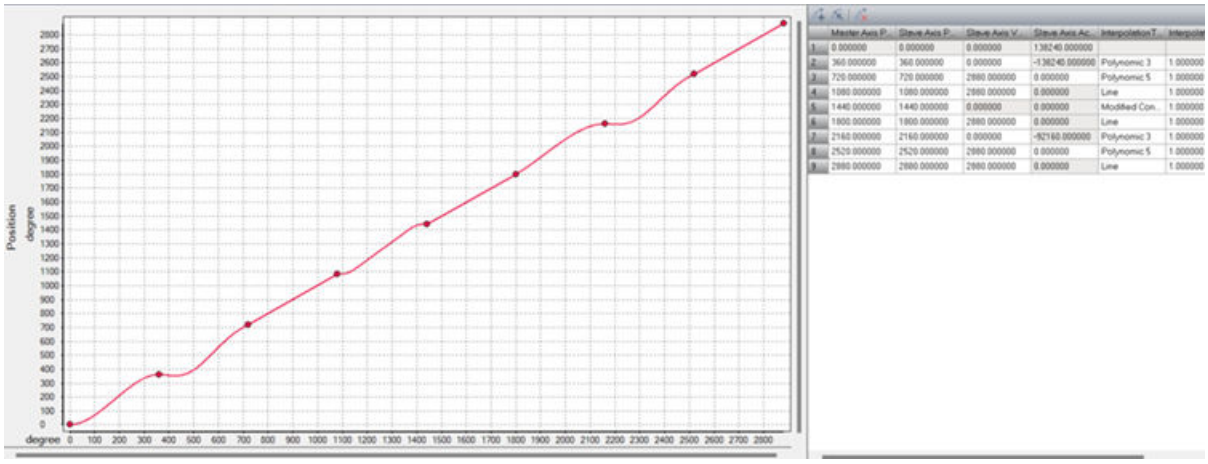
Compared to traditional cam mechanisms, the use of an electronic cam offers greater flexibility, precision and efficiency. An electronic cam controls the slave axis in synchronization with the master axis, using pulse trains, according to a pre-defined electronic cam table.

15.2.5.1 Configuring the cam table

The cam table is a data table used for specifying the positions of slave axis that move in conjunction with the movements of the master axis. The horizontal axis represents the phase of the master axis, while the vertical axis represents the position of the slave axis.

The cam table sets the master axis phase and the slave axis position. The master axis is always in the absolute coordinate system. The slave axis position is always in the relative coordinate system.

When key points are entered into the cam table and curves are set between these key points, such as a line or a polynomial curve of degree 3, a curved line connects all these points as depicted on the left. This results in an operating curve for the slave axis.



Add or delete a key-point

Note the following when adding key-point:

- It is not allowed to delete the first row or add row above it. Which means the phase of the master axis and the offset of the slave axis for the first point are set to 0 by default and cannot be changed.
- You must add at least 2 rows of data for a cam table.
- The master axis phase must be in ascending order.
- The position of the last point on the master axis and the cycle together determine the operating speed of the master axis. The cycle can be set in the parameter panel.
- The added point is the middle point between the two rows, for example, if a row is added in the middle of 0 to 1800, the position of the added row is 900 points.
- If a point is added below the last row, the key-point in the last row is considered as the middle point, and the difference between the two preceding rows are added above the last row. For example, if a row is added below 0 and 1800, the position of the added row is 3600.

Add row below current location

Add row above current location

Delete the current row

You can set cam key-point in the key-point configuration table according to application requirements.

Note the following rules when configuring the Cam:

Quantity limitation	Description
Maximum number of cams allowed to be added for each PLC	8
Maximum key-points per cam table support	361
Maximum number of key points added for all cam tables	1444

Set value for a key-point

To change the value of a master axis phase, slave axis position, slave axis velocity, slave axis acceleration, interpolation type or interpolation interval, double-click the cell you want to edit and enter the new value.

Note

The first line of the cam table cannot be edited.

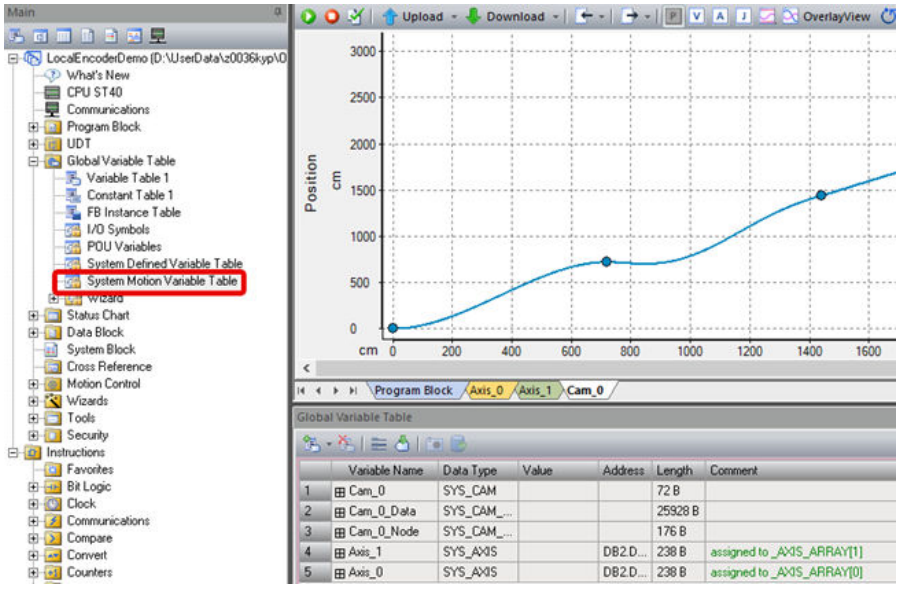
Parameter	Description
Master axis phase	Set the phase of master axis
Slave axis position	Set the position of slave axis
Slave axis velocity	• When the interpolation type is constant, or modified constant velocity, slave velocity is set to 0 and cannot be edited. • When the interpolation type is line, slave velocity is created automatically when selected. • When the interpolation type is cubic polynomial or quintic polynomial, you can edit it manually.
Slave axis acceleration	When the interpolation type is quintic polynomial, you can edit it manually.
Interpolation type	To set the interpolation type of a key-point, click the down arrow in the corresponding cell and select the interpolation type from the drop-down list. Curve type: constant, line, cubic polynomial, quintic polynomial, modified constant velocity
Interpolation interval	Maximum number of points per cam: 65535 Interpolate at each interval according to primary position.

Variables

When you configure a cam, the following three variables are created in the global variable table: "Cam_x", "Cam_x_Data", "Cam_x_Node".

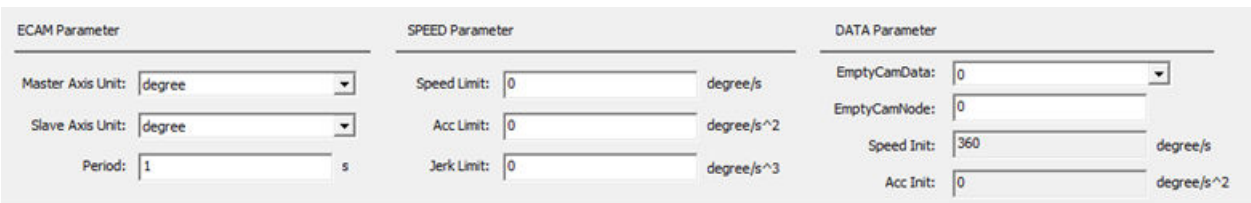
Note

The values of "Cam_x_Data" and "Cam_x_Node" can be changed by program segments or status chart.



15.2.5.2 Set the parameters

You can click from toolbar to show or hide the area for parameter setting.



ECAM Parameter

Master Axis Unit	Click the down arrow and select the unit from the drop-down list. Only affects the display of the horizontal axis unit of P V A J and does not check whether the unit of the bound master axis is the same as that set here during `CamIn`.
Slave Axis Unit	Click the down arrow and select the unit from the drop-down list. Only affects the display of the vertical axis unit of P V A J , and does not check whether the unit of the bound slave axis is the same as that set here during `CamIn`.
Cycle time	Cycle time allows you to simulate and determine the characteristics of the slave axis velocity profile and its maximum value under the current cycle. This helps you determine whether any velocity limit violations occurred. **Note**: The precondition here is that the master axis is in uniform motion.

Velocity Parameter

Velocity limit	• The velocity limit includes two ranges, positive and negative. For example, if you set a velocity limit of 10 mm/s, any velocity within positive or negative 10 mm/s is within the velocity limit. However, if the velocity exceeds 10 mm/s or is less than -10 mm/s, it is outside the velocity limit. • This velocity limit setting only affects the range that is marked as the velocity limit on the characteristic curve. Instead, it uses the velocity limitation that is specified in the individual axis configuration of the slave axis.
Acc Limit	• The acceleration limit includes two ranges, positive and negative. • This acceleration limit setting only affects the range that is marked as the acceleration limit on the characteristic curve. Instead, it uses the acceleration limitation that is specified in the individual axis configuration of the slave axis.
Jerk Limit	• The jerk limit includes two ranges, positive and negative. • This jerk limit setting only affects the range that is marked as the jerk limit on the characteristic curve. Instead, it uses the jerk limitation that is specified in the individual axis configuration of the slave axis.
Data parameter	
EmptyCamData	You can predefine the amount of empty data here, and this predefined data are included in the Cam_Data table.
EmptyCamNode	You can predefine the amount of empty node here, and this predefined data are included in the Cam_Node table.
Velocity Init	• This is the velocity for the first point. • Changing the settings of key points (including position, interpolation types, velocity, acceleration) might affect the initial velocity setting here. • This setting does not affect the initial velocity of the slave axis when the PLC actually runs this cam curve.
Acc Init	• This is the acceleration for the first point. • Changing the settings of key points (including position, interpolation types, velocity, acceleration) might affect the initial acceleration setting here. • This setting does not affect the initial acceleration of the slave axis when the PLC actually runs this cam curve.

15.2.5.3 View the curve

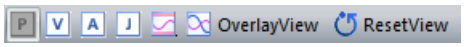
Zoom/Move the curve

Zoom in the curve: You can use the left mouse button to scroll and select a specific area to zoom in. You can also hold down CTRL and scroll the mouse wheel to zoom in or out.

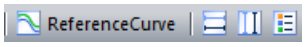
Move the curve: You can move the curve around by clicking and holding down the right mouse button in the curve area.

Tools for viewing curves

You can use the tools in the toolbar to view the curve.



	Curve showing the relationship between the phase of the master axis and the position of the slave axis. The curve cannot be hidden from view.
	Show or hide the graph of the relationship between the master axis phase and the slave axis velocity.
	Show or hide the graph of the relationship between the master axis phase and the slave axis acceleration.
	Show or hide the graph of the relationship between the master axis phase and the slave axis jerk.
	Show or hide limit curves.
	Overlay view: display all curves in the same graph.
	Reset view: you can use the left mouse button to scroll and select a specific area to zoom in. To return to the previous view, click the Reset View button.



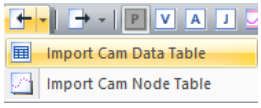
	Show or hide reference curves.
	Show or hide horizontal reference lines.
	Show or hide vertical reference lines.
	Show or hide vertical legend.

15.2.5.4 Import/Export cam table

STEP7 Micro/WIN SMART V3 allows you import or export the cam configuration or cam data.

Import cam table

1. To import cam configuration or cam data, click the down arrow beside the import button, and then select the specific table you want to import.



Note

Note the following when import cam table:

- **Update Rules**: Except for the node names, all other fields will overwrite the corresponding data in the existing table.
- **Value Validity Check**: The system automatically verifies the validity of the imported data.
- **Main axis Phase Incremental**: Ensure that the main axis phase values are in incremental order.
- **Skip Rows Not Meeting Requirements**: Any rows that do not meet the above requirements will be ignored and not imported into the system.
- **Point Count Calculation and Limits**: The number of imported points and the number after interpolation are dynamically calculated based on the actual imported nodes. Each table has a maximum point limit of 361, and the total number of points across all tables must not exceed 1444.
- **Handling Empty Data**: For empty data entries, the system will automatically select the nearest rounded-down value from the dropdown list.

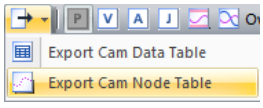
Note

Note the following when import cam data table:

Update Plot Area: After importing, a new reference curve will be added to the plot area.
Total Length Limit: The total length of the imported data table must not exceed 65535.

Export cam table

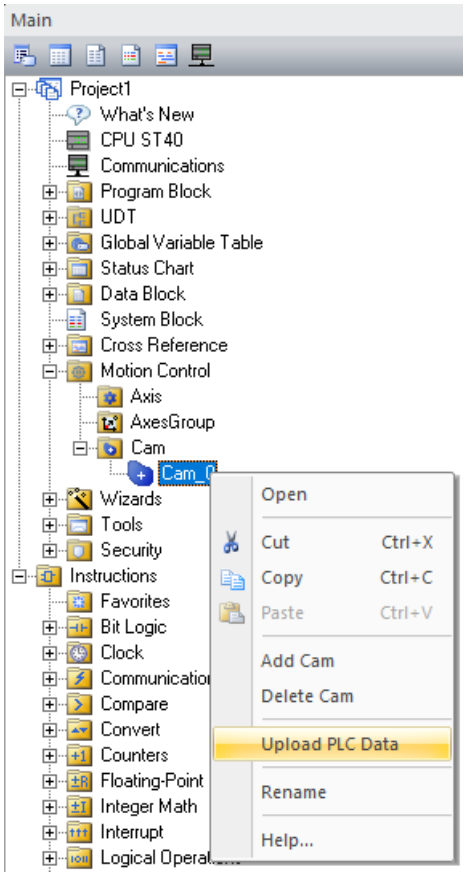
1. To export cam configuration or cam data, click the down arrow beside the export button, and then select the specific table you want to import.



Upload Cam table from PLC

STEP7 Micro/WIN SMART V3 allows you export the corresponding cam from the PLC as a CSV file, and simultaneously displays the cam curve from the CPU as a reference curve in the cam editor's graphical interface.

1. Connect to the PLC.
2. Click the Cam_x, then select "Upload PLC Data" in the context menu.



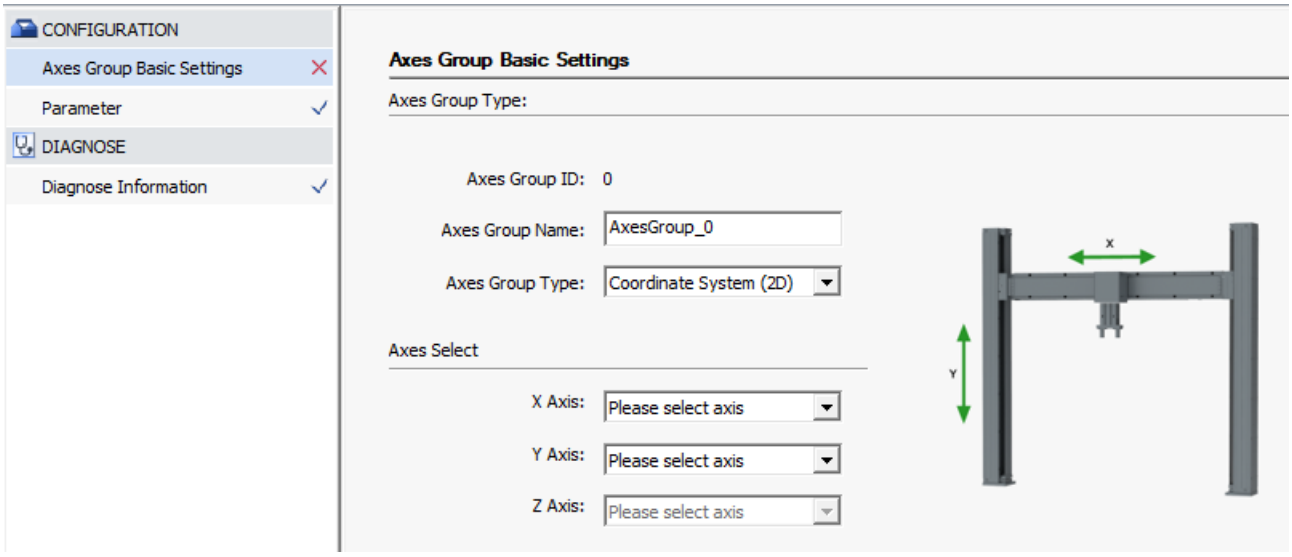
15.2.6 Configuring Axes Group

15.2.6.1 Axes Group Basic Setting

Axes Group Information

After an axes group is added, the "Axes Group Basic Settings" dialog displays as shown in the following figure.

You can configure a custom name for the newly added axes group. The default name is "AxesGroup_x", where "x" is equal to the axes group created in the numeric order.



Note

- The axes group name can contain up to 23 characters, including letters, numbers, Chinese characters, underscores and extended characters (ASCII 128 to ASCII 255). The first character is restricted to alpha or underscore. It is illegal to use keywords as variable names, or to use names that begin with a number, contain characters that are not alphanumeric or in the extended character set, or are duplicate with Instruction name or keyword. Axes group name cannot be duplicated or same with existing variable name.
- You can create a maximum of 2 Axes Group in Motion Control.
- You can reuse the same axis across different axes groups.

Axes Group Type

For the Axes Group type, select the appropriate dimensionality:

- 2D Axes Group: Used for planar motion control, such as X-Y plane
- 3D Axes Group: Used for spatial motion control, such as X-Y-Z coordinate system

Axes Select

Select individual axes to include in the axes group. Both PTO axes and virtual axes can be selected for the axes group.

Only supports linear axis, and the units of axis within the axes group must be consistent.

Parameter

Set the parameters of the Axes Group, such as maximum velocity, maximum acceleration, maximum deceleration.

15.2.6.2 Diagnose information

Command

The diagnose function can be enabled to monitor the most important status and error messages for the configured axes group.

You can enable the motion control diagnose process by click on the "Online" button, and disable by click on the "Offline" button.

Axes group status

The status of the axes group is shown in the dialog. The details would be displayed in "ErrorCode (Page 869)" dialogs.

Status message - axes group	Description
Error	An axis error occurred. Refer to the ErrorID and ErrorInfo parameters in the Motion Control instructions for details
Disabled	The axes group is disabled and cannot be controlled via Motion Control instructions
Standby	The axes group is in Standby state
Moving	The axes group is in Moving state
Stopping	The axes group is in Stopping state
ErrorStop	The axis is experiencing an error condition and is currently in the ErrorStop state

15.3 Commissioning with control panel(For PTO axis only)

You can use the "Control Panel" to help you develop the motion control function of STEP 7-Micro/WIN SMART V3. Through this interface, you can easily debug, operate and monitor the working status of STEP 7-Micro/WIN SMART V3. They can check whether the control system wiring is correct, adjust the motion control parameter configuration and test each predefined motion curve.

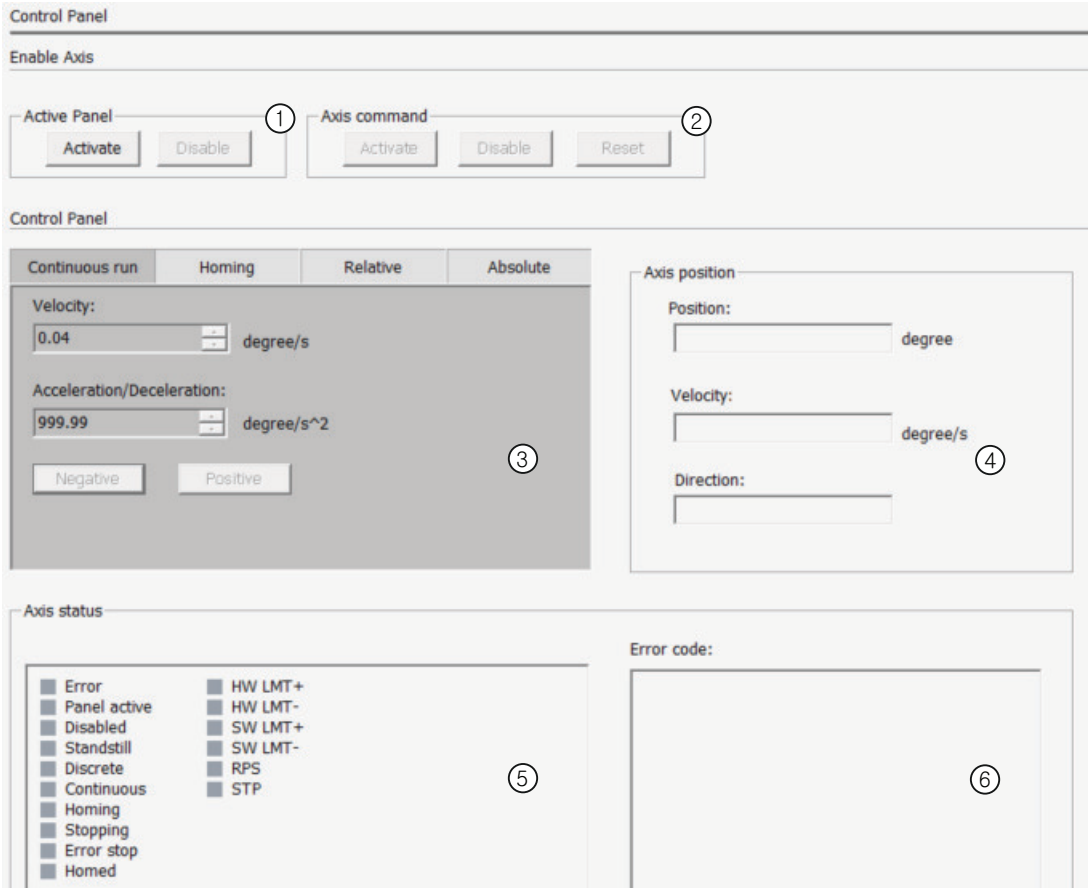
Precondition

Before using the Control Panel, please ensure that the following operations have been completed:

- Downloaded the project to the CPU. Otherwise, the CPU do not have the required valid program to operate.
- The CPU operating state is set to "STOP".

Open control panel

1. Click the tab of the axis you want to debug at the bottom of the program editor.
2. Click control panel.



①
- Activate: establish an online connection to the CPU and take over master control for the PTO axis.
- Disable: disable the axis control in the Control Panel.

②
- Activate: enable the axis, change the axis status from "Disabled" to "StandStill".
- Disable: disable the the axis, change the axis status from "StandStill" to "Disabled".
- RESET: reset the axis status machine from the state "ErrorStop" to "StandStill" or "Disabled".

③ Control panel

④ Real-time axis moving status

⑤ Show the current axis status

⑥ Show the error codes (Page 866) that occur while the axis is in motion.

Activate the control panel

Click the "Activate" button in ① to activate the control panel. If the specified activation conditions are not satisfied, a dialog box will appear, providing step-by-step instructions on the necessary actions to be taken.

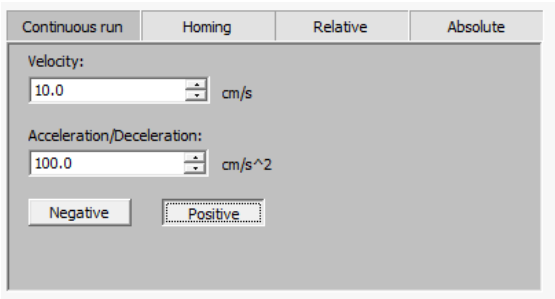
- If no CPU is connected, a pop-up window prompts you to connect the CPU.
- If the CPU is running, a pop-up window prompts you to set the operating state of the CPU to "STOP".
- If the program that is currently open is not the same as the program in the CPU, you will be prompted to download it again.

Control panel

Click the Activate button in ② to activate the motor drive, the axis state will change from "Disabled" to "Standstill".

You can configure and test the "Continuous run", "Homing", "Relative", and "Absolute" motions. The real-time position, velocity and direction of the axis can be monitored from the "Axis position" dialog.

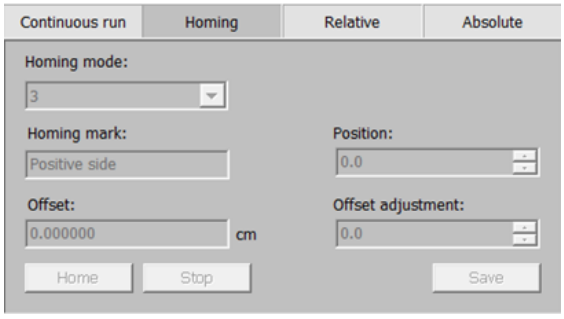
Continuous run



Enter the velocity[1] and the acceleration/deceleration [1]

- Click and hold "Positive" to execute the instruction to move at continuous speed in the positive direction.
- Click and hold "Negative" to execute the instruction to move at continuous speed in the negative direction.

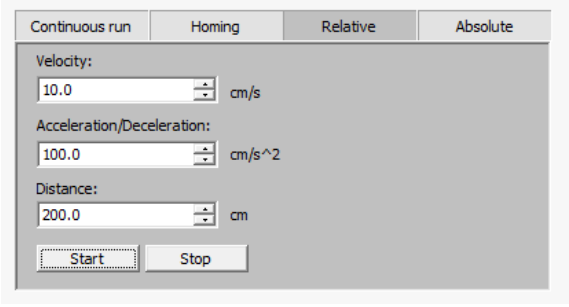
Homing



Homing mode/Homing mark/Offset are set in Homing (Page 793). You can adjust the Offset and click "Save".

- Click "Home", the triggered MC_Home instruction passes the value of off-set_adjust to the firmware for debug-ging.
- Click "Stop" to stop moving back to home position.

Relative
Absolute



Enter the velocity[1] and the acceleration/deceleration [1]

Set the distance[1] for relative movement and position[1] for absolute moment

Click "Start" to start the movement. Click "Stop" to stop the movement.

[1] can be increased or decreased to an accuracy of 0.01 using Spin.

Note

When an axis is in motion (Continuous Run\Homing\Relative\Absolute), if the CPU disconnects from STEP 7-Micro/WIN SMART V3 for more than five seconds, the axis enters the ErrorStop state. When CPU reconnects to STEP 7-Micro/WIN SMART V3, you will see error 0x1090 in the Control Panel.

15.4 Motion control instructions

15.4.1 Overview of variable

The new motion control in STEP 7-Micro/WIN SMART V3 follows the IEC 61131-3 standard. In motion control related programs, data such as parameter settings and status information are handled as variables.

Type of motion control system variable

Axis variables are system-defined variables composed of monitoring information such as axis parameters, current position, and error information for some MC function modules. When creating an axis through STEP 7-Micro/WIN SMART V3, the axis variables are registered in the global variable table in the order they are created.

The data type of the axis variables is a structure type, and the data name is "Axis_x".

- Axis variables
  Variables used to handle slave servo drives, encoder input terminals, position interface units, as well as virtual servo drives and encoder inputs.
  - STEP 7-Micro/WIN SMART V3 created variable names: Axis_0 ~ Axis_15

Note

Use MC type array to set MC instructions

When using a "FOR" loop to execute MC instructions, make sure you use an MC type array to set these instructions.

Because each MC instruction needs to be associated with an instance. If you do not use an MC type array for the setting, the program's execution results can be unpredictable.

To ensure the correct operation of the program, be sure to use an MC type array to set your MC instructions.

- Cam variables
  Variables used to handle the relationship between the position of a master axis and a slave axis.
  - STEP 7-Micro/WIN SMART V3 created variable names: Cam_0 ~ Cam_7, Cam_0_Data ~ Cam_7_Date, Cam_0_Node ~ Cam_7_Node

In the program, you can use system-defined variable names created by STEP 7-Micro/WIN SMART V3.

Example of motion control variable table

Global Variable Table

	Name	Data Type	Value	Address	Size	Comment
1	Axis_1	SYS_AXIS		DB2.D...	238 B	assigned to _AXIS_ARRAY[1]
2	Axis_0	SYS_AXIS		DB2.D...	238 B	assigned to _AXIS_ARRAY[0]
3	Version	BYTE	0	DB2.D...		
4	Axis_ID	BYTE	0	DB2.D...		
5	Config_State	BOOL	ON	DB2.D...		
6	Error_Code	WORD		DB2.D...		
7	Axis_Status	SYS_AXIS_STATUS		DB2.D...		
8	Axis_CMD_Status	SYS_AXIS_CMD_STATUS		DB2.D...		
9	Setting_Info	SYS_AXIS_PARAM		DB2.D...		
10	Actual_Info	SYS_AXIS_PARAM_ACTUAL		DB2.D...		
11	Following_Error	REAL		DB2.D...		
12	Probe0_Status	SYS_AXIS_PROB_STATUS		DB2.D...		
13	Probe1_Status	SYS_AXIS_PROB_STATUS		DB2.D...		
14	Axis_Panel	SYS_AXIS_PANEL		DB2.D...		
15	Axis_Config	SYS_AXIS_CONFIG		DB2.D...		
16	Reserved	DWORD		DB2.D...		

Variable Table 1 | Constant Table 1 | I/O Symbols | POU Variables | System Defined Variable Table | DAT0_SYM | FB Instance Table | System Motion Variable Table

Global Variable Table | Status Chart | Data Block | UDT

Motion control system variable properties

Motion control system variables share these characteristics:

Properties	Properties of motion control system variables
Global or Local	Global
R/W access	Cam_Node and Cam_Data are read-write; all other parameters are read-only
Retention	No

15.4.2 Motion control system variable

Axis variable hierarchy and name change example

Axis variables and Cam variables are system-defined variables. When these variables are used in user programs, they are referenced by the system-defined variable names (AX_x, Cam_x, Cam_x_Data, Cam_x_Node).

The following explanation uses AX_0 as an example. The same applies to other axis variables.

AX_0	Axis variable
AX_0.Axis_Status	The "axis status" level
AX_0.Axis_Status.Enabled	• Encoder axis enable status: 1 for PLC RUN, 0 for STOP
AX_0.Axis_Status.Error	1 - if error occurs in the axis

AX_0.Axis_Status.ControlPanelActive	1 - if axis control panel is activated
...	
AX_0.Axis_Status.Homed	Set to 1 when the axis is homed
AX_0.Axis_CMD_Status	The "axis command status" level
AX_0.Axis_CMD_Status.Power_CMD	
...	
AX_0.Setting_Info	The "axis setting information" level
AX_0.Setting_Info.Pos	
...	
AX_0.Actual_Info	The "axis actual information" level
AX_0.Actual_Info.Pos	
...	
AX_0.Driver_Status	The "driver status" level
AX_0.Driver_Status.In_Operation	
...	
AX_0.Probe0_Status	The "probe0 status" level
AX_0.Probe0_Status.Enable	
...	
AX_0.Probe1_Status	The "probe0 status" level
AX_0.Probe1_Status.Enable	
...	
AX_0.Axis_Panel	The "axis panel" level
AX_0.Axis_Panel.Target_Pos	
...	
AX_0.Axis_Config	The "axis configuration" level
AX_0.Axis_Config.Name	
...	
AX_0.Axis_Config.PositionLmt_SW	The "axis software position limit configuration " level
AX_0.Axis_Config.PositionLmt_SW.Maxlimit_En	
...	

Note: If you changed the axis name, for example from "Axis_0" to "My_Axis_0", the name of this variable is changed accordingly in the global variable table.

15.4.3 Basic Instructions

15.4.3.1 MC_Power

This instruction sets the axis to an operation state. You must trigger subsequent instructions based on the state of this instruction.

The subsequent motion instructions should be triggered based on the status of the MC_Power.

Table 15-2 MC_Power

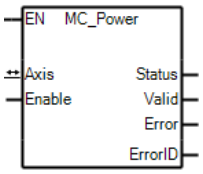
LAD / FBD	STL	Description
	**FBCALL MC_Power, INSTANCE, Axis, Enable, Status, Valid, Error, ErrorID**	Switch the axis to operational status

Table 15-3 Parameters for MC_Power

Parameter	Declaration	Data type	Initial value	Description
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Enable	IN	BOOL	-	• True - Power on • False - Power off
Status	OUT	BOOL	False	Becomes TRUE upon entry into the operational state
Valid	OUT	BOOL	False	If true, a valid sets of output is available at the function block
Error	OUT	BOOL	False	Certain error occurred with in the function block
ErrorID	OUT	WORD	0	Error code (Page 866)

Note

1. Requirements
   – Axis must be configured correctly
   – There is no enable-inhibiting error
2. Override response
   – The execution of "MC_Power" (including “enable” and “disable”) cannot be aborted or buffered by a Motion Control instruction
   – Disabling the axis (input parameter "Enable" = FALSE) aborts all Motion Control instructions for the associated axis immediately
3. Others
   – Disabling the axis while axis is at error stop state will keep the error stop state

15.4.3.2 MC_TouchProbe

Table 15-4 MC_TouchProbe

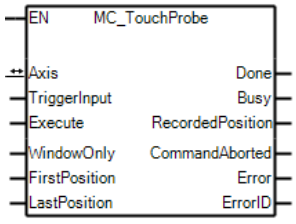
LAD / FBD	STL	Description
	**FBCALL MC_TouchProbe, INSTANCE, Axis, TriggerInput, Execute, WindowOnly, FirstPosition,LastPositio n,Done, Busy, RecordedPosition, CommandAborted, Error, ErrorID**	This instruction configures the touch probe.

Table 15-5 Parameters for MC_TouchProbe

Parameter	Declaration	Data type	Initial value	Description
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
TriggerInput	IN	MC_TRIG-GER_REF	-	Reference to the trigger signal source
Execute	IN	BOOL	-	Start the instruction at rising edge
WindowOnly	IN	BOOL	False	If set to TRUE, only trigger events within the window (as defined below) are valid.
FirstPosition	IN	REAL	0.0	Start position from where (positive direction) trigger events are accepted. Value included in window
LastPosition	IN	REAL	0.0	Stop position of the window. Value included in window
Done	OUT	BOOL	False	Trigger event recorded
Busy	OUT	BOOL	False	The instruction is not finished
RecordedPosi-tion	OUT	REAL	0.0	Capture the position where trigger event occurred
CommandA-borted	OUT	BOOL	False	Instruction is aborted by another command (for example, MC_AbortTrigger)
Error	OUT	BOOL	False	Signals that an error has occurred within the instruction
ErrorID	OUT	WORD	0	Error code (Page 866)

Note

1. Requirements
   - The axis has been configured correctly.
   - TriggerInput must be configured correctly. First add a variable for TriggerInput in the variable table, then set its data type as MC_TRIGGER_REF.
     - Associate ProbeID = 0: associated with axis configuration probe0
     - Associate ProbeID = 1: associated with axis configuration probe1
     - TriggerEdge = 0: rising edge of the probe is valid
     - TriggerEdge = 1: falling edge of the probe is valid
2. Override response
   - The MC_TouchProbe instruction can be aborted by the following Motion Control instructions:
     MC_ AbortTrigger
   - The MC_TouchProbe instruction cannot be buffered by any Motion Control instruction.
   - The new MC_TouchProbe instruction does not abort or buffer to any other active Motion Control instructions.

Note

Using MC_TouchProbe Command

- If two MC_TouchProbe commands have the same axis number and probe ID, the latter command interrupts the former command.
- You can configure a maximum of two probes, with the range [I0.0~I1.7] for a single axis. These probes must not conflict with HSC, INT, Limit, STP, RPS, ZP or any other probe inputs that are already in use.
- When multiple axes execute probe commands simultaneously, they do not interfere with each other.

15.4.3.3 MC_AbortTrigger

Table 15-6 MC_AbortTrigger

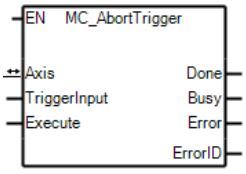
LAD / FBD	STL	Description
	**FBCALL MC_AbortTrigger, INSTANCE, Axis, TriggerInput, Execute,Done, Busy, Error, ErrorID**	This instruction is used to abort instructions which are connected to trigger events (e.g. MC_TouchProbe).

Table 15-7 Parameters for MC_AbortTrigger

Parameter	Declaration	Data type	Initial value	Description
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
TriggerInput	IN_OUT	MC_TRIG-GER_REF	-	Reference to the trigger signal source.
Execute	IN	BOOL	-	Aborts trigger event at rising edge
Done	OUT	BOOL	False	Trigger functionality aborted
Busy	OUT	BOOL	False	The instruction is not finished
Error	OUT	BOOL	False	Signals that an error has occurred within the instruction
ErrorID	OUT	WORD	0	Error code (Page 866)

Note

Requirement:

- The axis has been configured correctly.
- There is a MC_TouchProbe instruction running on the axis.

15.4.3.4 MC_Reset

Table 15-8 MC_Reset

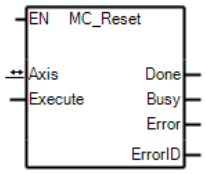
LAD / FBD	STL	Description
	**FBCALL MC_Reset, INSTANCE, Axis, Execute, Done, Busy, Error, ErrorID**	This instruction makes the transition from the state **ErrorStop** to **Standstill** or **Disabled** by re-setting all internal axis-related errors – it does not affect the output of the function block in-stances.

Table 15-9 Parameters for MC_Reset

Parameter	Declaration	Data type	Initial value	Description
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Execute	IN	BOOL	-	Start the instruction at rising edge
Done	OUT	BOOL	False	**Standstill** or **Disabled** state is reached
Busy	OUT	BOOL	False	The instruction is not finished
Error	OUT	BOOL	False	Signals that an error has occurred within the instruction
ErrorID	OUT	WORD	0	Error code (Page 866)

Note

1. Requirements
   - The axis has been configured correctly.
   - Axis is at ErrorStop state.
2. Override response
   - The MC_Reset instruction cannot be aborted or buffered by any other Motion Control instruction.
   - The new MC_Reset instruction does not abort or buffer to any other active Motion Control instructions.

15.4.3.5 MC_Home

Table 15-10 MC_Home

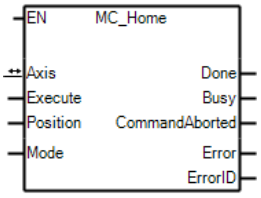
LAD / FBD	STL	Description
	**FBCALL MC_Home, INSTANCE, Axis, Execute, Position, Mode, Done, Busy, CommandAborted, Error, ErrorID**	This instruction set or search a home point for an axis.

Table 15-11 Parameters for MC_Home

Parameter	Declaration	Data type	Initial value	Description
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Execute	IN	BOOL	-	Start instruction at rising edge
Position	IN	REAL	0.0	Absolute position after the instruction is done
Mode	IN	BYTE	-	Homing mode: • 0: Direct home (Absolute) • 1: Direct home (Relative) • 3: Active home
Done	OUT	BOOL	False	Home point is set or found successfully
Busy	OUT	BOOL	False	The instruction is not finished
CommandA-borted	OUT	BOOL	False	The instruction is aborted by another instruction
Error	OUT	BOOL	False	Signals that an error has occurred within the instruction
ErrorID	OUT	WORD	0	Error code (Page 866)

Note

For a modulo axis, MC_Home convert the value on the Position pin to fit within the modulo range, and then set it as the axis position.

Note

1. Modes
   - 0 – Direct home (Absolute): The current axis position is set to the value of parameter "Position".
   - 1 – Direct home (Relative): The current axis position is offset by the value of parameter "Position".
   - 3 – Active home: The homing procedure is executed automatically.
2. Requirements
   - The axis has been configured correctly.
   - The axis is enabled.
   - The MC_Home instruction can only execute while the axis is at standstill or disabled in Mode 0 and 1.
   - The MC_Home instruction can only execute while the axis is at standstill in Mode 3.
   - Before executing Active home, you must configure the parameters correctly in the axis "Homing" page.
3. Override response
   - The MC_Home instruction cannot be aborted by any other Motion Control instruction except MC_Power (disabling), MC_Stop and MC_ImmediateStop.
   - The MC_Home instruction cannot be buffered by any Motion Control instruction.
   - The MC_Home instruction cannot abort any Motion Control instruction.
   - The new MC_Home instruction cannot buffer to any Motion Control instruction.

15.4.3.6 MC_Stop

Table 15-12 MC_Stop

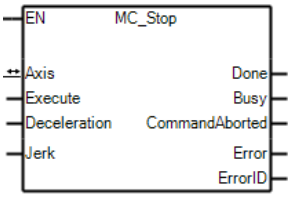
LAD / FBD	STL	Description
	**FBCALL MC_Stop, INSTANCE, Axis, Execute, Deceleration, Jerk, Done, Busy, CommandAborted, Error, ErrorID**	This instruction commands the axis to decelerate to a stop and transfers the axis to the state **Stopping**. It aborts any ongoing instruction execution. While the axis is in state **Stopping**, no other instruction can perform any motion on the same axis. After the axis has reached **Velocity** zero, the **Done** output is set to TRUE immediately. The axis remains in the state **Stopping** as long as **Execute** is still TRUE or **Velocity** zero is not yet reached. As soon as **Done** is **TRUE** and **Execute** is FALSE the axis goes to state **Standstill**.

Table 15-13 Parameters for MC_Stop

Parameter	Declaration	Data type	Initial value	Description
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Execute	IN	BOOL	-	Start the instruction at rising edge
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	Value of the **Deceleration**
Jerk	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	Reserved
Done	OUT	BOOL	False	Zero velocity reached
Busy	OUT	BOOL	False	The instruction is not finished
CommandAborted	OUT	BOOL	False	**Command** is aborted by another command
Error	OUT	BOOL	False	Signals that an error has occurred within the instruction
ErrorID	OUT	WORD	0	Error code (Page 866)

Note

1. Requirements
   - The axis has been configured correctly.
   - The axis is enabled.
2. Override response
   - The MC_Stop instruction can be aborted by the following Motion Control instructions:
     MC_ImmediateStop
   - The MC_Stop instruction cannot be buffered by any Motion Control instruction.
   - The new MC_Stop instruction aborts the following active Motion Control instructions:
     MC_Home
     MC_Halt
     MC_MoveAbsolute
     MC_MoveRelative
     MC_MoveVelocity
     MC_MoveJog
     MC_CamOut
     MC_GearOut
     MC_CamIn
     MC_GearIn
     MC_PhasingRelative
   - The new MC_Stop instruction cannot buffer to any Motion Control instruction.

15.4.3.7 MC_Halt

Table 15-14 MC_Halt

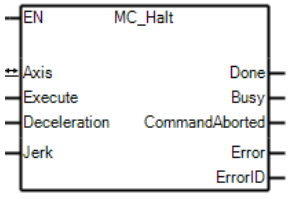
LAD / FBD	STL	Description
	**FBCALL MC_Halt, INSTANCE, Axis, Execute, Deceleration, Jerk, Done, Busy, CommandAborted, Error, ErrorID**	This instruction commands the axis to decelerate to a stop. The axis enters to the state **DiscreteMotion**, until the **Velocity** is zero. With the **Done** output set, the state is transferred to **Standstill**.

Table 15-15 Parameters for MC_Halt

Parameter	Declaration	Data type	Initial value	Description
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Execute	IN	BOOL	-	Start instruction at rising edge
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	Value of the **Deceleration**

Parameter	Declaration	Data type	Initial value	Description
Jerk	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	Reserved
Done	OUT	BOOL	False	Zero velocity reached
Busy	OUT	BOOL	False	The instruction is not finished
CommandAborted	OUT	BOOL	False	**Command** is aborted by another command
Error	OUT	BOOL	False	Signals that an error has occurred within the instruction
ErrorID	OUT	WORD	0	Error code (Page 866)

Note

1. Requirements
   - The axis has been configured correctly.
   - The axis is enabled.
2. Override response
   - The MC_Halt instruction can be aborted by the following Motion Control instructions:
     MC_Halt
     MC_Stop
     MC_ImmediateStop
     MC_MoveAbsolute
     MC_MoveRelative
     MC_MoveVelocity
     MC_MoveJog
     MC_CamIn
     MC_GearIn
   - The MC_Halt instruction can be buffered by the following Motion Control instructions:
     MC_MoveAbsolute
     MC_MoveRelative
     MC_MoveVelocity
     MC_CamIn
     MC_GearIn
   - The new MC_Halt instruction aborts the following active Motion Control instructions:
     MC_MoveAbsolute
     MC_MoveRelative
     MC_MoveVelocity
     MC_MoveJog
     MC_CamOut
     MC_GearOut
     MC_CamIn
     MC_GearIn
     MC_PhasingRelative
   - The new MC_Halt instruction cannot buffer to any Motion Control instruction.

15.4.3.8 MC_MoveAbsolute

Table 15-16 MC_MoveAbsolute

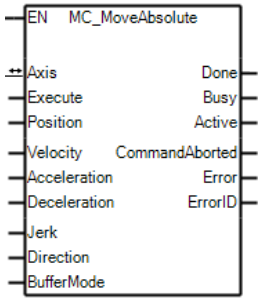
LAD / FBD	STL	Description
	**FBCALL MC_MoveAbsolute, INSTANCE, Axis, Execute, Position, Velocity, Acceleration,Deceleration , Jerk, Direction, BufferMode,Done, Busy, Active, CommandAborted, Error, ErrorID**	This instruction commands the axis to move to the specified absolute position.

Table 15-17 Parameters for MC_MoveAbsolute

Parameter	Declaration	Data type	Initial value	Description
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Execute	IN	BOOL	-	Start the instruction at rising edge
Position	IN	REAL	-	Specify the target position in absolute coordinates
Velocity	IN	REAL	-	Value of the **Velocity** (not necessarily reached)
Acceleration[1]	IN	REAL	Axis_Config. Dynamic_Default. Default_Acc	Value of the **Acceleration**
Deceleration[1]	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	Value of the **Deceleration**
Jerk[1]	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	Value of the **Jerk**
Direction[1]	IN	BYTE	0	Moving direction 0 - Positive, 1 - Negative , 2 - Shortest Distance, 3 - Current Direction
BufferMode[1]	IN	BYTE	0	Defines the chronological sequence of the instruction. 0：Aborting, 1：Buffered
Done	OUT	BOOL	False	Reach target position
Busy	OUT	BOOL	False	The instruction is not finished
Active	OUT	BOOL	False	Indicates that the instruction has control on the axis
CommandAborted	OUT	BOOL	False	**Command** is aborted by another command
Error	OUT	BOOL	False	Signals that an error has occurred within the instruction
ErrorID	OUT	WORD	0	Error code (Page 866)

[1] When you don't assign values to these parameters, the system uses the default initial values.

Note

1. Requirements
   - The axis has been configured correctly.
   - The axis is enabled.
   - The axis is homed.
2. Override response
   - The MC_MoveAbsolute instruction can be aborted by the following Motion Control instructions:
     MC_Halt
     MC_Stop
     MC_ImmediateStop
     MC_MoveAbsolute
     MC_MoveRelative
     MC_MoveVelocity
     MC_MoveJog
     MC_CamIn
     MC_GearIn
   - The MC_MoveAbsolute instruction can be buffered by the following Motion Control instructions:
     MC_MoveAbsolute
     MC_MoveRelative
     MC_MoveVelocity
     MC_CamIn
     MC_GearIn
   - The new MC_MoveAbsolute instruction aborts the following active Motion Control instructions:
     MC_Halt
     MC_MoveAbsolute
     MC_MoveRelative
     MC_MoveVelocity
     MC_MoveJog
     MC_CamOut
     MC_GearOut
     MC_CamIn
     MC_GearIn
     MC_PhasingRelative
   - The new MC_MoveAbsolute instruction can buffer to the following Motion Control instructions:
     MC_Halt
     MC_MoveAbsolute
     MC_MoveRelative

15.4.3.9 MC_MoveRelative

Table 15-18 MC_MoveRelative

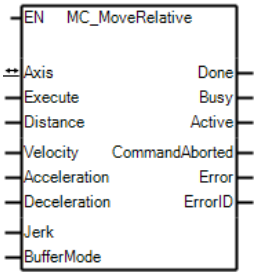
LAD / FBD	STL	Description
	**FBCALL MC_MoveRelative, INSTANCE, Axis, Execute, Distance, Velocity, Acceleration,Deceleration , Jerk, BufferMode,Done, Busy, Active, CommandAborted, Error, ErrorID**	This instruction commands the axis moves a specified distance relative to the current position.

Table 15-19 Parameters for MC_MoveAbsolute

Parameter	Declaration	Data type	Initial value	Description
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Execute	IN	BOOL	-	Start the instruction at rising edge
Distance	IN	REAL	-	Relative distance for the motion
Velocity	IN	REAL	-	Value of the **Velocity** (not necessarily reached)
Acceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Acc	Value of the **Acceleration**
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	Value of the **Deceleration**
Jerk	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	Value of the **Jerk**
BufferMode	IN	BYTE	0	Defines the chronological sequence of the instruction. 0：Aborting, 1: Buffered
Done	OUT	BOOL	False	Reach target position
Busy	OUT	BOOL	False	The instruction is not finished
Active	OUT	BOOL	False	Indicates that the instruction has control on the axis
CommandAborted	OUT	BOOL	False	**Command** is aborted by another command
Error	OUT	BOOL	False	Signals that an error has occurred within the instruction
ErrorID	OUT	WORD	0	Error code (Page 866)

Note

1. Requirements
   - The axis has been configured correctly.
   - The axis is enabled.
2. Override response
   - The MC_MoveRelative instruction can be aborted by the following Motion Control instructions:
     MC_Halt
     MC_Stop
     MC_ImmediateStop
     MC_MoveAbsolute
     MC_MoveRelative
     MC_MoveVelocity
     MC_MoveJog
     MC_CamIn
     MC_GearIn
   - The MC_MoveRelative instruction can be buffered by the following Motion Control instructions:
     MC_MoveAbsolute
     MC_MoveRelative
     MC_MoveVelocity
     MC_CamIn
     MC_GearIn
   - The new MC_MoveRelative instruction aborts the following active Motion Control instructions:
     MC_Halt
     MC_MoveAbsolute
     MC_MoveRelative
     MC_MoveVelocity
     MC_MoveJog
     MC_CamOut
     MC_GearOut
     MC_CamIn
     MC_GearIn
     MC_PhasingRelative
   - The new MC_MoveRelative instruction can buffer to the following Motion Control instructions:
     MC_Halt
     MC_MoveRelative
     MC_MoveAbsolute

15.4.3.10 MC_MoveVelocity

Table 15-20 MC_MoveVelocity

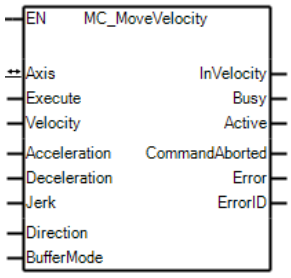
LAD / FBD	STL	Description
	**FBCALL MC_MoveVelocity, INSTANCE, Axis, Execute, Velocity, Acceleration, Deceleration, Jerk, Direction, BufferMode, Invelocity, Busy, Active, CommandAborted, Error, ErrorID**	This instruction command the axis to perform an infinite controlled motion at the specified velocity.

Table 15-21 Parameters for MC_MoveVelocity

Parameter	Declaration	Data type	Initial value	Description
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Execute	IN	BOOL	-	Start the instruction at rising edge
Velocity	IN	REAL	-	Value of the **Velocity**
Acceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Acc	Value of the **Acceleration**
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	Value of the **Deceleration**
Jerk	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	Value of the **Jerk**
Direction	IN	BYTE	0	Moving direction 0 - Positive, 1 - Negative. This pin, together with the sign of the Velocity pin, determines whether the axis moves in a positive or negative direction. For example, if Direction = 1 and Velocity = 10.0, the axis moves in the negative direction. If Direction = 1 and Velocity = -10.0, the axis moves in the positive direction.
BufferMode	IN	BYTE	0	Defines the chronological sequence of the instruction. 0：Aborting, 1: Buffered
InVelocity	OUT	BOOL	False	Commanded velocity reached
Busy	OUT	BOOL	False	The instruction is not finished
Active	OUT	BOOL	False	Indicates that the instruction has control on the axis
CommandAborted	OUT	BOOL	False	**Command** is aborted by another command

Parameter	Declaration	Data type	Initial value	Description
Error	OUT	BOOL	False	Signals that an error has occurred within the instruction
ErrorID	OUT	WORD	0	Error code (Page 866)

Note

1. Requirements
   - The axis has been configured correctly.
   - The axis is enabled.
2. Override response
   - The MC_MoveVelocity instruction can be aborted by the following Motion Control instructions:
     MC_Halt
     MC_Stop
     MC_ImmediateStop
     MC_MoveVelocity
     MC_MoveAbsolute
     MC_MoveRelative
     MC_MoveJog
     MC_CamIn
     MC_GearIn
   - The MC_MoveVelocity instruction cannot be buffered by any Motion Control instruction.
   - The new MC_MoveVelocity instruction aborts the following active Motion Control instructions:
     MC_Halt
     MC_MoveAbsolute
     MC_MoveRelative
     MC_MoveJog
     MC_CamOut
     MC_GearOut
     MC_MoveVelocity
     MC_CamIn
     MC_GearIn
   - The new MC_MoveVelocity instruction can buffer to the following Motion Control instructions:
     MC_Halt
     MC_MoveAbsolute
     MC_MoveRelative

15.4.3.11 MC_ImmediateStop

Table 15-22 MC_ImmediateStop

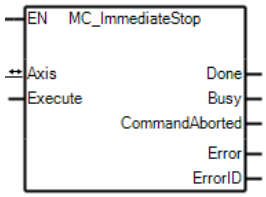
LAD / FBD	STL	Description
	**FBCALL MC_ImmediateStop, INSTANCE, Axis, Execute, Done, Busy, CommandAborted, Error, ErrorID**	This instruction commands axis immediate stop and transfers the axis to the state **Stopping**. It aborts any ongoing instruction execution. While the axis is in state **Stopping**, no other instruction can perform any motion on the same axis. After the axis has reached **Velocity** zero, the **Done** output is set to TRUE immediately. The axis remains in the state **Stopping** as long as **Execute** is still TRUE or **Velocity** zero is not yet reached. As soon as **Done** is **TRUE** and **Execute** is FALSE the axis goes to state **Standstill**.

Table 15-23 Parameters for MC_ImmediateStop

Parameter	Declaration	Data type	Initial value	Description
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Execute	IN	BOOL	-	Start the instruction at rising edge
Done	OUT	BOOL	False	Zero velocity reached
Busy	OUT	BOOL	False	The instruction is not finished
CommandAborted	OUT	BOOL	False	**Command** is aborted by another command
Error	OUT	BOOL	False	Signals that an error has occurred within the instruction
ErrorID	OUT	WORD	0	Error code (Page 866)

Note

1. Requirements
   - The axis has been configured correctly.
   - The axis is enabled.
2. Override response
   - The MC_ImmediateStop instruction cannot be aborted by any Motion Control instruction.
   - The MC_ImmediateStop instruction cannot be buffered by any Motion Control instruction.
   - The new MC_ImmediateStop instruction aborts the following active Motion Control instructions:
     MC_Home
     MC_Halt
     MC_Stop
     MC_MoveAbsolute
     MC_MoveRelative
     MC_MoveVelocity
     MC_MoveJog
     MC_CamOut
     MC_GearOut
     MC_CamIn
     MC_GearIn
     MC_PhasingRelative
   - The new MC_ImmediateStop instruction cannot buffer to any Motion Control instruction.

15.4.3.12 MC_MoveJog

Table 15-24 MC_MoveJog

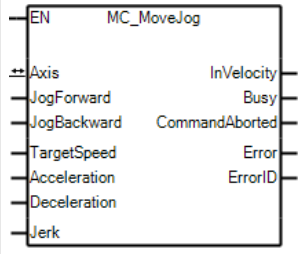
LAD / FBD	STL	Description
	**FBCALL MC_MoveJog, INSTANCE, Axis, JogForward, JogBackword, TargetSpeed, Acceleration, Deceleration, Jerk, InVelocity, Busy, CommandAborted, Error, ErrorID**	The MC_MoveJog instruction jogs an axis according to the specified target velocity.

Table 15-25 Parameters for MC_MoveJog

Parameter	Declaration	Data type	Initial value	Description
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
JogForward	IN	BOOL	-	Jogging in the forward direction, high-level triggered

Parameter	Declaration	Data type	Initial value	Description
JogBackward	IN	BOOL	-	Jogging in the reverse direction, high-level triggered
TargetSpeed	IN	REAL	-	Target velocity
Acceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Acc	Value of the **Acceleration**
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	Value of the **Deceleration**
Jerk	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	Value of the **Jerk**
InVelocity	OUT	BOOL	False	Velocity reached
Busy	OUT	BOOL	False	The instruction is not finished
CommandAborted	OUT	BOOL	False	The instruction is aborted by another instruction
Error	OUT	BOOL	False	Signals that an error has occurred within the instruction
ErrorID	OUT	WORD	0	Error code (Page 866)

Note

1. Requirements
   - The axis has been configured correctly.
   - The axis is enabled.
2. Override response
   - The MC_MoveJog instruction can be aborted by the following Motion Control instructions:
     MC_Halt
     MC_Stop
     MC_MoveJog
     MC_ImmediateStop
     MC_MoveAbsolute
     MC_MoveRelative
     MC_MoveVelocity
     MC_CamIn
     MC_GearIn
   - The MC_MoveJog instruction cannot be buffered by any Motion Control instruction.
   - The new MC_MoveJog instruction aborts the following active Motion Control instructions:
     MC_Halt
     MC_MoveAbsolute
     MC_MoveRelative
     MC_MoveVelocity
     MC_CamOut
     MC_GearOut
     MC_CamIn
     MC_GearIn
     MC_MoveJog
     MC_PhasingRelative
   - The new MC_MoveJog instruction cannot buffer to any Motion Control instructions.
   - If both JogForward and JogBackward pins are pulled up, ongoing JOG process will be aborted and axis will decelerate to standstill and an error will be reported at MC_MoveJog FB instance. Pulling down JogForward or JogBackward will make the axis moving backward or forward if JogBackward or JogForward is still pulled up.

15.4.4 Cam and Gear Instruction

15.4.4.1 MC_CamIn

Start the cam action using the specified cam table.

Table 15-26 MC_CamIn

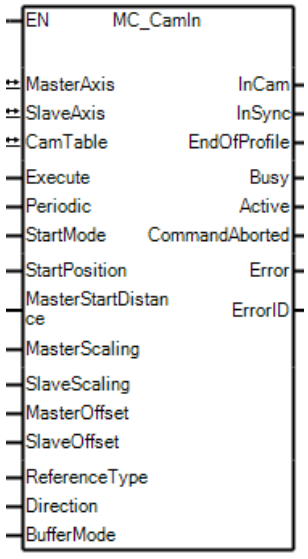
LAD / FBD	STL	Description
	**FBCALL MC_CamIn, INSTANCE, MasterAxis, SlaveAxis, CamTable, Execute, Periodic, StartMode, StartPosition, MasterStartDistance, MasterScaling, SlaveScaling, MasterOffset, SlaveOffset, ReferenceType, Direction, Buffermode, InCam, InSync, EndOfProfile, Busy, Active, CommandAborted, Error, ErrorID**	The MC_CamIn instruction starts a cam operation by using a specified cam table.

Table 15-27 Parameters for MC_CamIn

Parameter	Declaration	Data type	Initial value	Description
MasterAxis	IN_OUT	SYS_AXIS	-	Reference to a master axis
SlaveAxis	IN_OUT	SYS_AXIS	-	Reference to a slave axis
CamTable	IN_OUT	SYS_CAM	-	Reference to a cam table
Execute	IN	BOOL	-	Start the instruction at rising edge
Periodic	IN	BOOL	False	Specify whether to execute the specified cam table periodically or only once. • TRUE: Periodic • FALSE: Non-periodic
StartMode [1]	IN	WORD	0	Specify the coordinates for MasterStartDistance. • 0: Absolute position • 1: Relative position
StartPosition	IN	REAL	0.0	Specify the starting point of the cam table (0 phase)
MasterStart-Distance [1]	IN	REAL	0.0	Specify the position of the master axis when the slave axis starts the cam motion
MasterScaling	IN	REAL	1.0	Zoom in/out the phase of the master axis by the specified ratio

Parameter	Declaration	Data type	Initial value	Description
SlavingScaling	IN	REAL	1.0	Zoom in/out slave axis displacement by the specified ratio
MasterOffset [1]	IN	REAL	0.0	Offset of master table. Angular offset of the master shaft to cam
SlaveOffset [1]	IN	REAL	0.0	Offset of slave table. Sharpened cam (i.e higher elevation and deeper depression)
Reference-Type	IN	BYTE	0	Reserved Now the command position from the same task cycle is used by default
Direction	IN	BYTE	0	The slave axis cam moves when the master axis moves in the specified direction. Only if the master axis is moving opposite to the direction specified, the slave axis cam is stopped. • 0: Positive direction • 1: Negative direction • 2: No direction specified
BufferMode	IN	BYTE	0	Specify the behavior when executing more than one motion instruction 0：Aborting, 1: Buffered
InCam	OUT	BOOL	False	TRUE when the cam table start point is executed
InSync	OUT	BOOL	False	When the slave axis is synchronized with the master axis, it is true.
EndOfProfile	OUT	BOOL	False	TRUE when the cam table end point is executed
Busy	OUT	BOOL	False	The instruction is not finished
Active	OUT	BOOL	False	Indicate the instruction has control on the axis
CommandAborted	OUT	BOOL	False	Command is aborted by another command
Error	OUT	BOOL	False	Signals that an error has occurred within the instruction
ErrorID	OUT	WORD	0	Error code (Page 869)

[1] **Note**: These four parameters are now reserved and the values entered will not take effect. The default value of StartMode is 1 (Relative Mode), and the default value for others is 0.

Note

The slave axis number must be greater than the master axis number.

WARNING

Potential dangesr of sudden velocity changes in synchronized axes

If the velocity of the master axis and the auxiliary axis changes abruptly during the start or execution of a synchronous operation, the action of the slave axis will also change abruptly, which may cause excessive force on the device.

When using MC_CamIn, do not execute the MC_Home or MC_ImmediateStop instruction on the master axis. Executing the MC_Home/MC_ImmediateStop instruction on the master axis can cause the slave axis to track rapidly, which is very dangerous.

If you want to use the MC_Home/MC_ImmediateStop instruction on the master axis, release the relationship between the master and slave axes before executing this instruction.

Note

When configuring the pin Endofprofile, note the following :

- For non-periodic cams, Endofprofile is set to **TRUE** and maintained after the cam cycle.
- For periodic cams, a scan cycle is set at the end of each cam cycle.
- EndOfProfile is advanced a maximum of 24ms relative to the end of the cam table.

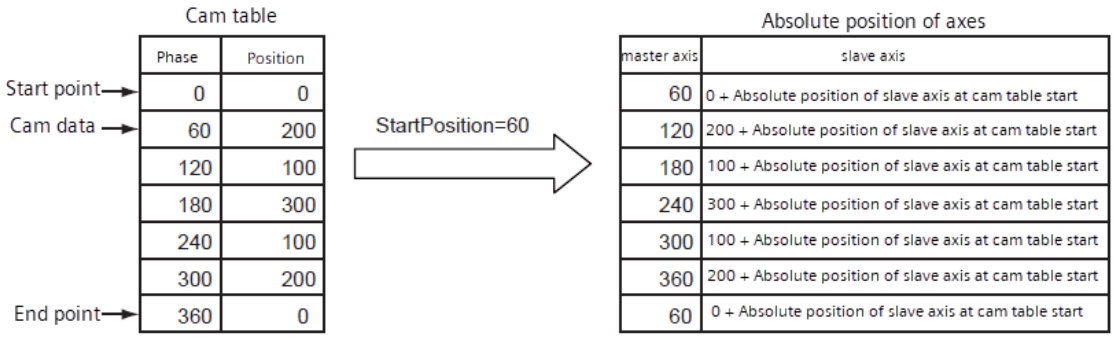
Note

1. Requirements
   - The master and slave axis have been configured correctly.
   - The slave axis is enabled.
2. Override response
   - The MC_CamIn instruction can be aborted by the following Motion Control instructions:
     MC_Halt
     MC_Stop
     MC_ImmediateStop
     MC_MoveAbsolute
     MC_MoveRelative
     MC_MoveVelocity
     MC_MoveJog
     MC_CamIn
     MC_GearIn
     MC_CamOut
   - The MC_CamIn instruction cannot be buffered by any Motion Control instruction.
   - The new MC_CamIn instruction aborts the following active Motion Control instructions:
     MC_Halt
     MC_MoveAbsolute
     MC_MoveRelative
     MC_MoveVelocity
     MC_MoveJog
     MC_CamOut
     MC_GearOut
     MC_CamIn
     MC_GearIn
     MC_PhasingRelative
   - The new MC_CamIn instruction can buffer to the following Motion Control instructions:
     MC_Halt
     MC_MoveAbsolute
     MC_MoveRelative

Start Cam movement

After the start command is issued, wait for the master axis to reach the StartPosition (the starting position of the cam table).
When the master axis passes through the StartPosition, the starting point of the cam table is executed, and the output variable InCam (cam movement in progress) becomes TRUE.

The phase and displacement of the cam table are specified as relative values from the zero point. Therefore, the absolute positions of each axis in each phase are relative values based on the absolute positions of the cam table's starting point for each axis. For example, when the counting mode of the master axis is a rotation mode of 0 ~ 360°, the StartPosition 60°. As shown below, the absolute position of the master axis is the phase of the cam table plus the value of StartPosition, and the absolute position of the slave axis is the displacement of the cam table plus the absolute position of the slave axis at cam table start.



Example: Slave axis movement varies depending on StartPosition

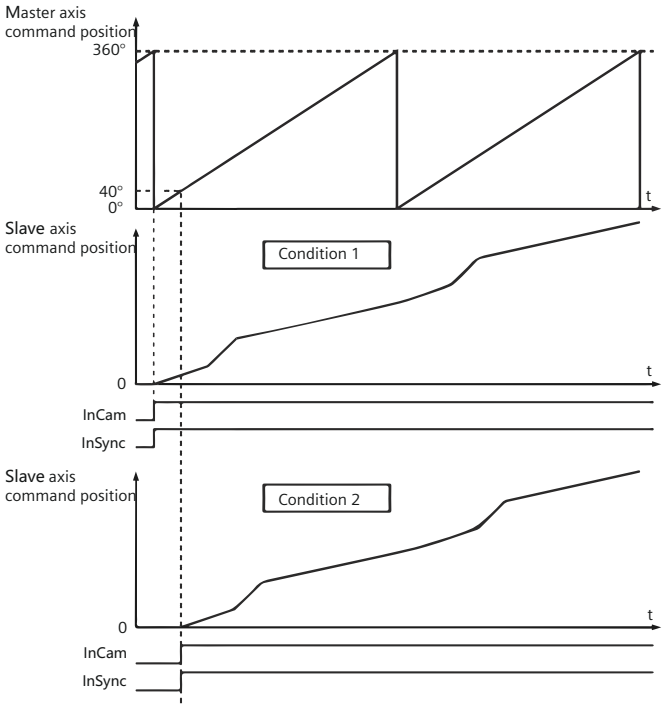
The cam table is set as follows.

Master axis	Slave axis	Connect velocity	Connect acceleration	Cam curve	Phase interval
0.000000	0.000000	360.000000	0.000000		
80.000000	80.000000	360.000000	0.000000	Line	0.100000
150.000000	220.000000	720.000000	0.000000	Line	0.100000
360.000000	360.000000	240.000000	0.000000	Line	0.100000

The starting conditions for the cam movement are as follows.

Input variable	Condition 1	Condition 2
Periodic(Repeat mode)	TRUE(Repeat)	TRUE(Repeat)
StartMode(Relative mode)	1	1
StartPosition	0	40
MasterStartDistance	0	0

- In condition 1, when the master axis passes 0°, the output variable InCam (cam movement in progress) and the output variable InSync (synchronisation in progress) both become TRUE and the cam movement starts from the axis.
- In condition 2, when the master axis passes 40° specified by StartPosition (cam table start position), the output variable InCam (cam movement in progress) and the output variable InSync (synchronisation in progress) both become TRUE and the slave axis starts the cam movement.



Periodic (Repeat mode)

When TRUE [Repeat] is set in Periodic(Repeat Mode), the cam movement is executed repeatedly from the start point to the end point of the cam table.

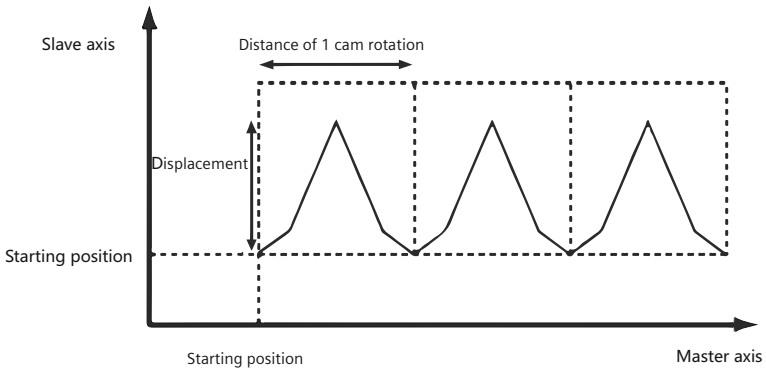
When FALSE [No repeat] is set, the cam motion is ended if execution continues to the end of the cam table.

If the travel positions of the slave axis at the start and end of the cam table are the same, the movement is performed as a reciprocating cam (see Reciprocating Cam movement).

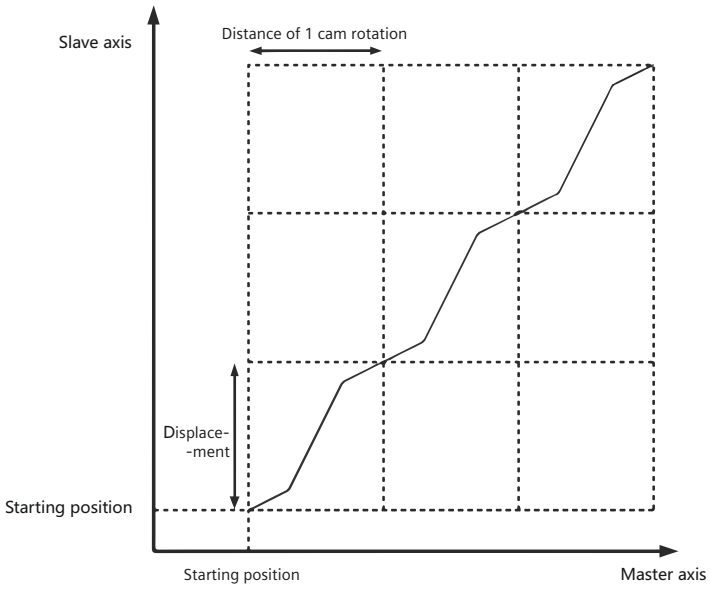
If the travel positions of the slave axis at the start and end are different, the movement is performed as a transfer cam (see Transfer Cam movement).

In the figure below, the horizontal axis shows the position of the main axis and the vertical axis shows the position of the slave axis.

Reciprocating cam movement



Transfer cam movement

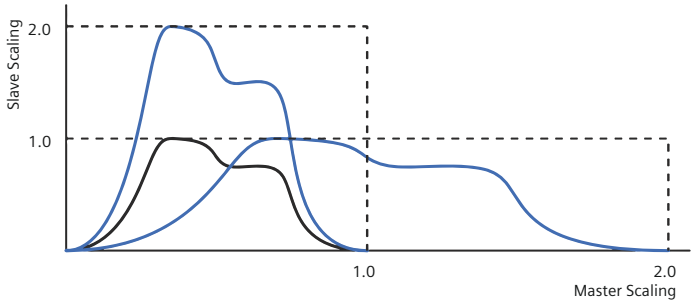


End Cam movement

To stop the cam halfway, use MC_CamOut (cam release), MC_ImmediateStop or MC_Stop (force stop).

Scaling

The phase of the master axis and the displacement of the slave axis for a specified cam table can be scaled by a specified ratio.
Different multipliers can be set for the master and slave axes.



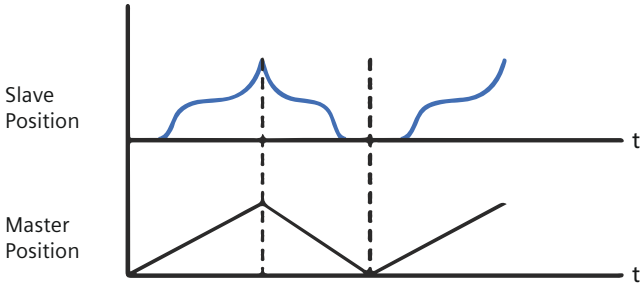
Direction

The cam movement of the slave axis can be started only when the direction of movement of the master axis is the same as the Direction.

When Direction is selected, it indicates that the system is currently in the InSync state.

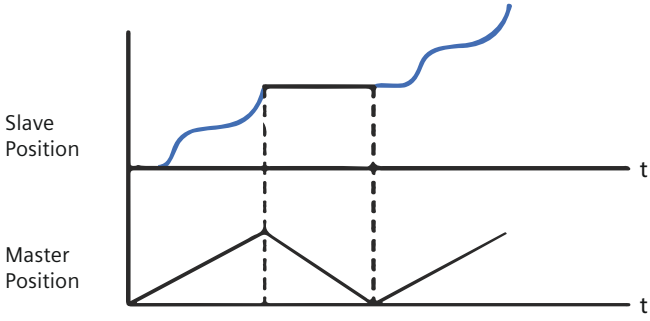
No direction specified

Cam movement starts regardless of master axis movement in positive or negative direction.



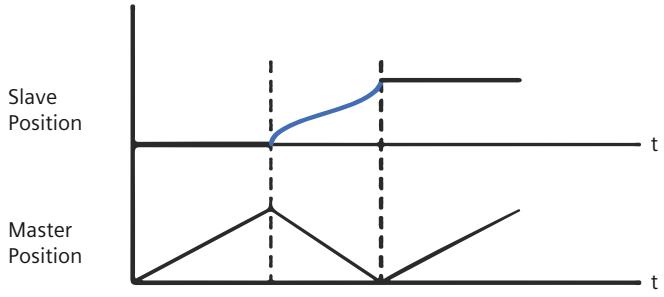
Positive direction

Cam movement starts when the master axis moves in the positive direction.



Negative specified

Cam movement starts when the master axis moves in the negative direction.



See also

Error code for single axis motion (Page 866)

15.4.4.2 MC_CamOut

Table 15-28 MC_CamOut

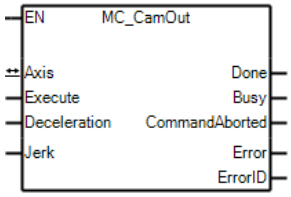
LAD / FBD	STL	Description
	**FBCALL MC_CamOut, INSTANCE, SlaveAxis, Execute, Deceleration, Jerk, Done, Busy, CommandAborted, Error, ErrorID**	Disable MC_CamIn cam behavior and decelerate the axis to stop

Table 15-29 Parameters for MC_CamOut

Parameter	Declaration	Data type	Initial value	Description
Axis	IN_OUT	SYS_AXIS	-	Reference to a slave axis
Execute	IN	BOOL	-	Start the instruction at rising edge
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	Value of the deceleration rate
Jerk	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	Reserved
Done	OUT	BOOL	False	TRUE when the instruction is completed
Busy	OUT	BOOL	False	The instruction is not finished
CommandAborted	OUT	BOOL	False	The instruction is aborted by another instruction
Error	OUT	BOOL	False	Signals that an error has occurred within the instruction
ErrorID	OUT	WORD	0	Error code (Page 869)

Note

Requirements

- The slave axis has been configured correctly.
- The slave axis is running the MC_CamIn instruction.

15.4.4.3 MC_GearIn

Table 15-30 MC_GearIn

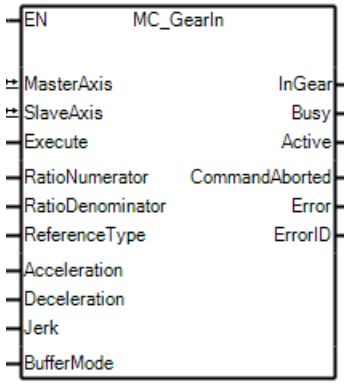
LAD / FBD	STL	Description
	**FBCALL MC_GearIn, INSTANCE, MasterAxis,SlaveAxis, Execute, RatioNumerator, RatioDenominator, ReferenceType, Acceleration, Deceleration, Jerk, BufferMode, InGear, Busy, Active, CommandAborted, Error, ErrorID**	This instruction specifies the gear ratio between the master axis and the slave axis and starts gear operation.

Table 15-31 Parameters for MC_GearIn

Parameter	Declaration	Data type	Initial value	Description
MasterAxis	IN_OUT	SYS_AXIS	-	Reference to a master axis
SlaveAxis	IN_OUT	SYS_AXIS	-	Reference to a slave axis
Execute	IN	BOOL	-	Start the instruction at rising edge
RatioNumerator	IN	DINT	1	Specify the numerator of the electronic gear ratio between the master and slave axes
RatioDenominator	IN	DWORD	1	Specify the denominator of the electronic gear ratio between the master and slave axes
ReferenceType	IN	BYTE	0	Reserved Now the command position from the same task cycle is used by default
Acceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Acc	Specify the acceleration rate
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	Specify the deceleration rate

Parameter	Declaration	Data type	Initial value	Description
Jerk	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	Specify the jerk
BufferMode	IN	BYTE	0	Specify the behavior when executing more than one motion instruction 0：Aborting, 1: Buffered
InGear	OUT	BOOL	False	Set as 1 once the slave axis speed synchronizes with the master axis speed
Busy	OUT	BOOL	False	The instruction is not finished
Active	OUT	BOOL	False	Indicate the instruction has control on the axis
CommandAborted	OUT	BOOL	False	The instruction is aborted by another instruction
Error	OUT	BOOL	False	Signals that an error has occurred within the instruction
ErrorID	OUT	WORD	0	Error code (Page 869)

Note

The slave axis number must be greater than the master axis number.

Note

When an encoder axis serves as the master axis:

- If MC_Home (direct homing) is triggered while the encoder axis is stationary, the slave axis remains stationary
- If MC_Home (direct homing) is triggered while the encoder axis is moving, the slave axis might report an overspeed error

Note

1. Requirements
   - The master and slave axis have been configured correctly.
   - The slave axis is enabled.
2. Override response
   - The MC_GearIn command can be aborted by the following Motion Control commands:
     MC_Halt
     MC_Stop
     MC_ImmediateStop
     MC_MoveAbsolute
     MC_MoveRelative
     MC_MoveVelocity
     MC_MoveJog
     MC_CamIn
     MC_GearIn
     MC_GearOut
   - The MC_GearIn command cannot be buffered by any Motion Control command.
   - The new MC_GearIn command aborts the following active Motion Control commands:
     MC_Halt
     MC_MoveAbsolute
     MC_MoveRelative
     MC_MoveVelocity
     MC_MoveJog
     MC_CamOut
     MC_GearOut
     MC_PhasingRelative
     MC_CamIn
     MC_GearIn
   - The new MC_GearIn command can buffer to the following Motion Control commands:
     MC_Halt
     MC_MoveAbsolute
     MC_MoveRelative

15.4.4.4 MC_GearOut

Table 15-32 MC_GearOut

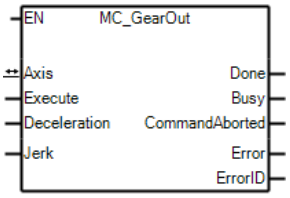
LAD / FBD	STL	Description
	`FBCALL MC_GearOut, INSTANCE, SlaveAxis, Exec, Dec, Jerk, Done, Busy, CommandAborted, Error, ErrorID`	Disable MC_GearIn gear behavior and decelerate the axis to stop

Table 15-33 Parameters for MC_GearOut

Parameter	Declaration	Data type	Initial value	Description
Aixs	IN_OUT	SYS_AXIS	-	Reference to a slave axis
Execute	IN	BOOL	-	Start the instruction at rising edge
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	Specify the deceleration rate
Jerk	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	Reserved
Done	OUT	BOOL	False	TRUE when the instruction is completed
Busy	OUT	BOOL	False	The instruction is not finished
CommandAborted	OUT	BOOL	False	The instruction is aborted by another instruction
Error	OUT	BOOL	False	Signals that an error has occurred within the instruction
ErrorID	OUT	WORD	0	Error code (Page 869)

Note

Requirements

- The slave axis has been configured correctly.
- The slave axis is running the MC_GearIn.

15.4.4.5 MC_PhasingRelative

Table 15-34 MC_PhasingRelative

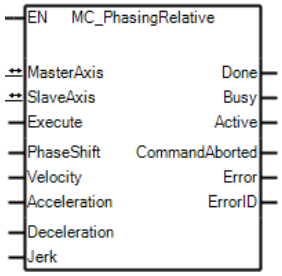
LAD / FBD	STL	Description
	`FBCALL MC_PhasingRelative, INSTANCE, MasterAxis, SlaveAxis, Execute, PhaseShift,Velocity,Acceleration,Deceleration, Jerk, Done, Busy,Active, CommandAborted, Error, ErrorID`	This instruction commands the MC_CamIn\MC_GearIn master axis to be offset by the specified phase relative to the current phase.

Table 15-35 Parameters for MC_PhasingRelative

Parameter	Declaration	Data type	Initial value	Description
MasterAxis	IN_OUT	SYS_AXIS	-	Reference to a master axis
SlaveAxis	IN_OUT	SYS_AXIS	-	Reference to a slave axis
Execute	IN	BOOL	-	Start the instruction at rising edge
PhaseShift	IN	REAL	-	Phase compensation
Velocity	IN	REAL	-	Specify the phase shift velocity
Acceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Acc	Specify the acceleration rate
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	Specify the deceleration rate
Jerk	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	Specify the jerk
Done	OUT	BOOL	False	TRUE when the instruction is completed
Busy	OUT	BOOL	False	The instruction is not finished
Active	OUT	BOOL	False	Indicate the instruction has control on the axis
CommandAborted	OUT	BOOL	False	The instruction is aborted by another instruction
Error	OUT	BOOL	False	Signals that an error has occurred within the instruction
ErrorID	OUT	WORD	0	Error code (Page 869)

Note

1. Requirements
   The master and slave axis must be running the MC_GearIn or MC_CamIn.
2. Override response
   The MC_PhasingRelative follows the behavior of MC_GearIn or MC_CamIn.

15.4.4.6 MC_GenerateCamTable

Table 15-36 MC_GenerateCamTable

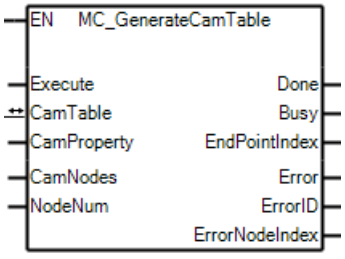
LAD / FBD	STL	Description
	**FBCALL MC_GenerateCamTable, INSTANCE, Exec, CamTable, CamProperty, CamNodes, NodeNum, Done, Busy, EndPointIndex, Error, ErrorID, ErrorNodeIndex**	The MC_GenerateCamTable instruction can be used to modifiy the cam table.

Table 15-37 Parameters for MC_GenerateCamTable

Parameter	Declaration	Data type	Initial value	Description
Execute	IN	BOOL	-	• Start the instruction at rising edge • Falling edge – When Done=1 or Error=1, set all output pins to 0 – No operation in all other case
CamTable	INOUT	SYS_CAM	-	Specify cam table to be modified and this value cannot be empty
CamProperty	INOUT	SYS_CAM_PROPERTY	-	Specify cam table properties and this value cannot be empty
CamNodes [1]	INOUT	SYS_CAM_NODE[361]	-	Key point array for modifying the designated cam table and this value cannot be empty
NodeNum [1]	IN	WORD	0	Specifies the number of elements in the key point array. Value range [2, 361]
Done	OUT	BOOL	-	Set to 1 when the instruction execution is complete
Busy	OUT	BOOL	-	Set to 1 when the instruction is executing
EndPointIndex	OUT	WORD	-	The index of the last cam data in the array when the instruction execution is complete
Error	OUT	BOOL	-	Set to 1 when error occurs during execution
ErrorID	OUT	WORD	-	Show the error code when error occurs during execution
ErrorNode-PointIndex	OUT	WORD	-	The index of the incorrectly key point in the Cam_Node array when the instruction execution fails

Note

The size of the array must be less than or equal to the NodeNum.

Data structure

Table 15-38 SYS_CAM_PROPERTY

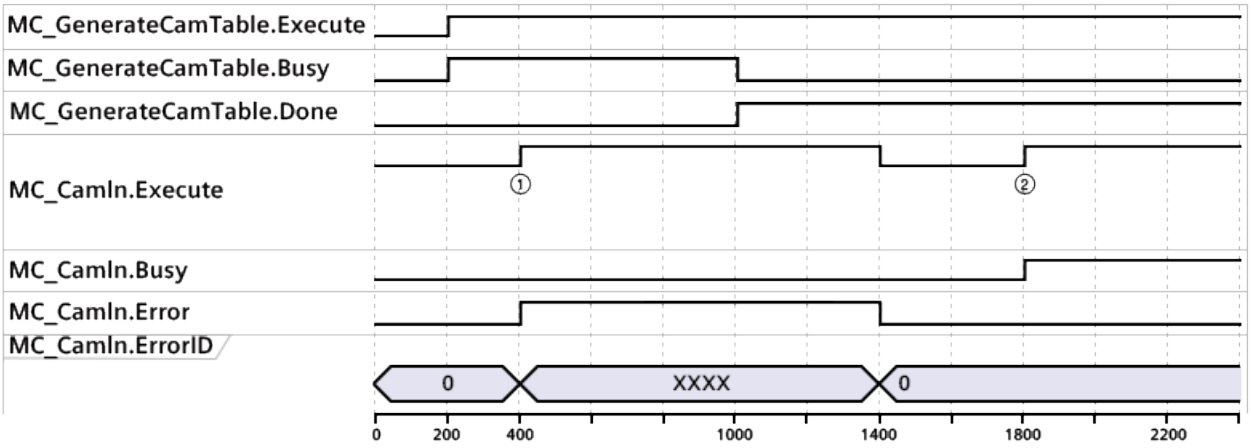
	Type	Whether to participate in calculation?
m_unit	BYTE	N
s_unit	BYTE	N
period	REAL	Y
init_vel	REAL	Y
init_acc	REAL	Y
limit_vel	REAL	N
limit_acc	REAL	N
limit_jerk	REAL	N
empty_datanum	WORD	N

Table 15-39 SYS_CAM_NODE

	Type
phase	REAL
position	REAL
velocity	REAL
acceleration	REAL
curve (0 : Constant, 1 : Line, 2 : Polynomic 3, 3 : Polynomic 5, 4 : Modified Constant Velocity)	WORD
interval	REAL

MC_GenerateCamTable and MC_CamIn

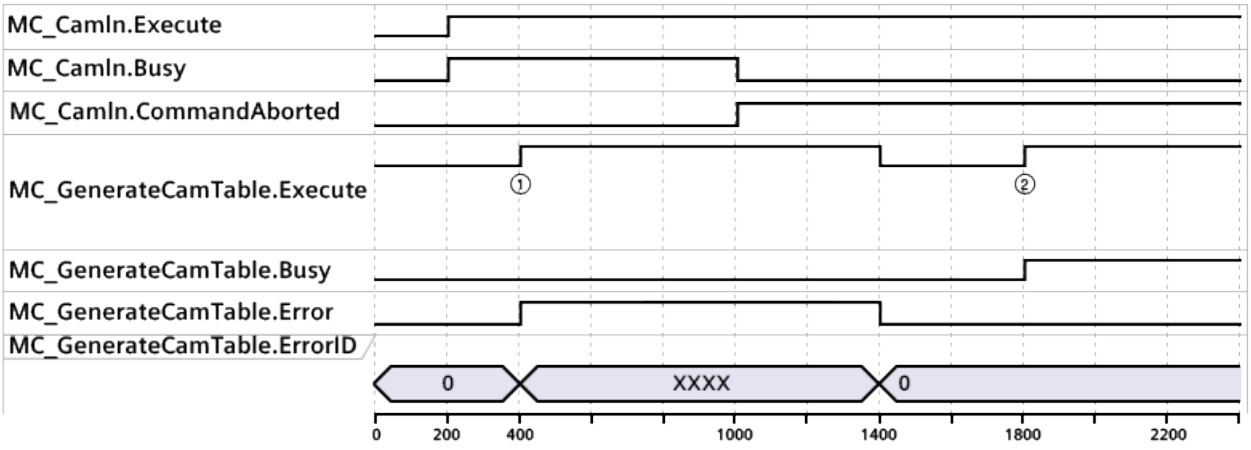
- You cannot execute the MC_CamIn instruction while executing MC_GenerateCamTable.



① Error occurs when executing MC_CamIn during busy state.

② MC_CamIn can be executed after MC_GenerateCamTable execution is complete.

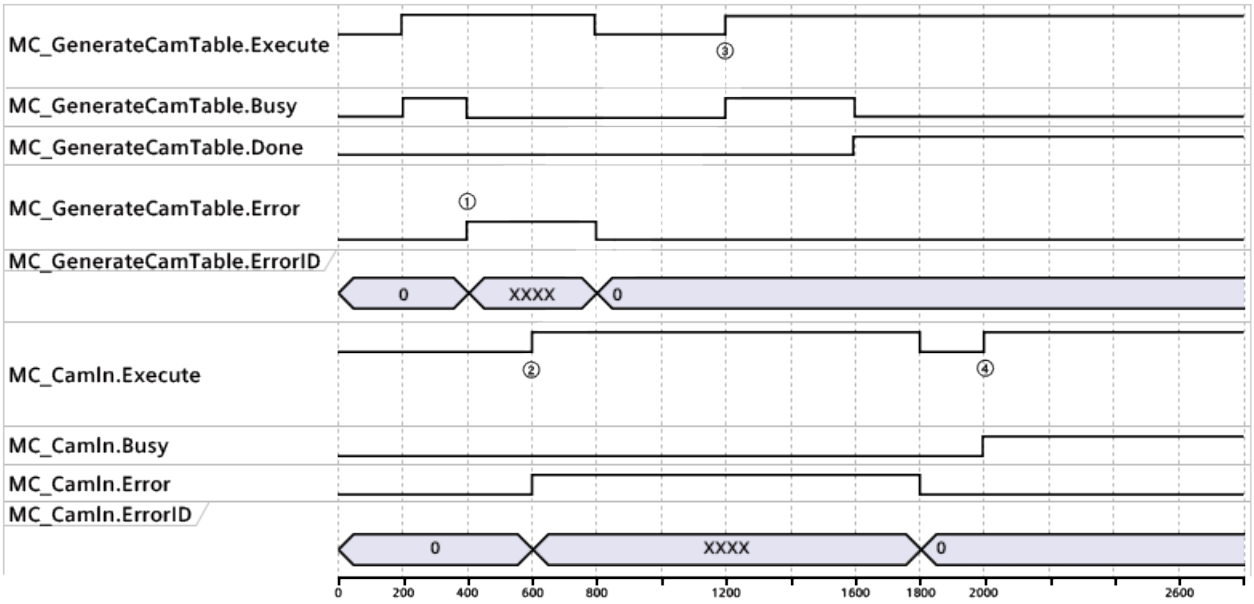
- Conversely, you cannot execute MC_GenerateCamTable instruction while executing MC_CamIn.
  In the following picture, CamIn.CommandAborted indicates that the axis synchronization motion is cancelled using CamOut.



① Error occurs when executing MC_GenerateCamTable during busy state.

② MC_GenerateCamTable can be executed after MC_CamIn execution is complete.

- If MC_GenerateCamTable reports an error, you must correctly modify the CamProperty and CamNodes data, then re-execute MC_GenerateCamTable successfully before you can execute CamIn again.



① Error occurs when executing MC_GenerateCamTable.

② Error occurs when executing MC_CamIn during error state.

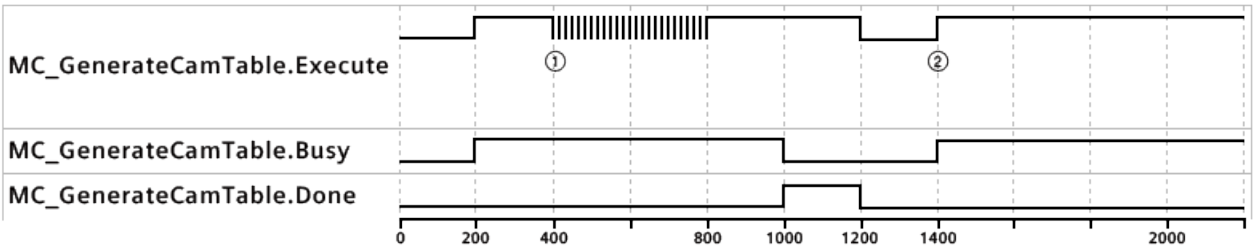
③ Handle the error and re-execute MC_GenerateCamTable.

④ MC_CamIn can be executed after MC_GenerateCamTable execution is complete.

Using MC_GenerateCamTable

Multiple MC_GenerateCamTable instructions cannot be executed simultaneously for the same cam table specified by the cam table pin.

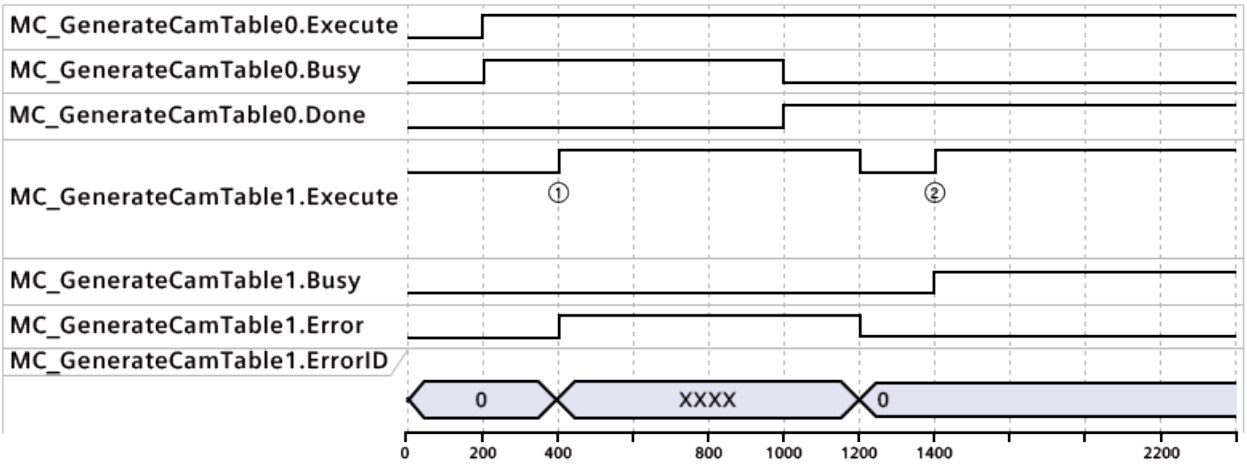
- The same FB instance



① Error occurs when executing the same MC_GenerateCamTable during a MC_GenerateCamTable instance is being executed.

② MC_GenerateCamTable can be executed again after execution is complete.

- Different FB instance

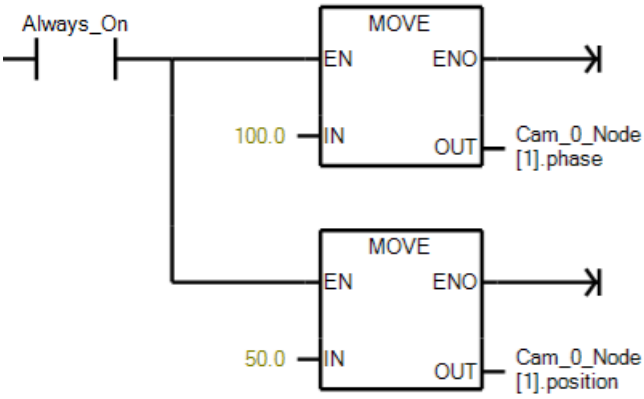


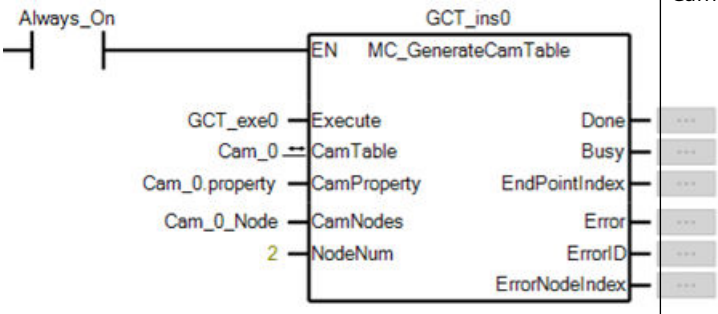
① The cam table cannot be executed while it is being modified.

② MC_GenerateCamTable can be executed again after modification is done.

Example: Use the CamProperty.CamNode generated by system

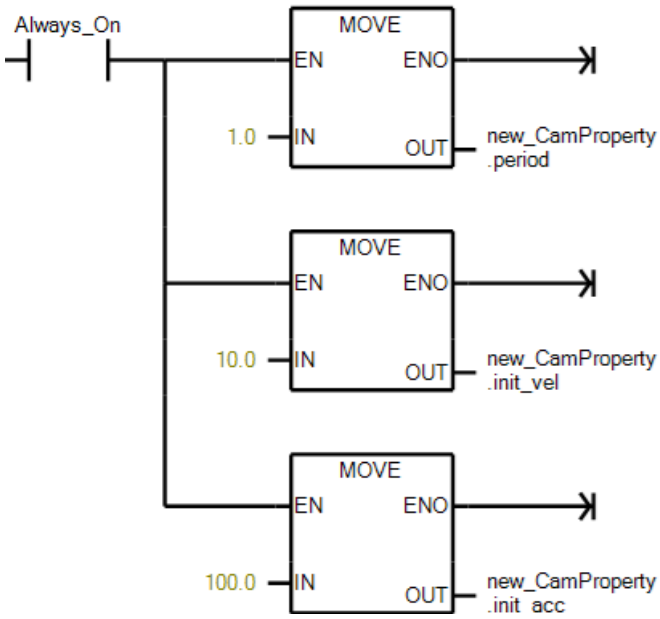
LAD/FBD	Description	STL

Network 1:
 
 	
	Modify the master axis phase and slave axis position of Cam_0_Node[1]	**LD Always_On**
		MOVE 100.0, Cam_0_Node[1].phase
		MOVE 50.0, Cam_0_Node[1].position

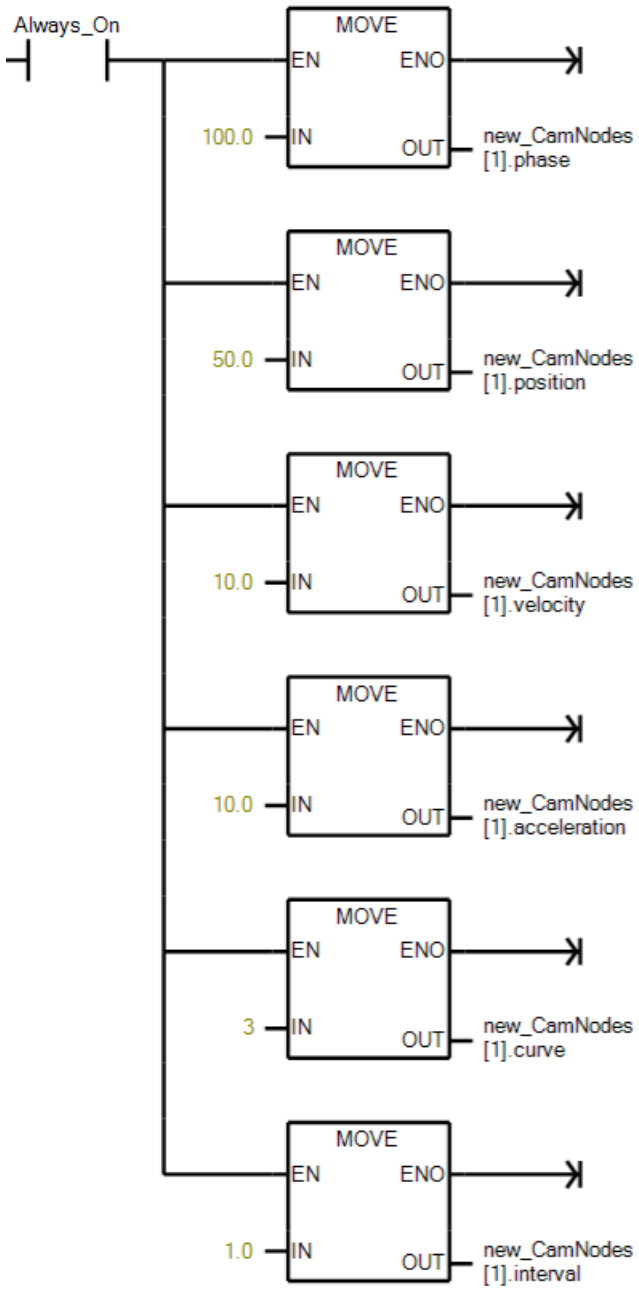
Network 2:  	Trigger MC_Generate-CamTable	**LD Always_On** **FBCALL MC_GenerateCamTable, GCT_ins0, GCT_exe0, Cam_0, Cam_0.property, Cam_0_Node, 2, GCT_ins0.Done, GCT_ins0.Busy, GCT_ins0.EndPointIndex, GCT_ins0.Error, GCT_ins0.ErrorID, GCT_ins0.ErrorNodeIndex**

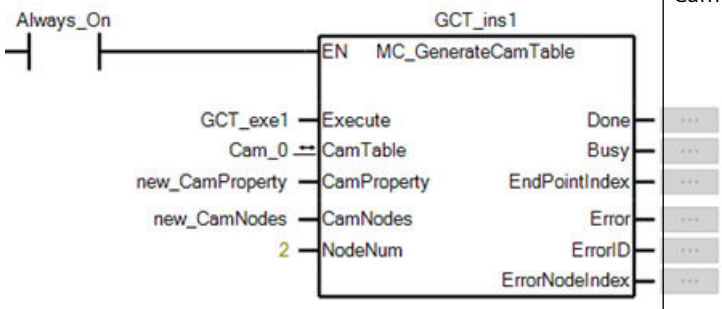
Example: Use the created global variable CamProperty, CamNode

LAD/FBD	Description	STL

Network 1:
 
 	
	1. Create two global variables:
	– new_CamProperty.period (type: SYS_CAM_PROPERTY)
	– new_CamNodes (type: Array of SYS_CAM_NODES)
	2. Set parameters for new_CamProperty
		LD Always_On
		MOVE 1.0, new_CamProperty.period
		MOVE 10.0, new_CamProperty.init_vel
		MOVE 100.0, new_CamProperty.init_acc

LAD/FBD	Description	STL

Network 2:
 
 	
	Set new_CamNode parameters, configure the first segment as a 5th curve	**LD Always_On**
		MOVE 100.0, new_CamNodes[1].phase
		MOVE 50.0, new_CamNodes[1].position
		MOVE 10.0, new_CamNodes[1].velocity
		MOVE 10.0, new_CamNodes[1].acceleration
		MOVE 3, new_CamNodes[1].curve
		MOVE 1.0, new_CamNodes[1].interval

LAD/FBD	Description	STL
Network 3:  	Trigger MC_Generate-CamTable	**LD Always_On** FBCALL MC_GenerateCamTable, GCT_ins1, GCT_exe1, Cam_0, new_CamProperty, new_CamNodes, 2, GCT_ins1.Done, GCT_ins1.Busy, GCT_ins1.EndPointIndex, GCT_ins1.Error, GCT_ins1.ErrorID, GCT_ins1.ErrorNodeIndex

15.4.5 Axes Group instructions

15.4.5.1 MC_GroupEnable

Table 15-40 MC_GroupEnable

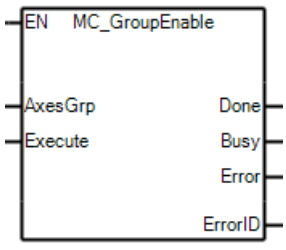
LAD / FBD	STL	Description
 	**FBCALL MC_GroupEnable, INSTANCE, AxesGrp, Execute, Done, Busy, Error, ErrorID**	The MC_GroupEnable instruction changes the axes group's state from GroupDisabled to GroupStandby. It is an administrative instruction that does not generate any movement.

Table 15-41 Parameters for MC_GroupEnable

Parameter	Declaration	Data type	Initial value	Description
AxesGrp	IN_OUT	SYS_AXES_GROUP	-	Reference to a axes group
Execute	IN	BOOL	-	Start the state change at the rising edge
Done	OUT	BOOL	False	When the instruction is completed, the axis group is in the GroupStandby state
Busy	OUT	BOOL	False	The instruction is not finished
Error	OUT	BOOL	False	Error occured during the instruction execution
ErrorID	OUT	WORD	0	Error code

Note

1. Requirements
   Each axis in the group and group itself must be configured and enabled
2. Override response
   The MC_GroupEnable cannot be aborted or buffered by any other Motion Control.

15.4.5.2 MC_GroupDisable

Table 15-42 MC_GroupDisable

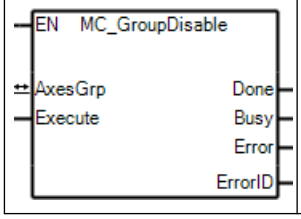
LAD / FBD	STL	Description
	**FBCALL MC_GroupDisable, INSTANCE, AxesGrp, Execute, Done, Busy, Error, ErrorID**	The MC_GroupDisable instruction changes the axes group state to GroupDisabled. It's an administrative instruction as no movement is generated.

Table 15-43 Parameters for MC_GroupDisable

Parameter	Declaration	Data type	Initial value	Description
AxesGrp	IN_OUT	SYS_AXES_GROUP	-	Reference to a group of axes
Execute	IN	BOOL	-	Start the state change at the rising edge
Done	OUT	BOOL	False	When the instruction is completed, the axis group is in the GroupDisabled state
Busy	OUT	BOOL	False	The instruction is not finished
Error	OUT	BOOL	False	Error occured during the instruction execution
ErrorID	OUT	WORD	0	Error code

Note

1. Requirements
   – Each axis in the group and group itself must be configured correctly
   – There is no error on the axes group
2. Override response
   – The MC_GroupDisable instruction cannot be aborted or buffered by any other Motion Control instruction.
   – Disabling the Axes Group aborts all Motion Control instructions for the associated Axes Group immediately.

15.4.5.3 MC_GroupReset

Table 15-44 MC_GroupReset

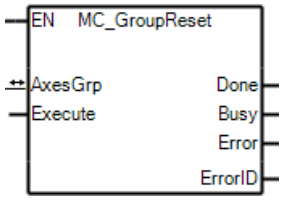
LAD / FBD	STL	Description
	**FBCALL MC_GroupReset, INSTANCE, AxesGrp, Execute, Done, Busy, Error, ErrorID**	The MC_GroupReset instruction transitions the state from GroupErrorStop to GroupStandby or GroupDisabled by resetting all internal axes group related errors. This instruction does not affect the outputs of any other instruction instances. The instruction also resets all axes in this axes group like MC_Reset.

Table 15-45 Parameters for MC_GroupReset

Parameter	Declaration	Data type	Initial value	Description
AxesGrp	IN_OUT	SYS_AXES_GROUP	-	Reference to a axes group
Execute	IN	BOOL	-	Start the execution at the rising edge
Done	OUT	BOOL	False	Resetting of all axes in the axes group completed
Busy	OUT	BOOL	False	The instruction is not finished
Error	OUT	BOOL	False	Error occured during the instruction execution
ErrorID	OUT	WORD	0	Error code

Note

1. Requirements
   - Each axis in the group and group itself must be configured correctly.
   - Axes Group must be in the ErrorStop state.
2. Override response
   - The MC_GroupReset instruction cannot be aborted or buffered by any other Motion Control instruction.
   - The new MC_GroupReset instruction does not abort or buffer any other active Motion Control instructions.

15.4.5.4 MC_GroupStop

Table 15-46 MC_GroupStop

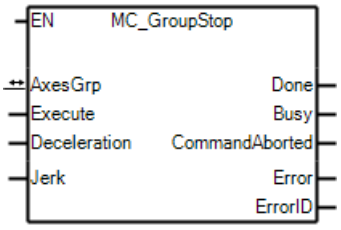
LAD / FBD	STL	Description
	**FBCALL MC_GroupStop, INSTANCE, AxesGrp, Execute, Deceleration, Jerk, Done, Busy, Active, CommandAborted, Error, ErrorID**	The MC_GroupStop instruction commands the axis group to decelerate and stop, and transfers the axes group to the state GroupStopping. The instruction aborts any ongoing instruction execution. When the axes group is in the GroupStopping state, no other instruction can perform motion commands on the same axes group. After the axes group has reached velocity zero, the Done output is set to TRUE immediately. The axes group remains in the GroupStopping state as long as the Execute input is TRUE or velocity zero is not yet reached. As soon as the Done output is set to 1, and the Execute input is 0, the axes group transitions to the GroupStandby state.

Table 15-47 Parameters for MC_GroupStop

Parameter	Declaration	Data type	Initial value	Description
AxesGrp	IN_OUT	SYS_AXES_GROUP	-	Reference to a group of axes
Execute	IN	BOOL	-	Start the execution at the rising edge
Deceleration	IN	REAL	-	Value of the deceleration
Jerk	IN	REAL	0.0	Value of the jerk
Done	OUT	BOOL	False	The axis group movement is completed
Busy	OUT	BOOL	False	The instruction is not finished
CommandAborted	OUT	BOOL	False	The command is aborted by other instruction
Error	OUT	BOOL	False	Error occured during the instruction execution
ErrorID	OUT	WORD	0	Error code

Note

1. Requirements
   - The positioning axes group has been configured correctly.
   - The axes group is enabled.
2. Override response
   - The MC_GroupStop instruction can be aborted by the following Motion Control instructions:
     MC_GroupDisable
     MC_Power
   - The MC_GroupStop instruction cannot be buffered by any Motion Control instruction.
   - The MC_GroupStop instruction aborts the following active Motion Control instructions:
     MC_MoveLinearAbsolute
     MC_MoveLinearRelative
   - The MC_GroupStop instruction cannot buffer to any Motion Control instruction.

15.4.5.5 MC_MoveLinearAbsolute

Table 15-48 MC_MoveLinearAbsolute

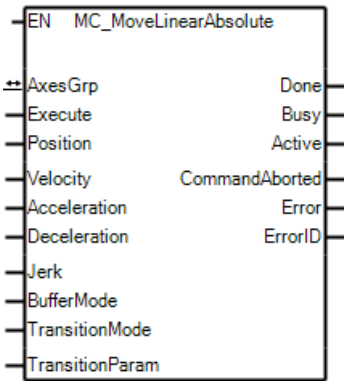
LAD / FBD	STL	Description
	**FBCALL MC_MoveLinearAbsolute , INSTANCE, AxesGrp, Execute, Position, Velocity, Acceleration, Deceleration, Jerk, BufferMode, TransitionMode, TransitionParameter, Done, Busy, Active, CommandAborted, Error, ErrorID**	The MC_MoveLinearAbsolute instruction command a linear interpolated movement on an axes group from the actual position of the axes group to an absolute position in the Cartesian coordinate system. This instruction supports 2D and 3D linear movement.

Table 15-49 Parameters for MC_MoveLinearAbsolute

Parameter	Declaration	Data type	Initial value	Description
AxesGrp	IN_OUT	SYS_AXES_GROUP	-	Reference to a axes group
Execute	IN	BOOL	-	Start the execution at the rising edge
Position	IN	ARRAY [0..3] OF REAL	-	Array [0..3] of absolute end position for each dimension (X, Y, Z axis) in the Certesian coordinate system. Array[3] is reserved

Parameter	Declaration	Data type	Initial value	Description
Velocity	IN	REAL	-	• Velocity [u/s] for the defined path of the Cartesian coordinate system • Always positive • Not necessarily reached
Acceleration	IN	REAL	-	• Value of the **Acceleration** • Always positive • Not necessarily reached
Deceleration	IN	REAL	-	• Value of the **Deceleration** • Always positive • Not necessarily reached
Jerk	IN	REAL	0.0	• Value of the **Jerk** • Always positive
BufferMode	IN	BYTE	0	• 0: Aborting. According to the Deceleration value of the original command, decelerate to zero with its original path. And then start this command. • 1: Buffered. Start instruction after current motion has finished
Transition-Mode	IN	BYTE	0	Reserved
TransitionPar-am	IN	DWORD	-	Reserved
Done	OUT	BOOL	False	All axes has reached the commanded end position
Busy	OUT	BOOL	False	The instruction is not finished
Active	OUT	BOOL	False	Indicate that the instruction has control on the axes group
CommandA-borted	OUT	BOOL	False	The command is aborted by another command
Error	OUT	BOOL	False	Error occurred during the instruction execution
ErrorID	OUT	WORD	0	Error code

Note

1. Requirements
   - The axes group is configured correctly.
   - The axes group is enabled.
2. Override response
   - The MC_MoveLinearAbsolute instruction can be aborted by the following Motion Control instructions:
     MC_MoveLinearAbsolute
     MC_MoveLinearRelative
     MC_GroupDisable
     MC_GroupStop
     MC_Power
   - The MC_MoveLinearAbsolute instruction can be buffered by the following Motion Control instructions:
     MC_MoveLinearAbsolute
     MC_MoveLinearRelative
   - The new MC_MoveLinearAbsolute instruction can abort the following active Motion Control instructions:
     MC_MoveLinearAbsolute
     MC_MoveLinearRelative
   - The new MC_MoveLinearAbsolute instruction can buffer to the following Motion Control instructions:
     MC_MoveLinearAbsolute
     MC_MoveLinearRelative

15.4.5.6 MC_MoveLinearRelative

Table 15-50 MC_MoveLinearRelative

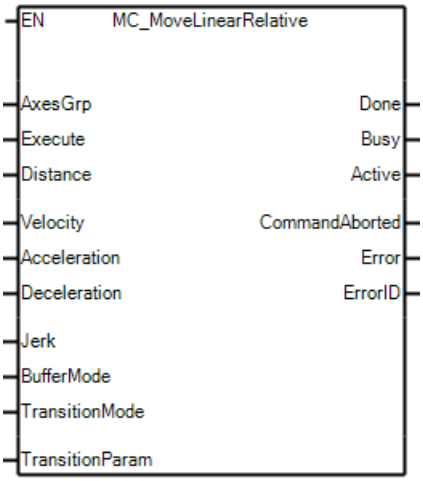
LAD / FBD	STL	Description
	**FBCALL MC_MoveLinearRelative , INSTANCE, AxesGrp, Execute, Distance, Velocity, Acceleration, Deceleration, Jerk, BufferMode, TransitionMode, TransitionParameter, Done, Busy, Active, CommandAborted, Error, ErrorID**	The MC_MoveLinearRelative instruction commands a linear interpolated movement on an axes group from the actual position of the axes group to a relative position in the Cartesian coordinate system. This instruction supports 2D and 3D linear movement.

Table 15-51 Parameters for MC_MoveLinearRelative

Parameter	Declaration	Data type	Initial value	Description
AxesGrp	IN_OUT	SYS_AXES_GROUP	-	Reference to a axes group
Execute	IN	BOOL	-	Start the execution at the rising edge
Distance	IN	ARRAY [0..3] of REAL	-	Array [0..3] of relative distance for each dimension (X, Y, Z axis) in the Certesian coordinate system. Array[3] is reserved
Velocity	IN	REAL	-	• Maximum velocity [u/s] for the defined path of the Cartesian coordinate system • Always positive • Not necessarily reached
Acceleration	IN	REAL	-	• Maximum **Acceleration** • Always positive • Not necessarily reached
Deceleration	IN	REAL	-	• Maximum **Deceleration** • Always positive • Not necessarily reached
Jerk	IN	REAL	0.0	• Maximum **Jerk** • Always positive

Parameter	Declaration	Data type	Initial value	Description
BufferMode	IN	BYTE	0	• 0: Aborting. According to the Deceleration value of the original instruction, decelerate to zero with its original path. And then start this instruction. • 1: Buffered. Start instruction after current motion has finished
Transition-Mode	IN	BYTE	0	0
TransitionPara	IN	DWORD	-	Reserved
Done	OUT	BOOL	False	All axes has reached the commanded end position
Busy	OUT	BOOL	False	The instruction is not finished
Active	OUT	BOOL	False	Indicate that the instruction has control on the axes group
CommandA-borted	OUT	BOOL	False	The command is aborted by another command
Error	OUT	BOOL	False	Error occurred during the instruction execution
ErrorID	OUT	WORD	0	Error code

Note

1. Requirements
   – The axes group is configured correctly.
   – The axes group is enabled.
2. Override response
   – The MC_MoveLinearRelative instruction can be aborted by the following Motion Control instructions:
     MC_MoveLinearAbsolute
     MC_MoveLinearRelative
     MC_GroupDisable
     MC_GroupStop
     MC_Power
   – The MC_MoveLinearRelative instruction can be buffered by the following Motion Control instructions:
     MC_MoveLinearAbsolute
     MC_MoveLinearRelative
   – The new MC_MoveLinearRelative instruction can abort the following active Motion Control instructions:
     MC_MoveLinearAbsolute
     MC_MoveLinearRelative
   – The new MC_MoveLinearRelative instruction can buffer to the following Motion Control instructions:
     MC_MoveLinearAbsolute
     MC_MoveLinearRelative

15.5 Error codes

15.5.1 Error code for single axis motion

Table 15-52 Error code for motion system configuration

Error code	Description
0x0001	Invalid axis type V3.0 supports 0-Virtual Axis, 1-PTO Axis, 2-Encoder Axis
0x0002	Invalid Axis Mode
0x0003	Invalid emergency stop trigger type
0x0004	Invalid Homing Reference mark type
0x0005	Invalid Homing mode
0x0006	Invalid Axis TO pointer area
0x0007	Invalid emergency stop response type
0x0008	Invalid hardware negative limit response type
0x0009	Invalid hardware positive limit response type
0x000A	Invalid Axis TO pointer type
0x000B	Unknown version
0x000C	Axis SDB Record length error
0x000D	Axis SDB Record ID duplicated
0x000E	Probe 0 input point occupied or input point out of range
0x000F	Probe 1 input point occupied or input point out of range
0x0010	The output point is misconfigured
0x0011	Axis TO address pointer check error
0x001A	Axis ID error
0x001B	Hardware positive limit Input point is occupied or Input point is out of range
0x001C	Hardware negative limit Input point is occupied or Input point is out of range
0x001D	Emergency stop Input point is occupied or Input point is out of range
0x001E	Homing reference Input point is occupied or Input point is out of range
0x001F	Zero signal Input point is occupied or Input point is out of range
0x0020	Invalid output type
0x0021	Incorrect Input mode of the encoder axis (single-phase/AB-phase...)
0x0022	Encoder axis phase A Input point is occupied or Input point is out of range (refers to the first Input point)
0x0023	Encoder axis phase B Input point is occupied or Input point is out of range (refers to the second Input point)
0x0024	The HSC channel used by the encoder axis is occupied or out of range (e.g. HSC6 is used for the first encoder axis and HSC7 is used for the second encoder axis in V3.0)
0x0025	Phase A Output Point configuration conflicts
0x0026	Phase B Output Point configuration conflicts
0x0028	The encoder axis is not configured in the same group when using two input points (I0. X or I1.X refers to a group of each one)
0x0029	Invalid axis TO pointer element number

0x0031	The velocity filtering time constant of the encoder axis is unreasonable. V3.0 Pulse range: [0 ms, 2000 ms]
0x0032	The number of PTO axes is out of limit
0x0033	The number of Virtual axes is out of limit
0x0034	The number of Encoder axes is out of limit
0x0035	The number of PN axes is out of limit
0x0036	The total number of axes is out of limit

Table 15-53 Error code for motion axis configuration

Error code	Description
0x0200	TO version error
0x0201	The unit of measurement is incorrect V3.0 only supports 0-pulse, 1-cm, 2-mm, 3-ft, 4-deg
0x0202	The number of pulses in one rotation of the PLS/REV motor/encoder is incorrect, which should be greater than 0. When the measurement system unit is configured as pulse, the default is 1
0x0203	The distance to which the axis is moved in one rotation is incorrect, and it should be greater than 0. When the measurement system unit is configured as pulse, the default is 1
0x0204	The gear numerator is incorrect and it should be greater than 0. When the measurement system unit is configured as pulse, the default is 1
0x0205	The gear denominator is incorrect and it should be greater than 0. When the measurement system unit is configured as pulse, the default is 1
0x0206	The axis movement mode is incorrect, only supports 0-linear axis and 1-rotary axis
0x0207	The modulo axis starts in the wrong position. The start of the modulo axis should be greater than the minimum value of the single-precision floating-point number, and the start of the modulo axis plus the mode length should be less than the maximum value of the single-precision floating-point number
0x0208	The modulo axis mode length is incorrect, and it should be greater than 0
0x020A	When the software limit is open and when it is equipped with a modulo axis, the position >= mode length + start position
0x020B	Linear axis, lower limit position >= upper limit position Or when it is configured as a modulo axis, the position >= the mode length + starting position Or when it is configured as a modulo axis, the position < the starting position
0x020C	The default acceleration is incorrect, which should be in this range: [1 pulse/s^2, maximum acceleration]
0x020D	The default deceleration is incorrect, which should be in this range: [1 pulse/s^2, maximum deceleration]
0x020E	The default jerk is wrong, which should be in this range: [0, 2*10^8 pulses/s^3]
0x020F	The maximum velocity is wrong, which should be in this range [200 Hz, 200 kHz] after conversion to pulse.
0x0210	The maximum acceleration is incorrect, which should be in this range: [1 pulse/s^2, 2*10^8 pulses/s^2]

0x0211	The maximum deceleration is incorrect, which should be in this range: [1 pulse/s^2, 2*10^8 pulses/s^2]
0x0215	The axis has activated the axis emergency stop
0x0216	The axis has activated hardware positive limit
0x0217	The axis has activated hardware negative limit
0x0218	The axis has activated software positive limit
0x0219	The axis has activated software negative limit
0x021A	An error occurred during execution of the motion instruction, refer to the error code in the corresponding instruction.
0x0220	When the axis is a modulo axis, the current position is not in the software limit window

Table 15-54 Error code for PTO single axis motion control

Error code	Description
0x2000	Multi-instance calls to MC_Power are not allowed
0x2400	The velocity of the axis is not 0, which means that the deceleration stop is not completed
0x2500	RPS is not enabled in the axis configuration
0x2501	The selected Homing mode is not supported (Numbers other than 0, 1, 3 entered)
0x2502	The current axis is Homing
0x2503	During the homing, the signal behaves unexpectedly
0x2504	position is out of the valid range
0x2C00	This error occurs when an instance of the MC_TouchProbe instruction is executed again while it is still in the BUSY state
0x2C01	MC_TouchProbe instruction not executed
0x2C02	This error occurs if MC_AbortTrigger or MC_TouchProbe is executed while Homing is in progress.
0x1090	When using the Control Panel, if the connection between the PLC and STEP 7-Micro/Win SMART V3 is interrupted.

Table 15-55 Error code for common errors reported at the instruction pin

Error code	Description
0x1000	CPU does not support this instruction
0x1001	TO_AXIS does not support this instruction
0x1002	Errors caused by multiple instructions executed on the same TO_Axis
0x1003	TO_AXIS does not exist or is invalid
0x1004	Invalid TO axis configuration
0x1005	TO_AXIS is in wrong PLCopen state
0x1006	The axis group is running and cannot execute the current single-axis instruction
0x1007	Error due to single axis Instruction is running
0x1008	Error due to zero position is unknown (not homed)
0x1009	This instruction does not support buffer mode
0x1010	Too many instructions buffered for this TO_AXIS
0x1011	Move direction is not supported by this instruction

0x1012	Error due to emergency stop is activated
0x1013	Error due to hardware positive limit is activated
0x1014	Error due to hardware negative limit is activated
0x1015	Distance/position is illegal
0x1016	Velocity is out of range
0x1017	Acceleration/deceleration is out of rage
0x1018	Jerk is out of range
0x1019	The instruction is interrupted by a higher priority task that is triggered simultaneously
0x1020	The input ProbeID for this instruction is invalid (Can only be 0, 1)
0x1021	An error occurs when the same ProbeID of the same axis calls different probe instructions (e.g. MC_TouchProbe)
0x1022	When there is a window, the instruction input WindowStart and WindowEnd exceed the maximum or minimum floating-point limits
0x1023	Illegal interrupt edge triggering mode (only 0: rising edge and 1: falling edge are supported.)
0x1024	Probe is not enabled
0x1025	When WindowOnly=true, the FirstPostion and LastPosition of the modulo axis input are out of range (0-mode length)
0x1030	Triggered the activated software positive limit switch
0x1031	Triggered the activated software negative limit switch
0x1032	Error detected on TO axis
0x1080	Both Jogforward and JogBackward are in rising edge
0x1FFF	Inner unexpected error code

15.5.2 Error code for axes group

Table 15-56 Error code for axes group

Error code	Description
0x0381	Axes group is stopped due to an axis error
0x5100	Axis group instruction execution failed due to an axis error

Error code for single axis motion (page 866)

15.5.3 Error codes for electronic cam and gear

Table 15-57 Error code for Electronic cam and gear

Error code	Description
0x4000	Same slave and master axis ID
0x4001	The master axis ID should be smaller than the slave axis ID
0x4002	After synchronization, the deceleration of the slave axis is out of the range
0x4003	After synchronization, the acceleration of the slave axis is out of the range

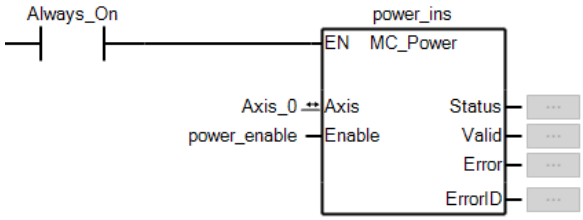
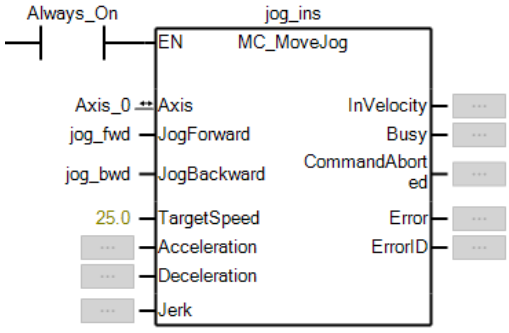
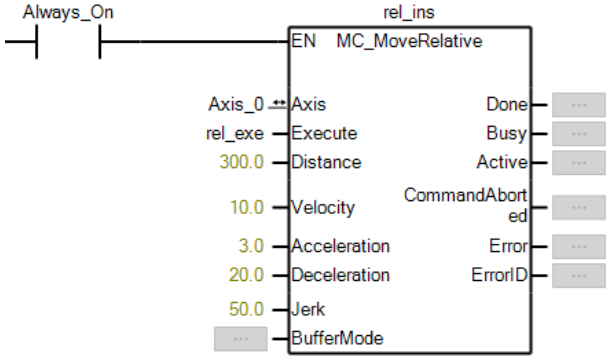
0x4005	Verify the parameter validity of the GearIn denominator
0x4006	Verify the parameter validity of the GearIn numerator
0x4007	During catch-up or synchronization, the velocity exceeds the maximum velocity in the axis configuration
0x4100	The instruction currently being executed is not GearIn
0x4200	Same slave and master axis ID
0x4201	The master axis ID should be smaller than the slave axis ID
0x4202	The direction of the instruction can only be 0 positive, 1 negative, and 2 no direction
0x4203	startmode only supports 0 absolute, 1 relative
0x4204	Verify the parameter validity of CamIn StartPosition
0x4205	Verify the Parameter validity of CamIn MasterStartDistance
0x4206	Verify the Parameter validity of CamIn MasterScaling
0x4207	Verify the Parameter validity of CamIn SlaveScaling
0x4208	Verify the Parameter validity of CamIn MasterOffset
0x4209	Verify the Parameter validity of CamIn SlaveOffset
0x4210	Cam table initialization failed
0x4211	Cam table initialization is not yet completed
0x4212	The cam table is currently being used by GenerateCamTable
0x4300	The instruction currently in execution is not CamIn
0x4400	An error occurred in both the synchronous motion Cam/Gear, and Phasing
0x4401	The master axis ID is incorrect. The current master axis must be the same as the current Gear or the current Cam
0x4402	PhaseShift is out of range
0x4403	Phasing cannot continue due to an unexpected error
0x4500	This cam table is in use
0x4501	The calculated CamData based on the specified CamNode array exceeds the original number of Data entries in the cam table
0x4505	The number of nodes is less than 2 or greater than 361.
0x4506	Cam table calculation timeout
0x4507	period parameter in property is out of valid range
0x4508	Invalid cam table curve type
0x4509	Initial velocity in property is out of valid range
0x450a	Initial acceleration in property is out of valid range
0x450b	Master axis phases are not in ascending order
0x450c	Master axis phases are too close together
0x450d	Master axis phase is out of valid range
0x450e	Slave axis position is out of valid range
0x450f	Slave axis velocity is out of valid range
0x4510	Slave axis acceleration is out of valid range
0x4511	Interpolation interval is out of range
0x4512	Required datanum for calculation results exceeds reserved capacity
0x4515	Cam table interpolation interrupted by CPU Stop

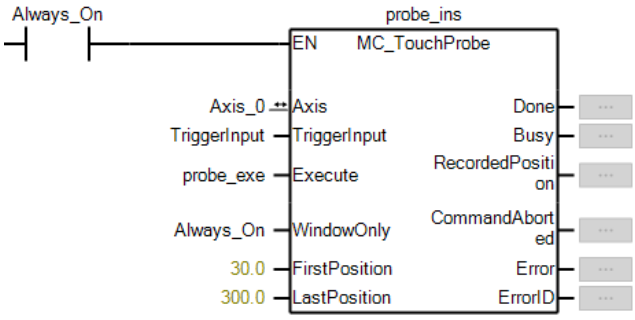
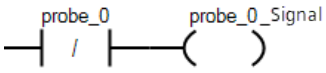
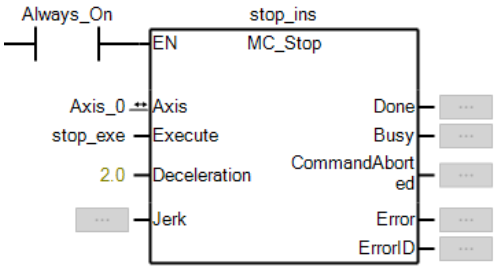
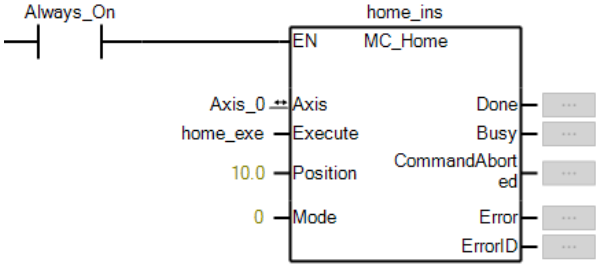
15.6 Example programs

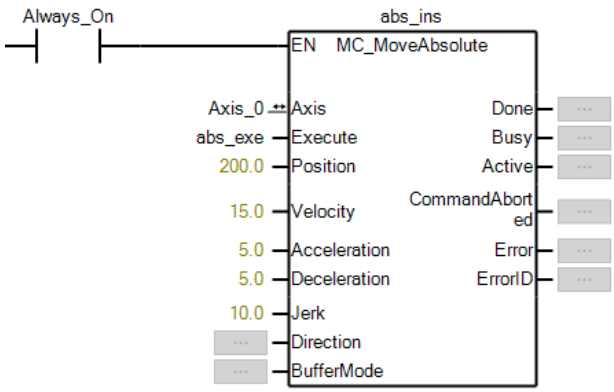
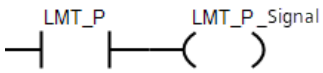
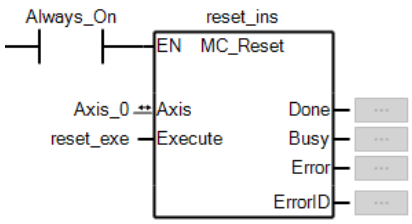
15.6.1 Single Axis example

The following program demonstrates the calling of several single-axis functions, such as Enable, Jog, and Probe.

Table 15-58 Example: Single Axis

LAD/FBD	Description	STL
Network 1:  	Enable axis	LD Always_On FBCALL MC_Power, power_ins, Axis_0, power_enable, power_ins.Status, power_ins.Valid, power_ins.Error, power_ins.ErrorID
Network 2:  	Axis_0, jog function	LD Always_On FBCALL MC_MoveJog, jog_ins, Axis_0, jog_fwd, jog_bwd, 25.0, Axis_0.Axis_Config.Dynamic_Default.Default_Acc, Axis_0.Axis_Config.Dynamic_Default.Default_Decel, Axis_0.Axis_Config.Dynamic_Default.Default_Jerk, jog_ins.InVelocity, jog_ins.Busy, jog_ins.CommandAborted, jog_ins.Error, jog_ins.ErrorID
Network 3:  	Axis_0, relative positioning	LD Always_On FBCALL MC_MoveRelative, rel_ins, Axis_0, rel_exe, 300.0, 10.0, 3.0, 20.0, 50.0, rel_ins.BufferMode, rel_ins.Done, rel_ins.Busy, rel_ins.Active, rel_ins.CommandAborted, rel_ins.Error, rel_ins.ErrorID

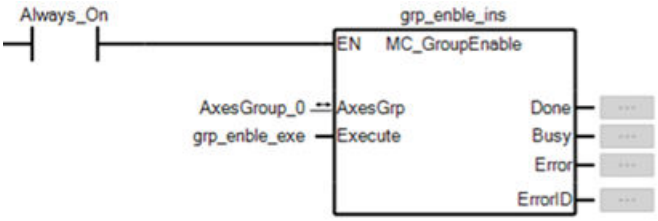
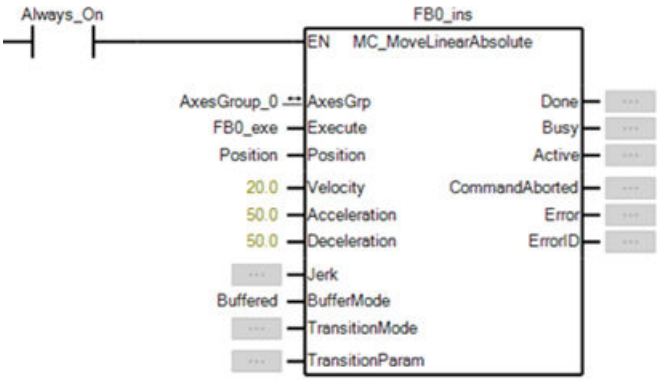
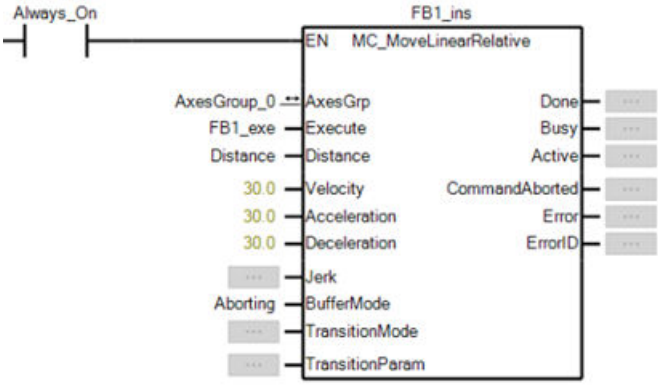
LAD/FBD	Description	STL
Network 4:  	Axis_0, probe	**LD Always_On** **FBCALL MC_TouchProbe, probe_ins, Axis_0, TriggerInput, probe_exe, Always_On, 30.0, 300.0, probe_ins.Done, probe_ins.Busy, probe_ins.RecordedPosition, probe_ins.CommandAborted, probe_ins.Error, probe_ins.ErrorID**
Network 5:  	Trigger probe	**LDN probe_0** **= CPU_Output11**
Network 6:  	Axis_0 decelerates to stop	**ID Always_On** **FBCALL MC_Stop, stop_ins, Axis_0, stop_exe, 2.0, Axis_0.Axis_Config.Dynamic_Default.Default_Jerk, stop_ins.Done, stop_ins.Busy, stop_ins.CommandAborted, stop_ins.Error, stop_ins.ErrorID**
Network 7:  	Set Axis_0 current position	**LD Always_On** **FBCALL MC_Home, home_ins, Axis_0, home_exe, 10.0, 0, home_ins.Done, home_ins.Busy, home_ins.CommandAborted, home_ins.Error, home_ins.ErrorID**

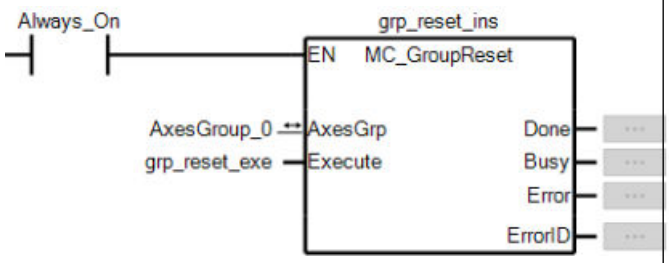
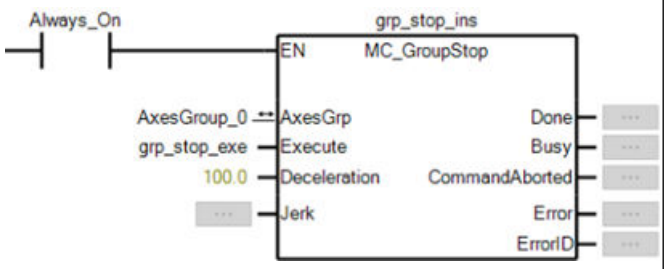
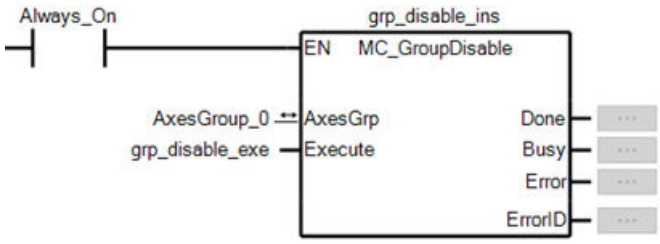
LAD/FBD	Description	STL
Network 8:  	Axis_0, absolute positioning	**LD Always_On** **FBCALL MC_MoveAbsolute, abs_ins, Axis_0, abs_exe, 200.0, 15.0, 5.0, 5.0, 10.0, abs_ins.Direction, abs_ins.BufferMode, abs_ins.Done, abs_ins.Busy, abs_ins.Active, abs_ins.CommandAborted, abs_ins.Error, abs_ins.ErrorID**
Network 9:  	Trigger positive limit signal	**LD LMT_P** **= CPU_Output9**
Network 10:  	Reset Axis_0	**LD Always_On** **FBCALL MC_Reset, reset_ins, Axis_0, reset_exe, reset_ins.Done, reset_ins.Busy, reset_ins.Error, reset_ins.ErrorID**

15.6.2 Axes group example

The following program demonstrates the calling of axes group.

Table 15-59 Example: Axes group

LAD/FBD	Description	STL
Network 1:  	Enable single axis in the axes group	**LD Always_On** FBCALL MC_GroupEnable, grp_enble_ins, AxesGroup_0, grp_enble_exe, grp_enble_ins.Done, grp_enble_ins.Busy, grp_enble_ins.Error, grp_enble_ins.ErrorID
Network 2:  	Call MC_MoveLinearAbsolute	**LD Always_On** FBCALL MC_MoveLinearAbsolute, FB0_ins, AxesGroup_0, FB0_exe, Position, 20.0, 50.0, 50.0, FB0_ins.Jerk, Buffered, FB0_ins.TransitionMode, FB0_ins.TransitionParam, FB0_ins.Done, FB0_ins.Busy, FB0_ins.Active, FB0_ins.CommandAborted, FB0_ins.Error, FB0_ins.ErrorID
Network 3:  	Call MC_MoveLinearRelative	**LD Always_On** FBCALL MC_MoveLinearRelative, FB1_ins, AxesGroup_0, FB1_exe, Distance, 30.0, 30.0, 30.0, FB1_ins.Jerk, Aborting, FB1_ins.TransitionMode, FB1_ins.TransitionParam, FB1_ins.Done, FB1_ins.Busy, FB1_ins.Active, FB1_ins.CommandAborted, FB1_ins.Error, FB1_ins.ErrorID

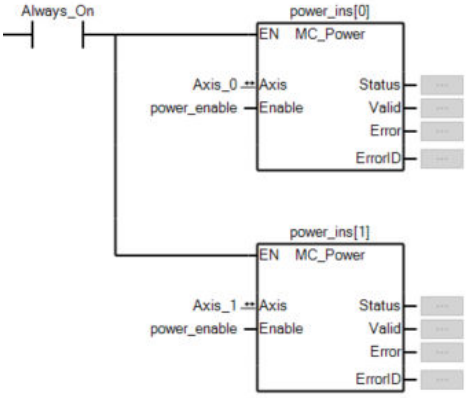
LAD/FBD	Description	STL
Network 4:  	Reset axes group	**LD Always_On** FBCALL MC_GroupReset, grp_reset_ins, AxesGroup_0, grp_reset_exe, grp_reset_ins.Done, grp_reset_ins.Busy, grp_reset_ins.Error, grp_reset_ins.ErrorID
Network 5:  	Stop axes group	LD Always_On FBCALL MC_GroupStop, grp_stop_ins, AxesGroup_0, grp_stop_exe, 100.0, grp_stop_ins.Jerk, grp_stop_ins.Done, grp_stop_ins.Busy, grp_stop_ins.CommandAborted, grp_stop_ins.Error, grp_stop_ins.ErrorID
Network 6:  	Disable the axes group function	ID Always_On FBCALL MC_GroupDisable, grp_disable_ins, AxesGroup_0, grp_disable_exe, grp_disable_ins.Done, grp_disable_ins.Busy, grp_disable_ins.Error, grp_disable_ins.ErrorID

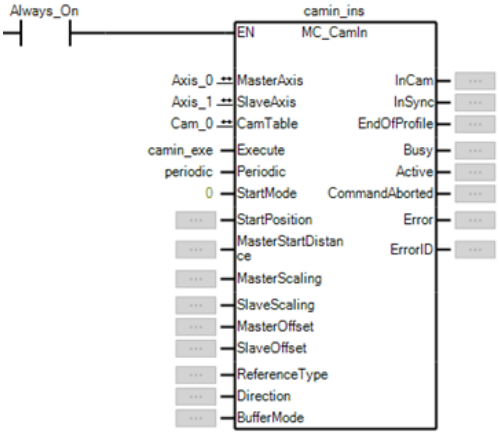
15.6.3 Cam example

The following program demonstrates how to call cam functions such as Enable, CamIn, CamOut.

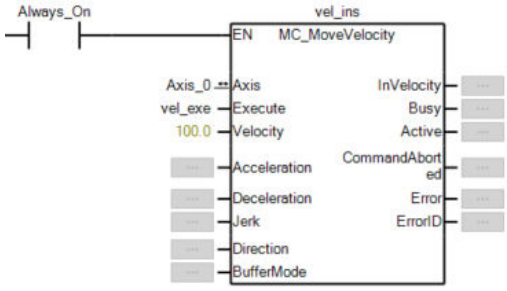
Table 15-60 Example: Cam

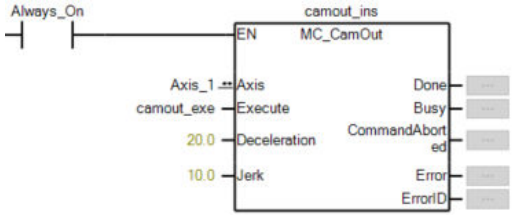
LAD/FBD	Description	STL

Network 1:
 
 	
	Enable axis	`LD Always_On`
		`FBCALL MC_Power, power_ins[0], Axis_0, power_enable, power_ins[0].Status, power_ins[0].Valid, power_ins[0].Error, power_ins[0].ErrorID`
		`FBCALL MC_Power, power_ins[1], Axis_1, power_enable, power_ins[1].Status, power_ins[1].Valid, power_ins[1].Error, power_ins[1].ErrorID`

Network 2:
 
 	
	Trigger slave axis MC_CamIn	`LD Always_On`
		
		`FBCALL MC_CamIn, camin_ins, Axis_0, Axis_1, Cam_0, camin_exe, periodic, 0, camin_ins.StartPosition, camin_ins.MasterStartDistance, camin_ins.MasterScaling, camin_ins.SlaveScaling, camin_ins.MasterOffset, camin_ins.SlaveOffset, camin_ins.ReferenceType, camin_ins.Direction, camin_ins.BufferMode, camin_ins.InCam, camin_ins.InSync, camin_ins.EndOfProfile, camin_ins.Busy, camin_ins.Active, camin_ins.CommandAborted, camin_ins.Error, camin_ins.ErrorID`

LAD/FBD	Description	STL

Network 3:
 
 	
	Trigger master axis continuous motion	`LD Always_On`
		
		`FBCALL MC_MoveVelocity, vel_ins, Axis_0, vel_exe, 100.0, Axis_0.Axis_Config.Dynamic_Default.Default_Acc, Axis_0.Axis_Config.Dynamic_Default.Default_Decel, Axis_0.Axis_Config.Dynamic_Default.Default_Jerk, vel_ins.Direction, vel_ins.BufferMode, vel_ins.InVelocity, vel_ins.Busy, vel_ins.Active, vel_ins.CommandAborted, vel_ins.Error, vel_ins.ErrorID`

Network 4:  	Decoupled slave cam	`LD Always_On` `FBCALL MC_CamOut, camout_ins, Axis_1, camout_exe, 20.0, 10.0, camout_ins.Done, camout_ins.Busy, camout_ins.CommandAborted, camout_ins.Error, camout_ins.ErrorID`

Open loop motion control(Legacy) 16

Note

Motion wizard is disabled by default

- Motion wizard is now only available for migrating projects created with STEP 7-Micro/WIN SMART V2.8.
- After migrating, you can use motion wizard to modify your project created STEP 7-Micro/WIN SMART V2.8.
- For any new projects, access the new Open loop motion control(PLCopen) (Page 769) for configuration.

The Motion wizard helps you define the parameters for the Axis of Motion and defines a set of position profiles for your application.

The S7-200 SMART CPU provides four methods of open loop motion control:

- Pulse Train Output (PTO): Built into the CPU for speed and position control. See Pulse Output instruction. (Note: There is no PTO wizard available. Use the Motion wizard instead.)
- Pulse Width Modulation (PWM): Built into the CPU for speed, position, or duty cycle control. See Pulse Output instruction.
- Axis of Motion: Built into the CPU for speed and position control of an Axis.
- Axes group of Motion: Built into the CPU for speed and position control of an Axes group.

The CPU provides four digital outputs (Q0.0, Q0.1, Q0.2 and Q0.3) that you can configure as PTO or PWM outputs by the PLS Instruction, PWM outputs by the PWM wizard, or as motion control outputs by the Motion wizard.

Note

Q0.2 is only for ST60.

Note

Only SR and ST CPU models support motion control.

CPU ST20 models only support 2D linear interpolation movement for Axes group of motion control; CPU ST30, CPU ST40 and CPU ST60 models support 2D and 3D linear interpolation movement for Axes group of motion control.

When you configure an output for PTO operation, the CPU generates a 50% duty cycle pulse train for open loop control of the speed and position for either stepper motors or servo motors. The built-in PTO function only provides the pulse train output. Your application program must supply direction and limit controls using I/O built into the PLC or provided by expansion modules.

When you configure an output for PWM operation, the CPU fixes the cycle time of the output, and your program controls the pulse width or duty cycle of the pulse. You can use the variations in pulse width to control the speed or position in your application.

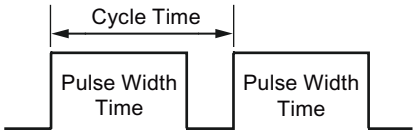
The Axis of Motion provides a single pulse train output with integrated direction control and disable outputs. It also includes programmable inputs which allow the CPU to be configured for several modes of operation, including automatic reference point seek. The Axis of Motion provides a unified solution for open loop control of the speed and position for either stepper motors or servo motors.

To simplify the use of motion control in your application, STEP 7-Micro/WIN SMART V3 provides a Motion wizard to configure Axis of Motion and Axes group of Motion, and a PWM wizard to configure PWM. The wizards generate motion instructions that you can use to provide dynamic control of speed and motion in your application. For the Axis of Motion, STEP 7-Micro/WIN SMART V3 also provides a control panel that allows you to control, monitor, and test your motion operations.

16.1 Using the PWM output

16.1.1 Using the PWM (pulse width modulation) output

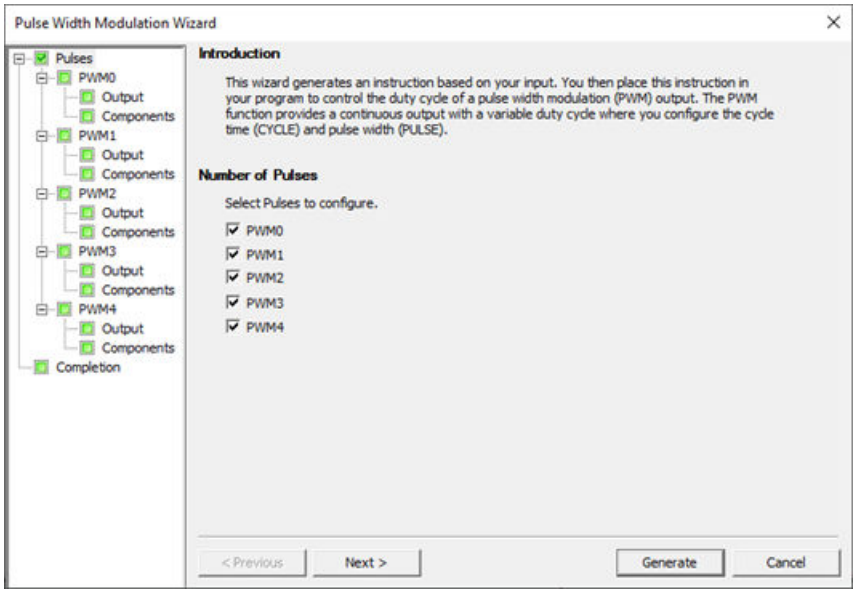
PWM provides a fixed cycle time output with a variable duty cycle. The PWM output runs continuously after being started at the specified frequency (cycle time). The pulse width is varied as required to affect the desired control. Duty cycle can be expressed as a percentage of the cycle time or as a time value corresponding to pulse width. The pulse width can vary from 0% (no pulse, always off) to 100% (no pulse, always on). See the following figure.



Since the PWM output can be varied from 0% to 100%, it provides a digital output that in many ways is analogous to an analog output. For example the PWM output can be used to control the speed of a motor from stop to full speed or it can be used to control the position of a valve from closed to full open.

16.1.2 Configuring the PWM output

To configure one of the built-in outputs for PWM control, use the PWM wizard.



Use one of the following methods to open the PWM wizard:

- Click the "PWM" button from the Wizards area of the Tools menu.

- Open the Wizards folder in the project tree and double-click "PWM", or select "PWM" and press the Enter key.

1. Select a pulse generator.
2. Change the name of a PWM channel, if required.
3. Configure the PWM channel output time base.
4. Generate project components.
5. Use the PWMx_RUN subroutine to control the duty cycle of your PWM output.

Note

PWM channels are hard-coded to specific outputs:

- The PWM0 is assigned to Q0.0.
- The PWM1 is assigned to Q0.1.
- The PWM2 is assigned to Q0.3.
- The PWM3 is assigned to Q0.2.
- The PWM3 is assigned to Q0.4.

16.1.3 PWMx_RUN subroutine

To simplify the use of Pulse Width Modulation (PWM) control in your application, STEP 7-Micro/WIN SMART V3 provides a PWM wizard (Page 881) to configure your on-board PWM generators and control the duty cycle of a PWM output.

The PWMx_RUN subroutine is used to execute PWM under program control.

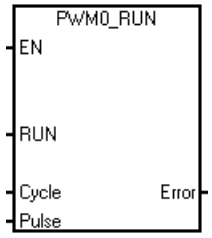
LAD / FBD	STL	Description
	`CALL PWMx_RUN, Cycle, Pulse, Error`	The PWMx_RUN subroutine allows you to control the duty cycle of the output by varying the pulse width from 0 to the pulse width of the cycle time.

Table 16-1 Parameters for the PWMx_RUN subroutine

Inputs/Outputs	Data types	Operands
Cycle, Pulse	Word	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
Error	Byte	IB, QB, VB, MB, SMB, LB, AC, *VD, *AC, *LD, Constant

The Cycle input is a word value that defines the cycle time for the pulse width modulation (PWM) output. The allowed range is from 2 to 65535 when milliseconds is the time base and 10 to 65535 when microseconds is the time base.

The Pulse input is a word value that defines the pulse width (Duty Cycle) for the PWM output. The allowed range of values is from 0 to 65535 units of the time base (microseconds or milliseconds) that was specified within the PWM wizard.

The Error is a byte value returned by the PWMx_RUN subroutine that indicates the result of execution. See the following table for a description of the possible error codes.

Table 16-2 PWMx_RUN instruction error codes

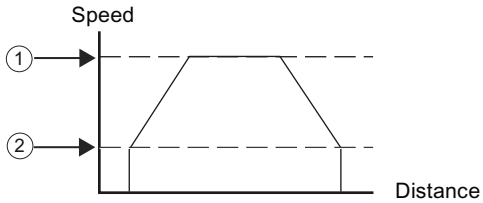
Error code	Description
0	No error, normal completion
131	Pulse generator already in use by another PWM or by a motion axis, or illegal time base change

16.2 Using motion control

The motion control built into the CPU uses an Axis of Motion to control both the speed and motion of a stepper motor or a servo motor.

Using the Axis of Motion requires expertise in the field of motion control. This chapter is not meant to educate the novice in this subject. However, it provides fundamental information that will help as you use the Motion wizard to configure the Axis of Motion for your application.

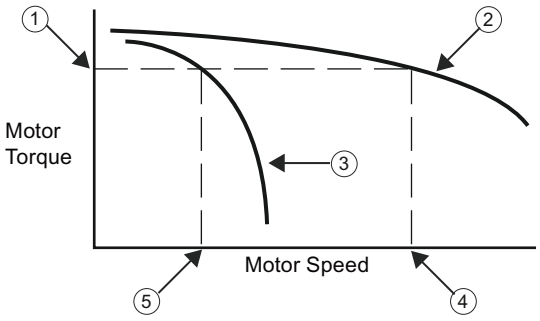
16.2.1 Maximum and start/stop speeds

	The Motion wizard prompts you for the maximum speed (MAX_SPEED) and Start/Stop Speed (SS_SPEED) for your application.

① MAX_SPEED
② SS_SPEED

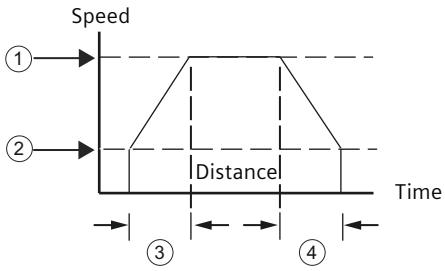
- MAX_SPEED: Enter the value for the optimum operating speed of your application within the torque capability of your motor. The torque required to drive the load is determined by friction, inertia, and the acceleration/deceleration times.
- The Motion wizard calculates and displays the minimum speed that can be controlled by the Axis of Motion based upon the MAX_SPEED that you specify.
- SS_SPEED: Enter a value within the capability of your motor to drive your load at low speeds. If the SS_SPEED value is too low, the motor and load could vibrate or move in short jumps at the beginning and end of travel. If the SS_SPEED value is too high, the motor could lose pulses on start up, and the load could overdrive the motor when attempting to stop.

Motor data sheets have different ways of specifying the start/stop (or pull-in/pull-out) speed for a motor and given load. Typically, a useful SS_SPEED value is 5% to 15% of the MAX_SPEED value. To help you select the correct speeds for your application, refer to the data sheet for your motor. The following figure shows a typical motor torque/speed curve.



① Torque required to drive the load
② Motor torque versus speed characteristic
③ Start/Stop speed versus torque: This curve moves towards lower speed as the load inertia increases.
④ Maximum speed that the motor can drive the load: MAX_SPEED should not exceed this value.
⑤ Start/Stop speed (SS_SPEED) for this load

16.2.2 Entering the acceleration and deceleration times



① MAX_SPEED

② SS_SPEED

③ ACCEL_TIME

④ DECEL_TIME

As part of the configuration, you set the acceleration and deceleration times. The default setting for both the acceleration time and the deceleration time is 1 second. Typically, motors can work with less than 1 second. See the following figure.

Note

Motor acceleration and deceleration times are determined by trial and error. You should start by entering a large value. Optimize these settings for the application by gradually reducing the times until the motor starts to stall.

Specify the following times in milliseconds:

- ACCEL_TIME: Time required for the motor to accelerate from SS_SPEED to MAX_SPEED. Default = 1000 ms
- DECEL_TIME: Time required for the motor to decelerate from MAX_SPEED to SS_SPEED. Default = 1000 ms

16.2.3 Configuring the motion profiles

A profile is a pre-defined motion description consisting of one or more speeds of movement that effect a change in position from a starting point to an ending point. You do not have to define a profile in order to use the Axis of Motion. The Motion wizard provides instructions for you to use to control moves without running a profile.

A profile is programmed in steps consisting of an acceleration/deceleration to a target speed followed by a fixed number of pulses at the target speed. In the case of single step moves or the last step in a move, there is also a deceleration from the target speed (last target speed) to stop.

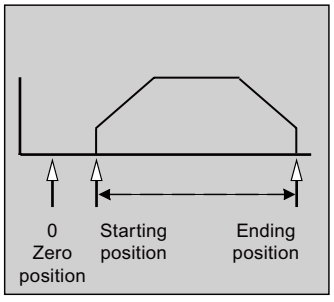
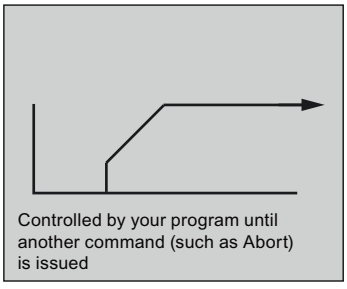
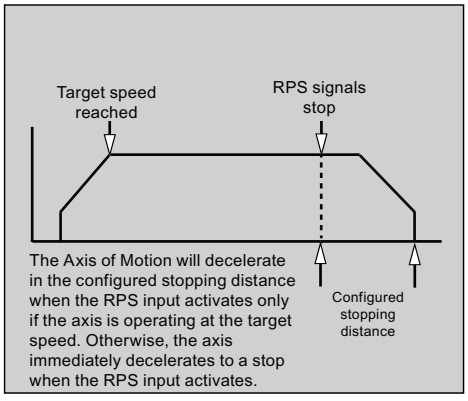
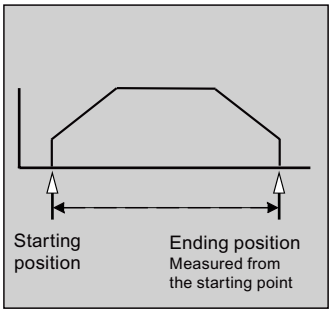
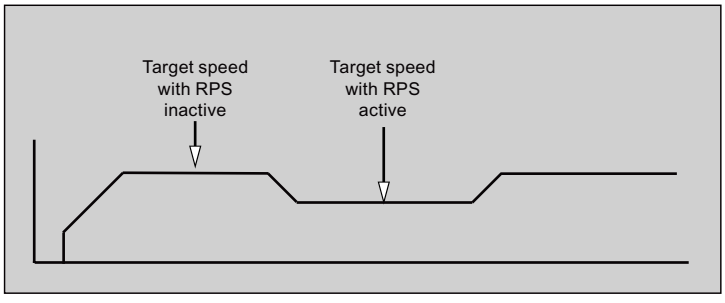
The Axis of Motion supports a maximum of 32 profiles.

Defining the motion profile

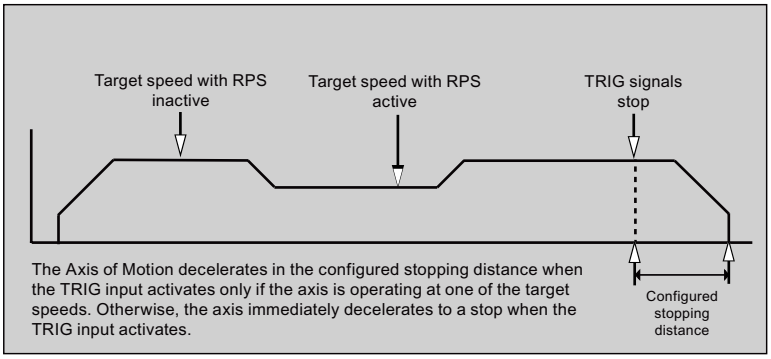
The Motion wizard guides you through a motion profile definition, where you define each motion profile for your application. For each profile, you select the operating mode and define the specifics of each individual step for the profile. The Motion wizard also allows you to define a symbolic name for each profile by simply entering the symbol name as you define the profile.

Selecting the mode of operation for the profile

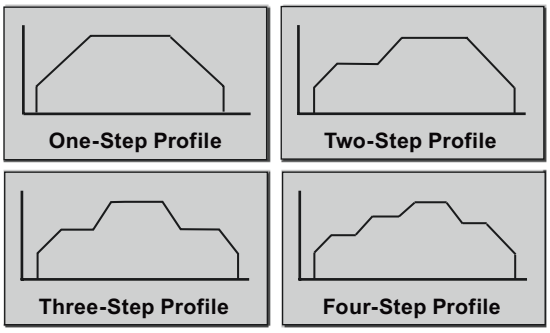
You configure the profile according to the mode of operation desired. The Axis of Motion supports absolute position, relative position, single-speed continuous rotation, and two-speed continuous rotation. The following figure shows the different modes of operation.

Mode selections for the Axis of Motion	
Absolute Position	**Single-Speed, Continuous Rotation**
	
Single-speed, Continuous Rotation, with triggered stop	**Relative Position**
	
Two-Speed, Continuous Rotation	
	

Mode selections for the Axis of Motion
Two-Speed, Continuous Rotation, with triggered stop



Creating the steps of the profile



A step is a fixed distance that a tool moves, including the distance covered during acceleration and deceleration times. The Axis of Motion supports a maximum of 16 steps in each profile.

You specify the target speed and ending position or number of pulses for each step. Additional steps are entered one at a time. The figure illustrates a one-step, two-step, three-step, and a four-step profile.

Notice that a one-step profile has one constant speed segment, a two-step profile has two constant speed segments, and so on. The number of steps in the profile matches the number of constant speed segments of the profile.

16.3 Features of motion control

Motion control provides the functionality and performance that you need for open-loop position control in up to four Axes of Motion:

- Provides high-speed control, with a range from 20 pulses per second up to 100,000 pulses per second
- Supports both jerk (S curve) or linear acceleration and deceleration
- Provides a configurable measuring system that allows you to enter data either as engineering units (such as inches or centimeters) or as a number of pulses
- Provides configurable backlash compensation
- Supports absolute, relative, and manual methods of position control

- Provides continuous operation
- Provides up to 32 motion profiles, with up to 16 speed changes per profile
- Provides four different reference-point seek modes, with a choice of the starting seek direction and the final approach direction for each sequence
- Provides support for SINAMICS V90 drives

You use STEP 7-Micro/WIN SMART V3 to create all of the configuration and profile information used by the Axis of Motion. This information is downloaded to the CPU with your program blocks.

Motion control provides six digital inputs and four digital outputs that provide the interface to your motion application. See the following table. These inputs and outputs are local to the CPU. The CPU technical specifications (Page 986) provide detailed information for the CPUs and include wiring diagrams for connecting each CPU to some of the more common motor driver/amplifier units.

Table 16-3 Motion control CPU inputs to configure

Signal	Description
STP	The STP input causes the CPU to stop the motion in progress. You can select the desired operation of STP within the Motion wizard.
RPS	The RPS (Reference Point Switch) input establishes the reference point or home position for absolute move operations. In some modes, the RPS input is also used to stop the motion in progress after traveling a specified distance.
ZP	The ZP (Zero Pulse) input helps establish the reference point or home position. Typically, the motor driver/amplifier pulses ZP once per motor revolution. Note: Only used in RP Seek Modes of 3 and 4.
LMT+ LMT-	LMT+ and LMT- inputs establish the maximum limits for motion travel. The Motion wizard allows you to configure the operation of LMT+ and LMT- inputs.
TRIG	The TRIG (Trigger) input causes the CPU, in some modes, to stop the motion in progress after traveling a specified distance.

Note

LMT+, LMT-, STP input signal abort Axes group linear interpolation movement.

If you have configured the LMT+/LMT-/STP input for any Axis in the Axes group, CPU will abort the running Axes group linear interpolation movement after receiving LMT+/LMT-/STP signal.

WARNING

Risks with changes to filter time for digital input channel

If the filter time for a digital input channel is changed from a previous setting, a new "0" level input value may need to be presented for up to 12.8 ms accumulated duration before the filter becomes fully responsive to new inputs. During this time, short "0" pulse events of duration less than 12.8 ms may not be detected or counted.

This changing of filter times can result in unexpected machine or process operation, which may cause death or serious injury to personnel, and/or damage to equipment.

To ensure that a new filter time goes immediately into effect, cycle the power of the CPU.

Table 16-4 Motion control CPU hard-coded outputs

Signal	Description
P0 P1	P0 and P1 are pulse outputs that control the movement and direction of movement of the motor.
DIS	DIS is an output used to disable or enable the motor driver/amplifier.

16.4 Using Axis of motion

16.4.1 Programming an Axis of Motion

STEP 7-Micro/WIN SMART V3 provides easy-to-use tools for configuring and programming the Axis of Motion. Simply follow these steps:

1. Configure the Axis of Motion: STEP 7-Micro/WIN SMART V3 provides a Motion wizard for creating the configuration/profile table and the position instructions. See "Configuring the Axis of Motion" for information about configuring the Axis of Motion.
2. Test the operation of the Axis of Motion: STEP 7-Micro/WIN SMART V3 provides a Motion control panel for testing the wiring of the inputs and outputs, the configuration of the Axis of Motion, and the operation of the motion profiles. See "Monitoring the Axis of Motion with the Motion control panel" for information about the Motion control panel.
3. Create the program to be executed by the CPU: The Motion wizard automatically creates the motion instructions that you insert into your program. See "Instructions created by the Motion wizard for the Axis of Motion" for information about the motion instructions. Insert the following instructions into your program:
   - To enable the Axis of Motion, insert an AXISx_CTRL instruction. Use SM0.0 (Always On) to ensure that this instruction is executed every scan.
   - To move the motor to a specific location, use an AXISx_GOTO or an AXISx_RUN instruction. The AXISx_GOTO instruction moves to a location specified by the inputs from your program. The AXISx_RUN instruction executes the motion profiles you configured with the Motion wizard.
   - To use absolute coordinates for your motion, you must establish the zero position for your application. Use an AXISx_RSEEK or an AXISx_LDPOS instruction to establish the zero position.
   - The other instructions that are created by the Motion wizard provide functionality for typical applications and are optional for your specific application.
4. Compile your program and download the system block, data block, and program block to the CPU.

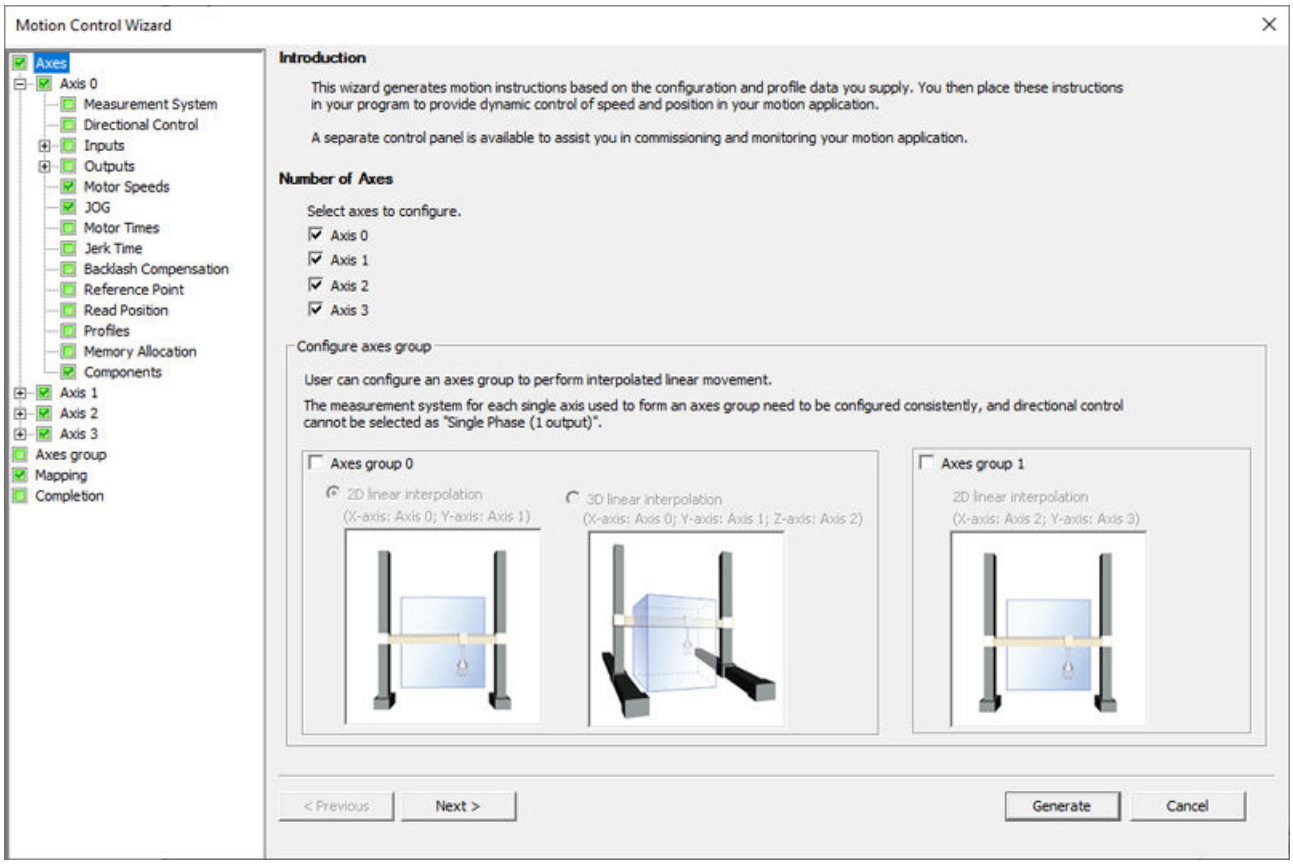
Note

Make sure to match the measurement system configuration to the pulses/revolution and distance/revolution specifications of your stepper/servo motor controller system.

16.4.2 Configuring an Axis of Motion

Configuration/profile table

You must create a configuration/profile table for the Axis of Motion in order for the CPU to control your motion application. The Motion wizard makes the configuration process quick and easy by leading you step-by-step through the configuration process. Refer to the "Advanced Topics" section of this chapter for detailed information about the configuration/profile table.



The Motion wizard also allows you to create the configuration/profile table offline. You can create the configuration without being connected to a CPU.

Starting the Motion wizard

To start the Motion wizard, either click the Tools icon in the navigation bar and then double-click the Motion wizard icon, or select the Tools > Motion wizard menu command.

Selecting type of measurement

Select the measurement system: You can select either engineering units or pulses:

- If you select pulses, no other information is required.
- If you select engineering units, enter the number of pulses required to produce one revolution of the motor (refer to the data sheet for your motor or drive), the base unit of measurement (such as inch, foot, millimeter, or centimeter), and the distance traveled in one revolution of the motor.

If you change the measurement system later, you must delete the entire configuration including any instructions generated by the Motion wizard. You must then enter your selections consistent with the new measurement system.

Configuring the input pin locations

You can program inputs related to motion control, to include STP, LMT-, LMT+, RPS, TRIG, and ZP, with a configuration in SDB0.

Table 16-5 STP, RPS, LMT+, LMT-, TRIG, and ZP pin locations

Pin definition for inputs	Description
LMT+, LMT-, STP, RPS, TRIG	Input pin 0 of the CPU can act as the LMT+, LMT-, STP, RPS, TRIG input (I0.0).
	Input pin 1 of the CPU can act as the LMT+, LMT-, STP, RPS, TRIG input (I0.1).
	Input pin 2 of the CPU can act as the LMT+, LMT-, STP, RPS, TRIG input (I0.2).
	Input pin 3 of the CPU can act as the LMT+, LMT-, STP, RPS, TRIG input (I0.3).
	Input pin 4 of the CPU can act as the LMT+, LMT-, STP, RPS, TRIG input (I0.4).
	Input pin 5 of the CPU can act as the LMT+, LMT-, STP, RPS, TRIG input (I0.5).
	Input pin 6 of the CPU can act as the LMT+, LMT-, STP, RPS, TRIG input (I0.6).
	Input pin 7 of the CPU can act as the LMT+, LMT-, STP, RPS, TRIG input (I0.7).
	Input pin 8 of the CPU can act as the LMT+, LMT-, STP, RPS, TRIG input (I1.0).
	Input pin 9 of the CPU can act as the LMT+, LMT-, STP, RPS, TRIG input (I1.1).
	Input pin 10 of the CPU can act as the LMT+, LMT-, STP, RPS, TRIG input (I1.2).
	Input pin 11 of the CPU can act as the LMT+, LMT-, STP, RPS, TRIG input (I1.3).
ZP	ZP input (I0.0), (I0.1), (I0.2), (I0.3).

Note

After you configure an input to a specific function (for example, RPS) for a particular Axis of Motion, you cannot use that input for any other Axis of Motion or for any other input, counter, or interrupt function.

Note

High-speed input wiring must use shielded cables

Use shielded cable with a maximum length of 50 m, when connecting HSC input channels I0.0, I0.1, I0.2, and I0.3.

Mapping the I/O

STEP 7-Micro/WIN SMART V3 enforces a fixed output assignment for PWM and Axis of Motion.

P0 and P1 outputs

At a minimum, any axis that is enabled has a P0 output pin configured for it. It may also have P1 output if its "Phase" configuration is anything other than "Single-phase (1 output)". Refer to the "Editing default input and output configuration" section for more information. These output pins are hard-coded to a specific output, depending on the following criteria:

<table>
<tr><th>Output</th><th colspan="2">Axis 0</th><th colspan="2">Axis 1</th><th colspan="2">Axis 2</th><th>Axis 3</th></tr>
<tr><td>P0</td><td colspan="2">Q0.0</td><td colspan="2">Q0.1</td><td colspan="2">Q0.3</td><td>Q0.2</td></tr>
<tr><td rowspan="3">P1[1]</td><td rowspan="2">Q0.2</td><td>Axis 0 is configured for "Two-phase (2 output)" or "AB quadrature phase (2 output)"</td><td rowspan="2">Q0.3</td><td rowspan="2">Axis 1 is configured for "Two-phase (2 output)" or "AB quadrature phase (2 output)"</td><td colspan="2" rowspan="3">Q1.0 : Axis 2 is configured for "Single phase (2 output)".
Note:
Axis 2 is not available for use if the "Phase" for Axis 1 is configured for "Two-phase (2 output)" or "AB quadrature phase (2 output)".</td><td rowspan="3">Q1.1: Axis 3 is configured for "Single phase (2 output)".
Note:
Axis 3 is not available for use if the "Phase" for Axis 0 is configured for "Two-phase (2 output)" or "AB quadrature phase (2 output)".</td></tr>
<tr><td>Axis 0 is configured for "Single-phase (2 output)", and you don't configure Axis 3</td></tr>
<tr><td>Q0.6</td><td>Axis 0 is configured for "Single-phase (2 output)", and you configure Axis 3</td><td>Q0.7</td><td>Axis 1 is configured for "Single-phase (2 output)".</td></tr>
<tr><td>DIS</td><td colspan="2">Q0.4</td><td colspan="2">Q0.5</td><td>Q0.6</td><td>Axis 3 is not configured.</td><td rowspan="2">Q1.3</td></tr>
<tr><td colspan="2"></td><td colspan="2"></td><td>Q1.2</td><td>Axis 3 is configured.</td></tr>
</table>

[1] If the "Phase" for any axis is configured for "Single-phase (1 output)", no P1 output is assigned.

DIS outputs

If an axis has a DIS output configured, then an entry is present in the mapping table for that output. The DIS output is also hard-coded to specific outputs based upon the following rules:

If you don't configure Axis 3:

- Axis 0: Q0.4
- Axis 1: Q0.5
- Axis 2: Q0.6

If you configure Axis 3:

- Axis 0: Q0.4
- Axis 1: Q0.5
- Axis 2: Q1.2
- Axis 3: Q1.3

Pulse output units are connected to standard 24V outputs.

Editing default input and output configuration

To change or view the default configuration of the integrated inputs/outputs select the required input/output node:

- In the "Active level" field, use the dropdown list to select the active level (High or Low). When the level is set to High, a logic 1 is read when current is flowing in the input. When the level is set to Low, a logic 1 is read when there is no current flow in the input. A logic 1 level is always interpreted as meaning the condition is active. The LEDs are illuminated when current flows in the input, regardless of activation level. (Default = active high)
- In the "System Block", "Digital Inputs" node, you can select the filter time constant (0.20 ms to 12.80 ms) for the STP, RPS, LMT+, LMT-, and TRIG inputs. Increasing the filter time constant eliminates more noise, but it also slows down the response time to a signal state change. (Default = 6.4 ms)

  WARNING

  Risks with changes to filter time for digital input channel

  If the filter time for a digital input channel is changed from a previous setting, a new "0" level input value may need to be presented for up to 12.8 ms accumulated duration before the filter becomes fully responsive to new inputs. During this time, short "0" pulse events of duration less than 12.8 ms may not be detected or counted.

  This changing of filter times can result in unexpected machine or process operation, which may cause death or serious injury to personnel, and/or damage to equipment.

  To ensure that a new filter time goes immediately into effect, a power cycle of the CPU must be applied.

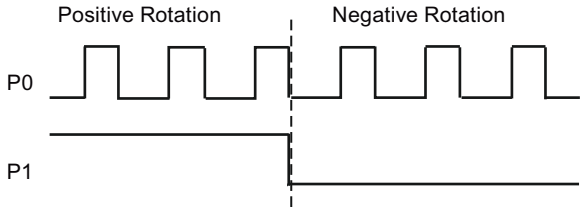
- In the "Directional Control" node, you can select the following "Phasing" modes:
  - Single phase (2 output)
  - Two-phase (2 output)
  - AB quadrature phase (2 output)
  - Single phase (1 output)

  You can also select the "Polarity" (positive or negative) of the outputs.

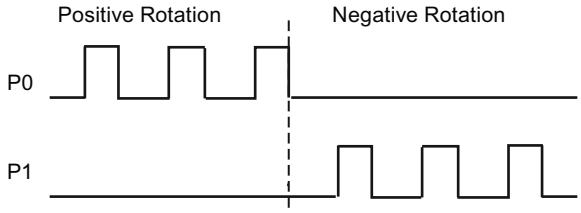
Phasing

You have four options for the "Phasing" interface to the stepper/servo drive. These options are as follows:

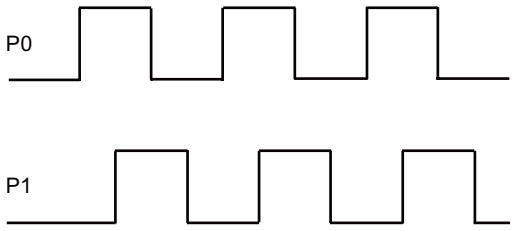
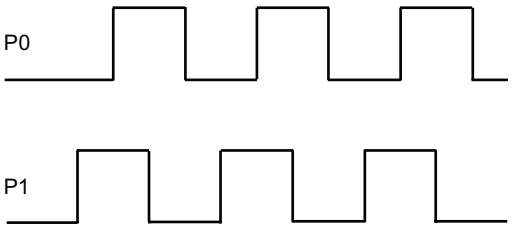
- Single phase (2 output): If you select the single phase (2 output) option, then one output (P0) controls the pulsing, and one output (P1) controls the direction. P1 is high (active) if the pulsing is in the positive direction. P1 is low (inactive) if the pulsing is in the negative direction. Single phase (2 output) is shown in the figure below (assuming positive polarity):



- Two-phase (2 output): If you select the Two-phase (2 output) option, then one output (P0) pulses for positive directions, and a different output (P1) pulses for negative directions. Two-phase (2 output) is shown in the figure below (assuming positive polarity):



- AB quadrature phase (2 output): If you select the AB quadrature phase (2 output) option, then both outputs pulse at the speed specified, but 90 degrees out-of-phase. The AB quadrature phase (2 output) is a 1X configuration, meaning a generated pulse is measured from one positive transition of the output to the next positive transition of the same output. In this case, the direction is determined by which output transitions high first. P0 leads P1 for the positive direction. P1 leads P0 for the negative direction. AB quadrature phase (2 output) is shown in the figures below (assuming positive polarity):

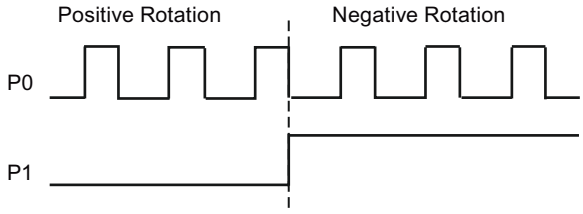
AB quadrature phase (2 output)	
(Positive polarity): positive rotation	(Positive polarity): negative rotation
	
P0 leads P1	P1 leads P0

- Single phase (1 output): If you select the single phase (1 output) option, then the output (P0) controls the pulsing. Only positive motion commands are accepted by the CPU in this mode. The Motion wizard restricts you from making illegal negative configurations when you select this mode. You can save an output if your motion application is in one direction only. Single phase (1 output) is shown in the figure below (assuming positive polarity):

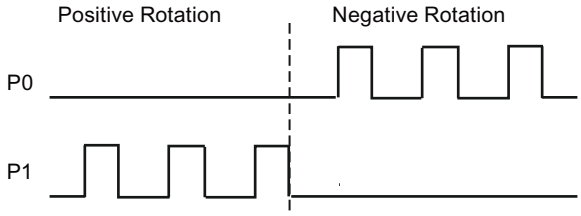
Polarity

You can switch positive and negative directions with the "Polarity" parameter. If the motor is wired in the wrong direction, this is typically done. You can avoid re-wiring the hardware by setting this parameter to negative. The negative setting changes the output operation as follows:

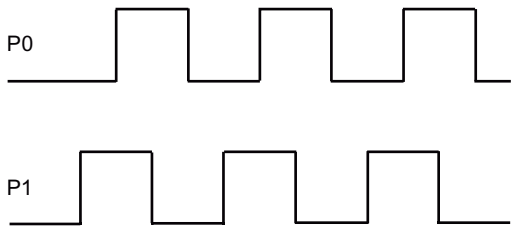
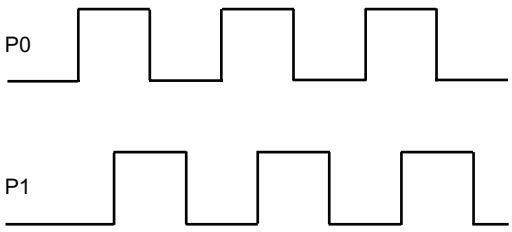
- Single phase (2 output): P1 is low (inactive) if the pulsing is in the positive direction. P1 is high (active) if the pulsing is in the negative direction. This is shown in the figure below:



- Two-phase (2 output): P0 pulses for negative directions. P1 pulses for positive directions. This is shown in the figure below:



- AB quadrature phase (2 output): P0 leads P1 for a negative direction. P1 leads P0 for a positive direction. This is shown in the figure below:

AB quadrature phase (2 output)	
(Negative polarity): positive rotation	(Negative polarity): negative rotation
	
P1 leads P0	P0 leads P1

- Single phase (1 output): Negative polarity is not allowed in this phasing mode.

The default setting for the "Directional Control" dialog is "Single phase (2 output)" and "Positive polarity".

Note

You cannot choose to which pins P0 and P1 are configured; this is hardcoded to a specific pin. Refer to the Mapping I/O section (preceding this section) for the pin mapping list.

WARNING

Safety precautions when using an Axis of Motion

The limit and stop functions in the Axis of Motion are electronic logic implementations that do not provide the level of protection provided by electromechanical controls.

Control devices and Axis of Motion functions can fail in unsafe conditions, which can result in unpredictable operation of controlled equipment. Such unpredictable operations could result in death or serious personal injury, and/or property damage.

Consider using an emergency stop function, electromechanical overrides, or redundant safeguards that are independent of the Axis of Motion and the CPU.

Configure response to physical inputs

1. Select the response to the LMT+, the LMT-, and the STP inputs.
2. Use the dropdown list to select: decelerate to a stop (default) or immediate stop.

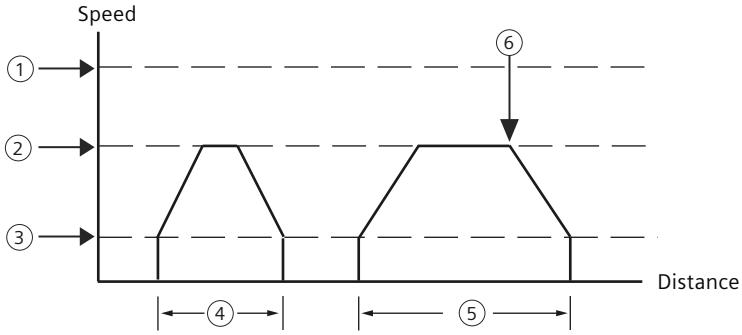
Entering maximum start and stop speed

Enter the maximum speed (MAX_SPEED) and Start/Stop Speed (SS_SPEED) for your application.

Entering jog parameters

Enter the JOG_SPEED and the JOG_INCREMENT values:

- JOG_SPEED: The JOG_SPEED (Jog speed for the motor) is the maximum speed that can be obtained while the JOG command remains active.
- JOG_INCREMENT: Distance that the tool is moved by a momentary JOG command.
  The following figure shows the operation of the Jog command. When the Axis of Motion receives a Jog command, it starts a timer. If the Jog command is terminated before 0.5 seconds has elapsed, the Axis of Motion moves the tool the amount specified in the JOG_INCREMENT at the speed defined by JOG_SPEED. If the Jog command is still active when the 0.5 seconds have elapsed, the Axis of Motion accelerates to the JOG_SPEED. Motion continues until the Jog command is terminated. The Axis of Motion then performs a decelerated stop. You can enable the Jog command either from the Motion Control Panel or with a motion instruction. A representation of a JOG operation is shown in the figure below.



1. MAX_SPEED
2. JOG_SPEED
3. SS_SPEED
4. JOG_INCREMENT: JOG command is active for less than 0.5 seconds.
5. JOG command is active for more than 0.5 seconds.
6. JOG command is terminated (Starts ramp from JOG_SPEED to SS_SPEED).

Entering acceleration and deceleration times

Enter the acceleration and deceleration times in the edit boxes.

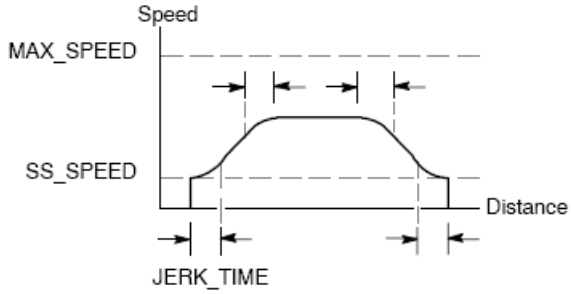
Note

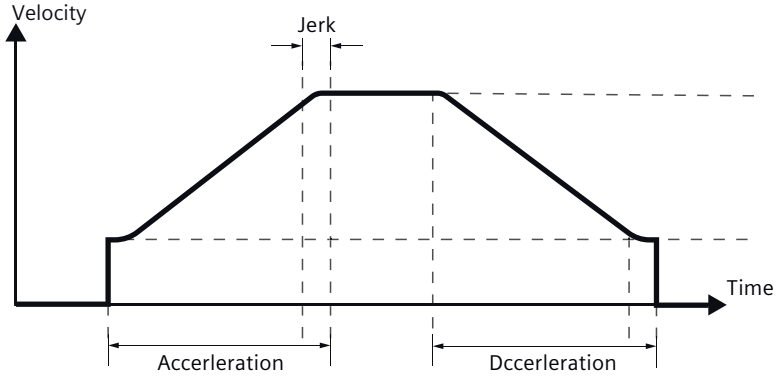
Axes group of motion

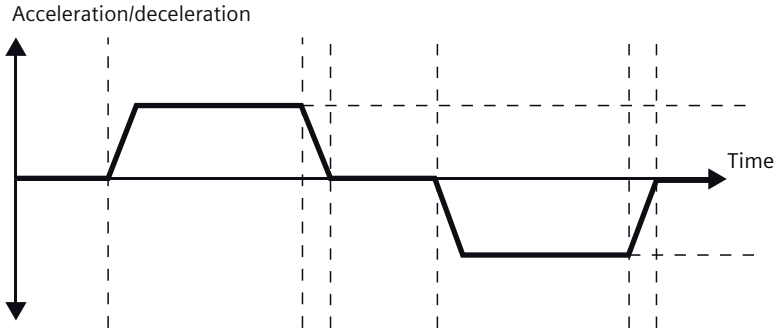
If you have configured the acceleration/decerlation times for each Axis in the Axes group, then the configuration will only take into effect for Axis of motion, but not for Axes group of motion.You can configure the acceleration/decerlation times for Axes group linear interpolation movemen in the GRP0_2/3D_MOVELINEAR (Page 963) instruction input parameter, or in the Axes group segment planning (Page 955) in the motion wizard.

Entering jerk time

Available on certain types of moves, jerk compensation provides smoother position control by reducing the jerk (rate of change) in the acceleration and deceleration parts of the motion profile. See the following figures:







Jerk compensation is also known as "S curve profiling". This compensation is applied equally to the beginning and ending portions of both the acceleration and deceleration curve. Jerk compensation is not applied to the initial and final step between zero speed and SS_SPEED.

Specify the jerk compensation by entering a time value (JERK_TIME). This is the time required for acceleration to change from zero to the maximum acceleration rate. A longer jerk time yields smoother operation with a smaller increase in total cycle time than would be obtained

by decreasing the ACCEL_TIME and DECEL_TIME. A value of 0 ms (the default value) indicates that no compensation is to be applied.

Note

A good first value for JERK_TIME is 40% of ACCEL_TIME.

Note

Jerk compensation is not available for two speed moves, manual speed change moves, aborted moves, and automatic deceleration reactions upon reaching a limit or STP input.

Note

Axes group of motion

If you have configured the jerk time for each Axis in the Axes group, then the configuration will only take into effect for Axis of motion, but not for Axes group of motion.You can configure the jerk time for Axes group linear interpolation movement in the GRP0_2/3D_MOVELINEAR (Page 963) instruction input parameter, or in the Axes group segment planning (Page 955) in the motion wizard.

Configuring the Backlash compensation

Backlash compensation: Distance that the motor must move to eliminate the backlash (slack) in the system on a direction change. Backlash compensation is always a positive value:

- Default = 0
- Choose a Reference Point search sequence to use backlash.

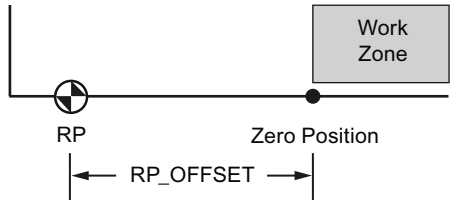
Note

Axes group of motion

The backlash compensation configuration for each Axis in the Axes group doesn't work for the motor in Axes group linear interpolation movement.

Configuring reference point and seek parameters

1. Select using a reference point or not using a reference point for your application:
   - If your application requires that movements start from or be referenced to an absolute position, you must establish a reference point (RP) or zero position that fixes the position measurements to a known point on the physical system.
   - If a reference point is used, you will want to define a way to automatically relocate the reference point. The process of automatically locating the reference point is called Reference Point Seek. Defining the Reference Point Seek process requires two steps in the wizard.
2. Enter the Reference Point seek speeds (a fast seek speed and a slow seek speed):
   - **RP_FAST** is the initial speed the module uses when performing an RP seek command. Typically, the RP_FAST value is approximately 2/3 of the MAX_SPEED value.
   - **RP_SLOW** is the speed of the final approach to the RP. A slower speed is used on approach to the RP, so as not to miss it. Typically, the RP_SLOW value is the SS_SPEED value.
3. Define the initial seek direction and the final reference point approach direction:
   - **RP_SEEK_DIR** is the initial direction for the RP seek operation. Typically, this is the direction from the work zone to the vicinity of the RP. Limit switches play an important role in defining the region that is searched for the RP. When performing a RP seek operation, encountering a limit switch can result in a reversal of the direction, which allows the search to continue. (Default = Negative)
   - **RP_APPR_DIR** is the direction of the final approach to the RP. To reduce backlash and provide more accuracy, the reference point should be approached in the same direction used to move from the RP to the work zone. (Default = Positive)
4. The Motion wizard provides advanced reference point options that allow you to specify an RP offset (RP_OFFSET), which is the distance from the RP to the zero position. See the following figure:
   - **RP_OFFSET**: Distance from the RP to the zero position of the physical measuring system.
   - Default = 0



5. The Axis of Motion provides a reference point switch (RPS) input that is used when seeking the RP. The RP is identified by a method of locating an exact position with respect to the RPS. The RP can be centered in the RPS Active zone, the RP can be located on the edge of the RPS Active zone, or the RP can be located a specified number of zero pulse (ZP) input transitions from the edge of the RPS Active zone.
6. You can configure the sequence that the Axis of Motion uses to search for the reference point. The following figure shows a simplified diagram of the default RP search sequence. Select one of the following options for the RP search sequence:
   - **RP Seek mode 0**: Does not perform a RP seek sequence
   - **RP Seek mode 1**: The RP is where the RPS input goes active on the approach from the work zone side. (Default)
   - **RP Seek mode 2**: The RP is centered within the active region of the RPS input.
   - **RP Seek mode 3**: The RP is located outside the active region of the RPS input. RP_Z_CNT specifies how many ZP (Zero Pulse) input counts should be received after the RPS becomes inactive.
   - **RP Seek mode 4**: The RP is generally within the active region of the RPS input. RP_Z_CNT specifies how many ZP (Zero Pulse) input counts should be received after the RPS becomes active.

RP seek mode 1

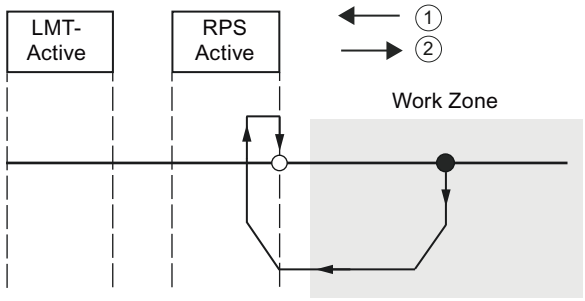


Figure 16-1 ①: RP seek direction②: RP approach direction

Note

The RPS Active region (which is the distance that the RPS input remains active) must be greater than the distance required to decelerate from the RP_FAST speed to the RP_SLOW speed. If the distance is too short, the Axis of Motion generates an error.

Defining the motion profile

1. In the motion profile definition screen, click the new profile button to enable defining the profile.
2. Choose the desired mode of operation:
   - For an absolute position profile:
     Fill in the target speed and the ending position.
     If more than one step is needed, click the new step button and fill in the step information as required.
   - For a relative position profile:
     Fill in the target speed and the ending position.
     If more than one step is needed, click the new step button and fill in the step information as required.
   - For a single-speed, continuous rotation:
     Enter the target speed value in the edit box.
     Select the direction of rotation.
     If you wish to terminate the single speed, continuous rotation move using the RPS input, click the checkbox.
     Fill in the distance to move after the RPS input goes active (RPS input must be enabled).
   - For a two-speed, continuous rotation (RPS input must be enabled):
     Enter the target speed value when RPS is inactive in the edit box.
     Enter the target speed value when RPS is active in the edit box.
     Select the direction of rotation.
     If you wish to terminate the two speed, continuous rotation move using the TRIG input, click the checkbox. (TRIG input must be enabled.)
     Fill in the distance to move after the TRIG input goes active.
3. Define as many profiles and steps as you need to perform the desired movement.

Finishing the configuration

1. After you have configured the operation of the Axis of Motion, you simply click "Generate". The Motion wizard performs the following tasks:
   - Inserts the axis configuration and profile table into the system block and data block for your CPU program
   - Creates a global symbol table for the motion parameters
   - Adds the motion instruction subroutines into the project program block for you to use in your application
2. You can run the Motion wizard again in order to modify any configuration or profile information.

Note

Because the Motion wizard makes changes to the program block, the data block, and the system block, you must download all three blocks to the CPU. Otherwise, the Axis of Motion will not have all the program components that it needs for proper operation.

16.4.3 Subroutines created by the Motion wizard for the Axis of Motion

You must ensure for each motion action that only one motion subroutine is active at a time in addition to the AXISx_CTRL, which must be active every scan. Each motion subroutine is prefixed with an "AXISx_" where "x" is the axis number channel. There are 13 motion subroutines.

Motion subroutine	Description
AXISx_CTRL (Page 904)	Provides initialization and overall control of the axis
AXISx_MAN (Page 906)	Used for manual mode operation of the axis
AXISx_GOTO (Page 907)	Commands the axis to go to a specified location
AXISx_RUN (Page 909)	Commands the axis to execute a configured motion profile
AXISx_RSEEK (Page 910)	Initiates a reference point seek operation
AXISx_LDOFF (Page 911)	Establishes a new zero position that is offset from the reference point position
AXISx_LDPOS (Page 912)	Changes the axis position to a new value
AXISx_SRATE (Page 913)	Modifies the configured acceleration, deceleration, and jerk compensation times
AXISx_DIS (Page 914)	Controls the DIS output
AXISx_CFG (Page 914)	Reads the configuration block and updates the axis setup as required
AXISx_CACHE (Page 915)	Pre-caches a configured motion profile
AXISx_RDPOS (Page 916)	Returns the current axis position
AXISx_ABSPOS (Page 917)	Reads the absolute position value from a SINAMICS V90 servo drive

Note

The motion subroutines increase the amount of memory required for your program by up to 1700 bytes. You can delete unused motion subroutines to reduce the amount of memory required. To prevent the generation of unneeded motion subroutines, uncheck the "Generate" box for each unneeded subroutine in the "Components" node of the Motion wizard. To restore generation of a particular motion subroutine, start the Motion wizard again, navigate to the "Components" node, and check the "Generate" box for the subroutine. Click the "Generate" button to rebuild the wizard-generated subroutines.

See also

Using the Motion wizard

16.4.3.1 Guidelines for using the Motion subroutines

You must ensure that only one motion subroutine is active at a time.

You can execute the AXISx_RUN and AXISx_GOTO from an interrupt routine as long as the interrupt is called cyclically. However, it is very important that you do not attempt to start a motion subroutine in an interrupt routine if the Axis of Motion is busy processing another command. If you start a subroutine in an interrupt routine, then you can use the outputs of the AXISx_CTRL subroutine to monitor when the Axis of Motion has completed the movement.

The Motion wizard automatically configures the values for the speed parameters (Speed and C_Speed) and the position parameters (Pos or C_Pos) according to the measurement system that you selected. For pulses, these parameters are DINT values. For engineering units, the parameters are REAL values for the type of unit that you selected. For example: selecting centimeters (cm) stores the position parameters as REAL values in centimeters and stores the speed parameters as REAL values in centimeters per second (cm/sec).

Some "generate" guidelines when using motion subroutines are as follows:

- Insert the AXISx_CTRL subroutine in your program, and use the SM0.0 contact to execute it every scan.
- To specify motion to an absolute position, you must first use either an AXISx_RSEEK or an AXISx_LDPOS subroutine to establish the zero position.
- To move to a specific location, based upon inputs from your program, use the AXISx_GOTO subroutine.
- To run the motion profiles you configured with the Motion wizard, use the AXISx_RUN subroutine.

16.4.3.2 AXISx_CTRL subroutine

Table 16-6 AXISx_CTRL

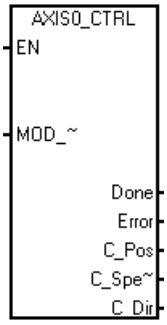
LAD / FBD	STL	Description
	**CALL AXISx_CTRL, MOD_EN, Done, Error, C_Pos, C_Speed, C_Dir**	The AXISx_CTRL subroutine (Control) enables and initializes the Axis of Motion by automatically commanding the Axis of Motion to load the configuration/profile table each time the CPU changes to RUN mode. Use this subroutine only once in your project per motion axis, and ensure that your program calls this subroutine every scan. Use SM0.0 (Always On) as the input for the EN parameter. This subroutine is also used to initialize the axis of motion in the Axes group.

Table 16-7 Parameters for the AXISx_CTRL subroutine

Inputs/Outputs	Data type	Operands
MOD_EN	BOOL	I, Q, V, M, SM, S, T, C, L, Power Flow
Done, C_Dir	BOOL	I, Q, V, M, SM, S, T, C, L
Error	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD
C_Pos, C_Speed	DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *AC, *LD

The MOD_EN parameter must be on to enable the other motion subroutines to send commands to the Axis of Motion. If the MOD_EN parameter turns off, then the Axis of Motion aborts any command that is in progress and performs a decelerated stop.

Note

Abort Axes group linear interpolation movement

To abort the Axes group linear interpolation movement, you can turn off the MOD_EN parameter of any AXISx_CTRL subroutine. The Axes group will decelerate to stop in the planned movement path.

The output parameters of the AXISx_CTRL subroutine provide the current status of the Axis of Motion.

The Done parameter turns on when the Axis of Motion completes any subroutine.

The Error parameter (Page 939) contains the result of this subroutine.

The C_Pos parameter is the current position of the Axis of Motion. Based upon the units of measurement, the value is either a number of pulses (DINT) or the number of engineering units (REAL).

The C_Speed parameter provides the current speed of the Axis of Motion. If you configured the measurement system for the Axis of Motion for pulses, C_Speed is a DINT value containing the number of pulses/second. If you configured the measurement system for engineering units, C_Speed is a REAL value containing the selected engineering units/second (REAL).

The C_Dir parameter indicates the current direction of the motor:

- Signal state of 0 = positive
- Signal state of 1 = negative

Note

The Axis of Motion reads the configuration/profile table only at power-up or when commanded to load the configuration.

- If you use the Motion wizard to modify the configuration, then the AXISx_CTRL subroutine automatically commands the Axis of Motion to load the configuration/profile table every time the CPU changes to RUN mode.
- If you use the Motion control panel to modify the configuration, clicking the Update Configuration button commands the Axis of Motion to load the new configuration/profile table.
- If you use another method to modify the configuration, then you must also issue an AXISx_CFG command to the Axis of Motion to load the configuration/profile table. Otherwise, the Axis of Motion continues to use the old configuration/profile table.

16.4.3.3 AXISx_MAN subroutine

Table 16-8 AXISx_MAN

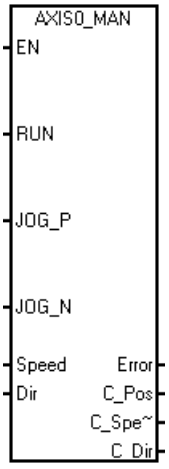
LAD / FBD	STL	Description
	**CALL AXISx_MAN, RUN, JOG_P, JOG_N, Speed, Dir, Error, C_Pos, C_Speed, C_Dir**	The AXISx_MAN subroutine (Manual Mode) puts the Axis of Motion into manual mode. This allows the motor to be run at different speeds or to be jogged in a positive or negative direction. You can enable only one of the RUN, JOG_P, or JOG_N inputs at a time.

Table 16-9 Parameters for the AXISx_MAN subroutine

Inputs/Outputs	Data type	Operands
RUN, JOG_P, JOG_N	BOOL	I, Q, V, M, SM, S, T, C, L, Power Flow
Speed	DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *AC, *LD, Constant
Dir, C_Dir	BOOL	I, Q, V, M, SM, S, T, C, L
Error	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD
C_Pos, C_Speed	DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *AC, *LD

Enable the RUN (Run/Stop) parameter to command the Axis of Motion to accelerate to the specified speed (Speed parameter) and direction (Dir parameter). You can change the value of the Speed parameter while the motor is running, but the Dir parameter must remain constant. Disabling the RUN parameter commands the Axis of Motion to decelerate until the motor comes to a stop.

Enable the JOG_P (Jog Positive Rotation) or the JOG_N (Jog Negative Rotation) parameter to command the Axis of Motion to jog in either a positive or negative direction. If the JOG_P or JOG_N parameter remains enabled for less than 0.5 seconds, the Axis of Motion issues pulses to travel the distance specified in JOG_INCREMENT. If the JOG_P or JOG_N parameter remains enabled for 0.5 seconds or longer, the Axis of Motion begins to accelerate to the specified JOG_SPEED.

The Speed parameter determines the speed when RUN is enabled. If you configured the measuring system of the Axis of Motion for pulses, the speed is a DINT value for pulses/second. If you configured the measuring system of the Axis of Motion for engineering units,

the speed is a REAL value for units/second. You can change this parameter while the motor is running.

Note

The Axis of Motion may not react to small changes in the Speed parameter, especially if the configured acceleration or deceleration time is short and the difference between the configured maximum speed and start/stop speed is large.

The Dir parameter determines the direction to move when RUN is enabled. You cannot change this value when the RUN parameter is enabled.

The Error parameter (Page 939) contains the result of this subroutine.

The C_Pos parameter contains the current position of the Axis of Motion. Based upon the units of measurement selected, the value is either a number of pulses (DINT) or the number of engineering units (REAL).

The C_Speed parameter contains the current speed of the Axis of Motion. Based upon the units of measurement selected, the value is either the number of pulses/second (DINT) or the engineering units/second (REAL).

The C_Dir parameter indicates the current direction of the motor:

- Signal state of 0 = positive
- Signal state of 1 = negative

16.4.3.4 AXISx_GOTO subroutine

Table 16-10 AXISx_GOTO

LAD / FBD	STL	Description
	CALL AXISx_GOTO, START, Pos, Speed, Mode, Abort, Done, Error, C_Pos, C_Speed	The AXISx_GOTO subroutine commands the Axis of Motion to go to a desired location.

Table 16-11 Parameters for the AXISx_GOTO subroutine

Inputs/Outputs	Data type	Operands
START	BOOL	I, Q, V, M, SM, S, T, C, L, Power Flow
Pos, Speed	DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *AC, *LD, Constant
Mode	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD, Constant
Abort, Done	BOOL	I, Q, V, M, SM, S, T, C, L

Inputs/Outputs	Data type	Operands
Error	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD
C_Pos, C_Speed	DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *AC, *LD

Turn on the EN bit to enable the subroutine. Ensure that the EN bit stays on until the DONE bit signals that the execution of the subroutine has completed.

Turn on the START parameter to send a GOTO command to the Axis of Motion. For each scan when the START parameter is on and the Axis of Motion is not currently busy, the subroutine sends a GOTO command to the Axis of Motion. To ensure that only one GOTO command is sent, use an edge detection element to pulse the START parameter on.

The Pos parameter contains a value that signifies either the location to move (for an absolute move) or the distance to move (for a relative move). Based upon the units of measurement selected, the value is either a number of pulses (DINT) or the engineering units (REAL).

The Speed parameter determines the maximum speed for this movement. Based upon the units of measurement, the value is either a number of pulses/second (DINT) or the engineering units/second (REAL).

The Mode parameter selects the type of move:

- 0: Absolute position
- 1: Relative position
- 2: Single-speed, continuous positive rotation
- 3: Single-speed, continuous negative rotation

The Done parameter turns on when the Axis of Motion completes this subroutine.

Turn on the Abort parameter to command the Axis of Motion to stop execution of this command and decelerate until the motor comes to a stop.

The Error parameter (Page 939) contains the result of this subroutine.

The C_Pos parameter contains current position of the Axis of Motion. Based upon the units of measurement, the value is either a number of pulses (DINT) or the number of engineering units (REAL).

The C_Speed parameter contains the current speed of the Axis of Motion. Based upon the units of measurement, the value is either a number of pulses/second (DINT) or the engineering units/second (REAL).

Override

A running AXISx_GOTO instruction can be overridden by another AXISx_GOTO instruction. AXISx_GOTO instruction supports override feature only when in absolute or relative mode.

You can use the override feature to change speed, position and movement mode of a running AXISx_GOTO instruction. When override is triggered,

- if the speed is changed, the CPU will accelerate or decelerate to reach to the new target speed.
- if the movement direction is changed to the opposite, the CPU will decelerate to stop before moving in the changed direction.
- any illegeal configuration in override will abort the current movement.

There are two ways to trigger override:

- the running AXISx_GOTO instruction is re-activated through the rising edge.
- activate another AIXSx_GOTO instruction when running an AXISx_GOTO intruction.

16.4.3.5 AXISx_RUN subroutine

Table 16-12 AXISx_RUN

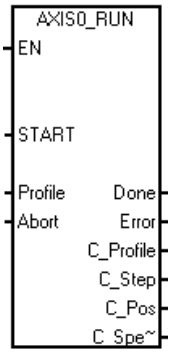
LAD / FBD	STL	Description
	**CALL AXISx_RUN, START, Profile, Abort, Done, Error, C_Profile, C_Step, C_Pos, C_Speed**	The AXISx_RUN subroutine (Run Profile) commands the Axis of Motion to execute the motion operation in a specific profile stored in the configuration/profile table.

Table 16-13 Parameters for the AXISx_RUN subroutine

Inputs/Outputs	Data type	Operands
START	BOOL	I, Q, V, M, SM, S, T, C, L, Power Flow
Profile	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD, Constant
Abort, Done	BOOL	I, Q, V, M, SM, S, T, C, L
Error, C_Profile, C_Step	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD
C_Pos, C_Speed	DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *AC, *LD

Turn on the EN bit to enable the subroutine. Ensure that the EN bit stays on until the Done bit signals that the execution of the subroutine has completed.

Turn on the START parameter to send a RUN command to the Axis of Motion. For each scan when the START parameter is on and the Axis of Motion is not currently busy, the subroutine sends a RUN command to the Axis of Motion. To ensure that only one command is sent, use an edge detection element to pulse the START parameter on.

The Profile parameter contains the number or the symbolic name for the motion profile. The "Profile" input must be between 0 - 31. If not, the subroutine will return an error.

Turn on the Abort parameter to command the Axis of Motion to stop the current profile and decelerate until the motor comes to a stop.

The Done parameter turns on when the Axis of Motion completes this subroutine.

The Error parameter (Page 939) contains the result of this subroutine.

The C_Profile parameter contains the profile currently being executed by the Axis of Motion.

The C_Step parameter contains the step of the profile currently being executed.

The C_Pos parameter contains the current position of the Axis of Motion. Based upon the units of measurement, the value is either a number of pulses (DINT) or the number of engineering units (REAL).

The C_Speed parameter contains the current speed of the Axis of Motion. Based upon the units of measurement, the value is either a number of pulses/second (DINT) or the engineering units/second (REAL).

16.4.3.6 AXISx_RSEEK subroutine

Table 16-14 AXISx_RSEEK

LAD / FBD	STL	Description
	CALL AXISx_RSEEK, START, Done, Error	The AXISx_RSEEK subroutine (Seek Reference Point Position) initiates a reference point seek operation, using the search method in the configuration/profile table. When the Axis of Motion locates the reference point and motion has stopped, the Axis of Motion loads the RP_OFFSET parameter value into the current position. In Axes group of Motion, this subroutine is also used for the reference point seek operation for each Axis in the Axes group.

Table 16-15 Parameters for the AXISx_RSEEK subroutine

Inputs/Outputs	Data type	Operands
START	BOOL	I, Q, V, M, SM, S, T, C, L, Power Flow
Done	BOOL	I, Q, V, M, SM, S, T, C, L
Error	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD

The default value for RP_OFFSET is 0. You can use the Motion wizard, the Motion Control Panel, or the AXISx_LDOFF (Load Offset) subroutine to change the RP_OFFSET value.

Turn on the EN bit to enable the subroutine. Ensure that the EN bit stays on until the Done bit signals that the execution of the subroutine has completed.

Turn on the START parameter to send a RSEEK command to the Axis of Motion. For each scan when the START parameter is on and the Axis of Motion is not currently busy, the subroutine sends a RSEEK command to the Axis of Motion. To ensure that only one command is sent, use an edge detection element to pulse the START parameter on.

The Done parameter turns on when the Axis of Motion completes this subroutine.

The Error parameter (Page 939) contains the result of this subroutine.

16.4.3.7 AXISx_LDOFF subroutine

Table 16-16 AXISx_LDOFF

LAD / FBD	STL	Description
	CALL AXISx_LDOFF, START, Done, Error	The AXISx_LDOFF subroutine (Load Reference Point Offset) establishes a new zero position that is at a different location from the reference point position. Before executing this subroutine, you must first determine the position of the reference point. You must also move the machine to the starting position. When the subroutine sends the LDOFF command, the Axis of Motion computes the offset between the starting position (the current position) and the reference point position. The Axis of Motion then stores the computed offset to the RP_OFFSET parameter and sets the current position to 0. This establishes the starting position as the zero position. In the event that the motor loses track of its position (such as on loss of power or if the motor is repositioned manually), you can use the AXISx_RSEEK subroutine to re-establish the zero position automatically.

Table 16-17 Parameters for the AXISx_LDOFF subroutine

Inputs/Outputs	Data type	Operands
START	BOOL	I, Q, V, M, SM, S, T, C, L, Power Flow
Done	BOOL	I, Q, V, M, SM, S, T, C, L
Error	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD

Turn on the EN bit to enable the subroutine. Ensure that the EN bit stays on until the Done bit signals that the execution of the subroutine has completed.

Turn on the START parameter to send a LDOFF command to the Axis of Motion. For each scan when the START parameter is on and the Axis of Motion is not currently busy, the subroutine sends a LDOFF command to the Axis of Motion. To ensure that only one command is sent, use an edge detection element to pulse the START parameter on.

The Done parameter turns on when the Axis of Motion completes this subroutine.

The Error parameter (Page 939) contains the result of this subroutine.

16.4.3.8 AXISx_LDPOS subroutine

Table 16-18 AXISx_LDPOS

LAD / FBD	STL	Description
	CALL AXISx_LDPOS, START, New_Pos, Done, Error, C_Pos	The AXISx_LDPOS subroutine (Load Position) changes the current position value in the Axis of Motion to a new value. You can also use this subroutine to establish a new zero position for any absolute move command.

Table 16-19 Parameters for the AXISx_LDPOS subroutine

Inputs/Outputs	Data type	Operands
START	BOOL	I, Q, V, M, SM, S, T, C, L, Power Flow
New_Pos, C_Pos	DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *AC, *LD
Done	BOOL	I, Q, V, M, SM, S, T, C, L
Error	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD

Turn on the EN bit to enable the subroutine. Ensure that the EN bit stays on until the Done bit signals that the execution of the subroutine has completed.

Turn on the START parameter to send a LDPOS command to the Axis of Motion. For each scan when the START parameter is on and the Axis of Motion is not currently busy, the subroutine sends a LDPOS command to the Axis of Motion. To ensure that only one command is sent, use an edge detection element to pulse the START parameter on.

The New_Pos parameter provides the new value to replace the current position value that the Axis of Motion reports and uses for absolute moves. Based upon the units of measurement, the value is either a number of pulses (DINT) or the engineering units (REAL).

The Done parameter turns on when the Axis of Motion completes this subroutine.

The Error parameter (Page 939) contains the result of this subroutine.

The C_Pos parameter contains the current position of the Axis of Motion. Based upon the units of measurement, the value is either a number of pulses (DINT) or the number of engineering units (REAL).

16.4.3.9 AXISx_SRATE subroutine

Table 16-20 AXISx_SRATE

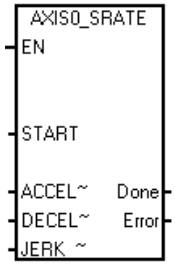
LAD / FBD	STL	Description
	**CALL AXISx_SRATE, START, ACCEL_Time, DECEL_Time, JERK_Time, Done, Error**	The AXISx_SRATE subroutine (Set Rate) commands the Axis of Motion to change the acceleration, deceleration, and jerk times.

Table 16-21 Parameters for the AXISx_SRATE subroutine

Inputs/Outputs	Data type	Operands
START	BOOL	I, Q, V, M, SM, S, T, C, L
ACCEL_Time, DECEL_Time, JERK_Time	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *AC, *LD, Constant
Done	BOOL	I, Q, V, M, SM, S, T, C, L
Error	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD

Turn on the EN bit to enable the subroutine. Ensure that the EN bit stays on until the Done bit signals that the execution of the subroutine has completed.

Turn on the START parameter to copy the new time values to the configuration/profile table and sends a SRATE command to the Axis of Motion. For each scan when the START parameter is on and the Axis of Motion is not currently busy, the subroutine sends an SRATE command to the Axis of Motion. To ensure that only one command is sent, use an edge detection element to pulse the START parameter on.

The ACCEL_Time, DECEL_Time, and JERK_Time parameters determine the new acceleration time, deceleration time, and jerk time in milliseconds (ms).

The Done parameter turns on when the Axis of Motion completes this subroutine.

The Error parameter (Page 939) contains the result of this subroutine.

16.4.3.10 AXISx_DIS subroutine

Table 16-22 AXISx_DIS

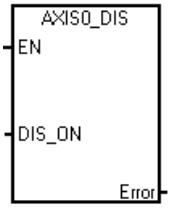
LAD / FBD	STL	Description
	**CALL AXISx_DIS, DIS_ON, Error**	The AXISx_DIS subroutine turns the DIS output of the Axis of Motion on or off. This allows you to use the DIS output for disabling or enabling a motor controller. If you use the DIS output on the Axis of Motion, then this subroutine can be called every scan or only when you need to change the value of the DIS output.

Table 16-23 Parameters for the AXISx_DIS subroutine

Inputs/Outputs	Data type	Operands
DIS_ON	BOOL	I, Q, V, M, SM, S, T, C, L, Power Flow
Error	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD

When the EN bit turns on to enable the subroutine, the DIS_ON parameter controls the DIS output of the Axis of Motion.

Note

If you have not defined a "DIS" output in the Motion wizard, the AXISx_DIS subroutine will return an error.

Note

The AXISx_DIS subroutine cannot be used to control the configured Axes group. Using AXISx_DIS subroutine in the Axes group linear interpolation movement will result in error message prompt.

The Error parameter (Page 939) contains the result of this subroutine.

16.4.3.11 AXISx_CFG subroutine

Table 16-24 AXISx_CFG

LAD / FBD	STL	Description
	CALL AXISx_CFG, START, Done, Error	The AXISx_CFG subroutine (Reload Configuration) commands the Axis of Motion to read the configuration block from the location specified by the configuration/profile table pointer. The Axis of Motion then compares the new configuration with the existing configuration and performs any required setup changes or recalculations.

Table 16-25 Parameters for the AXISx_CFG subroutine

Inputs/Outputs	Data type	Operands
START	BOOL	I, Q, V, M, SM, S, T, C, L, Power Flow
Done	BOOL	I, Q, V, M, SM, S, T, C, L
Error	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD

Turn on the EN bit to enable the subroutine. Ensure that the EN bit stays on until the Done bit signals that the execution of the subroutine has completed.

Turn on the START parameter to send a CFG command to the Axis of Motion. For each scan when the START parameter is on and the Axis of Motion is not currently busy, the subroutine sends a CFG command to the Axis of Motion. To ensure that only one command is sent, use an edge detection element to pulse the START parameter on.

The Done parameter turns on when the Axis of Motion completes this subroutine.

The Error parameter (Page 939) contains the result of this subroutine.

16.4.3.12 AXISx_CACHE subroutine

Table 16-26 AXISx_CACHE

LAD / FBD	STL	Description
	CALL AXISx_CACHE, START, Profile, Done, Error	The AXISx_CACHE subroutine (Cache Profile) commands a caching of a motion profile before the profile is executed. This allows you to pre-cache needed commands before execution. Pre-caching reduces the amount of time and provides consistency between executing a motion instruction and starting the move.

Table 16-27 Parameters for the AXISx_CACHE subroutine

Inputs/Outputs	Data type	Operands
START	BOOL	I, Q, V, M, SM, S, T, C, L, Power Flow
Profile	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD, Constant
Abort, Done	BOOL	I, Q, V, M, SM, S, T, C, L
Error, C_Profile, C_Step	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD
C_Pos, C_Speed	DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *AC, *LD

Turn on the EN bit to enable the subroutine. Ensure that the EN bit stays on until the Done bit signals that the execution of the subroutine has completed.

Turn on the START parameter to send a CACHE command to the Axis of Motion. For each scan when the START parameter is on and the Axis of Motion is not currently busy, the subroutine sends a CACHE command to the Axis of Motion. To ensure that only one command is sent, use an edge detection element to pulse the START parameter on.

The Profile parameter contains the number or the symbolic name for the motion profile. The "Profile" input must be between 0 - 31. If not, the subroutine will return an error.

The Done parameter turns on when the Axis of Motion completes this subroutine.

The Error parameter (Page 939) contains the result of this subroutine.

16.4.3.13 AXISx_RDPOS subroutine

Table 16-28 AXISx_RDPOS

LAD / FBD	STL	Description
	CALL AXISx_RDPOS, Error, I_Pos	The AXISx_RDPOS subroutine returns the current motion axis position.

Table 16-29 Parameters for the AXISx_RDPOS subroutine

Inputs/Outputs	Data type	Operands
Error	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD
I_Pos	DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *AC, *LD

Turn on the EN bit to enable the subroutine.

The Error parameter (Page 939) contains the result of this subroutine.

The I_Pos parameter contains the current motion axis position.

Note

Execution of this command returns the actual current position of the axis. Position status values provided in other motion subroutines, such as AXISx_CTRL, AXISx_GOTO, and so forth, are updated periodically. Therefore, the position values reported by those commands and the position value reported by this command may differ somewhat as a result and is normal.

16.4.3.14 AXISx_ABSPOS subroutine

Table 16-30 AXISx_ABSPOS

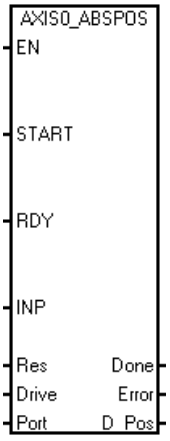
LAD / FBD	STL	Description
	**CALL AXISx_ABSPOS, START, RDY, INP, Res, Drive, Port, Done, Error, D_Pos**	The AXISx_ABSPOS subroutine reads the absolute position from certain Siemens servo drives, such as the V90. You read the absolute position value in order to update the current position value in the Axis of Motion. This capability is supported when using a SINAMICS V90 servo drive combined with a SIMOTICS-1FL6 servo motor that has an absolute encoder installed.

Table 16-31 Parameters for the AXISx_ABSPOS subroutine

Inputs/Outputs	Data type	Operands
START	BOOL	I, Q, V, M, SM, S, T, C, L, Power Flow
RDY, INP	BOOL	I, Q, V, M, SM, S, T, C, L
Res	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *AC, *LD, Constant
Drive	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD, Constant
Port	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD, Constant
Done	BOOL	I, Q, V, M, SM, S, T, C, L
Error	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD
D_Pos	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *AC, *LD

Turn on the EN bit to enable the subroutine. Ensure that the EN bit stays on until the DONE bit signals that the execution of the subroutine has completed.

Turn on the START parameter to obtain the current absolute position from the specified drive. To ensure that only one operation to read the current position is performed, use an edge detection element to pulse the START parameter on.

The RDY parameter indicates the readiness state of the servo drive, which is typically provided by a digital output signal from the drive. This subroutine will read the absolute position from the drive only if this parameter is on.

The INP parameter indicates the standstill state of the motor, which is typically provided by a digital output signal from the drive. This subroutine will read the absolute position from the drive only if this parameter is on.

The Res parameter must be set to the resolution of the absolute encoder connected to your servo motor. For example, the single turn resolution of a SIMOTICS S-1FL6 servo motor with absolute encoder is 20 bits or 1048576.

Set the Drive parameter to match the RS485 address of the servo drive to be accessed by this subroutine. Valid addresses of individual drives are 0 to 31.

Set the Port parameter to designate the CPU port to be used to communicate with the servo drive:

- 0: Onboard RS485 port (Port 0)
- 1: RS485/RS232 signal board (if present, Port 1)

The Done parameter turns on when the subroutine's work is complete.

The Error parameter (Page 939) contains the result of this subroutine.

The D_Pos parameter contains the current absolute position returned by the servo drive.

Note

To use this subroutine, you must configure the measurement system setting for this Axis of Motion to Engineering Units.

Note

Additional subroutines

The Motion wizard also creates the subroutines ABSPOS_SBR and ABSPOS_INT when you enable read position in the wizard configuration to read an absolute position from a drive.

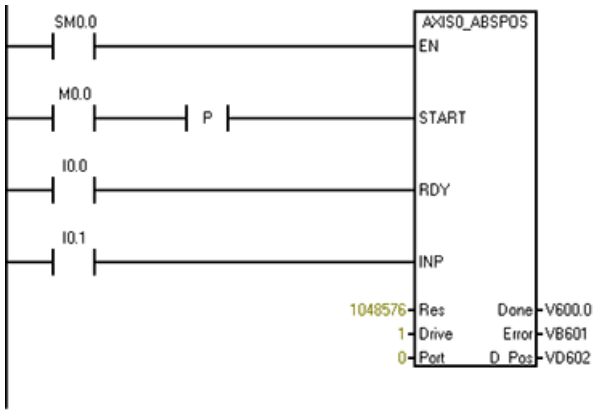
16.4.4 Using the AXISx_ABSPOS subroutine to read the absolute position from a SINAMICS servo drive

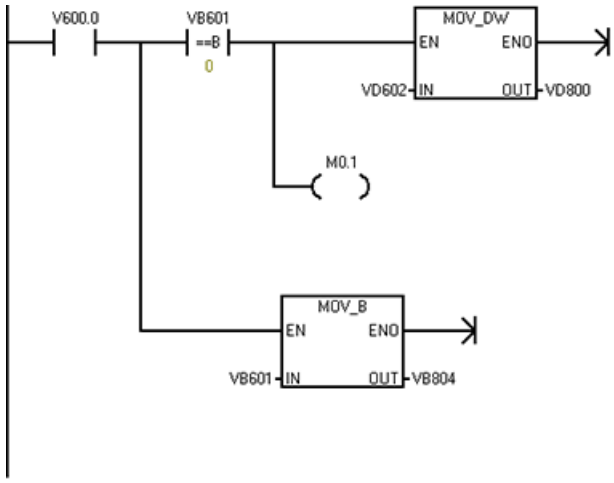
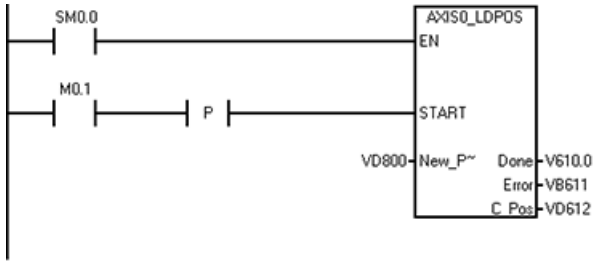
The following sections provide information about how to use the AXISx_ABSPOS subroutine in your project to read the absolute position from a SINAMICS V90 servo drive.

16.4.4.1 AXISx_ABSPOS and AXISx_LDPOS subroutines usage examples

The absolute position is valid only after successful completion of the AXISx_ABSPOS subroutine (Done parameter = ON and Error parameter = "no error") when executed with the START parameter on. Since the Error and D_Pos parameters revert to default values when the subroutine is executed with the START input off, you must include instructions in your program to capture the valid absolute position value after completion of the subroutine.

Table 16-32 Example: Using the AXISx_ABSPOS subroutine to read the absolute position from a SINAMICS V90 servo drive

LAD/FBD	Description	STL
Network 1:  	Read the servo position from the drive.	**LD SM0.0** **= L60.0** **LD M0.0** **EU** **= L63.7** **LD I0.0** **= L63.6** **LD I0.1** **= L63.5** **LD L60.0** **CALL AXIS0_AB-SPOS, L63.7, L63.6, L63.5, 1048576,1, 0, V600.0,** **VB601, VD602**

Network 2:  	When the operation is done, capture the error code and also capture the servo position, if no error.	**LD V600.0** **LPS** **AB= VB601, 0** **MOVD VD602, VD800** **= M0.1** **LPP** **MOVB VB601, VB804**
Network 3:  	Update the current position in this Axis of Motion with the captured servo position value.	**LD SM0.0** **= L60.0** **LD M0.1** **EU** **= L63.7** **LD L60.0** **CALL AXIS0_LDPOS, L63.7, VD800, V610.0, VB611, VD612**

16.4.4.2 Interconnections

Digital I/O

Refer to the section "Connection examples with PLCs" in the *SINAMICS V90 / SIMOTICS S-1FL6 Operating Instructions* document to find wiring diagrams for connection of the suggested digital control signals between an S7-200 SMART CPU and a V90 servo drive.

Communications

The AXISx_ABSPOS subroutine obtains the position data from the drive using serial communications on the RS485 link between the two devices. Therefore, connect a cable between the RS485 port on the S7-200 SMART CPU (or optionally the S7-200 SMART CM01 signal board, if your CPU model supports it) and the RS485 port on the V90 servo drive.

Refer to the appropriate sections of the *S7-200 SMART System Manual* and the *SINAMICS V90 / SIMOTICS S-1FL6 Operating Instructions* documents for descriptions of the RS485 ports on the S7-200 SMART CPU and V90 servo drive.

16.4.4.3 Commissioning

Control mode

"PTI" mode is the drive control mode setting that allows movement speed and distance to be controlled from an external pulse train. The default control mode in the V90 servo drive is basic "PTI" mode, but you can check the mode setting by reading the value of parameter "p29003" and verifying that the value = "0". It is possible to use compound control modes (PTI/S and PTI/T) with the pulse train output from the S7-200 SMART CPU. These are advanced features and are not within the scope of this document. For assistance with these features, refer to the *SINAMICS V90 / SIMOTICS S-1FL6 Operating Instructions* document.

Setpoint pulse input channel

For correct operation with the digital outputs of the S7-200 SMART CPU, you must select the "24 V DC single end pulse train input" setting for the setpoint pulse input channel parameter (parameter "p29014" = 1) in the V90 servo drive.

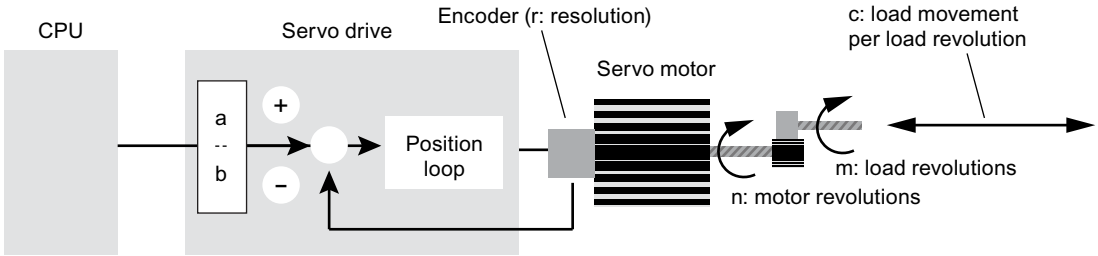
Setpoint pulse train input format

Ensure that the CPU's Axis of Motion output phasing and polarity settings (established in the "Directional Control" dialog of the STEP 7-Micro/WIN SMART V3 Motion wizard) are consistent with the V90 servo drive's setpoint pulse train input format setting (parameter "p29010").

Common engineering units basis

When using a motion axis on the S7-200 SMART CPU to control the movement speed and distance of a servo motor, you must establish a common definition of the engineering units between the Axis of Motion (CPU) and the drive.

The following diagram shows the elements of a motion system:



To establish a common engineering unit definition between the CPU and the servo drive, you must consider the following motion system variables when commissioning your system:

Electronic gearing

In the V90 servo drive, the "a" and "b" values determine the drive's electronic gear ratio, a feature that allows a frequency conversion on the pulse train issued from the CPU. Since the maximum pulse frequency issued from an Axis of Motion in the S7-200 SMART CPU is 100 kHz, while the encoder resolution of SIMOTICS S-1FL6 servo motors installed with absolute encoders is 2^20 pulses per revolution, use of the drive's electronic gear feature is likely, in many applications, to achieve higher motor speeds. For example, to achieve a 10x increase in the setpoint pulse frequency within the servo drive compared to the frequency of the CPU pulse train supplied to the drive, then you must set the electronic gear ratio to "10:1".

In the V90 servo drive, setting parameter "p29012[0]" establishes the numerator of the electronic gearing ratio ("a"), while setting parameter "p29013" establishes the denominator of the ratio ("b"). Also, when using electronic gearing, set the parameter "p29011" value to "0". The valid range for the electronic gear ratio (a / b) in the V90 servo drive is between "0.02" and "200".

Refer to the "Electronic Gear Ratio" section of the *SINAMICS V90 / SIMOTICS S-1FL6 Operating Instructions* document for more information.

Mechanical factors

The "m" and "n" values establish the mechanical relationship between a load revolution and a motor revolution, applicable when a gearing mechanism is used. When the V90 servo drive is in "PTI" control mode, its internal mechanical gearing ratio parameters are fixed at "1:1", but the physical "m" and "n" values are important in establishing the correct engineering unit conversion factors for the Axis of Motion, as shown below.

The "c' value establishes the relationship between load movement in the specified engineering unit, and load revolutions. "20 cm of load movement per load revolution" and "360 degrees of load movement per load revolution" are examples of this conversion factor.

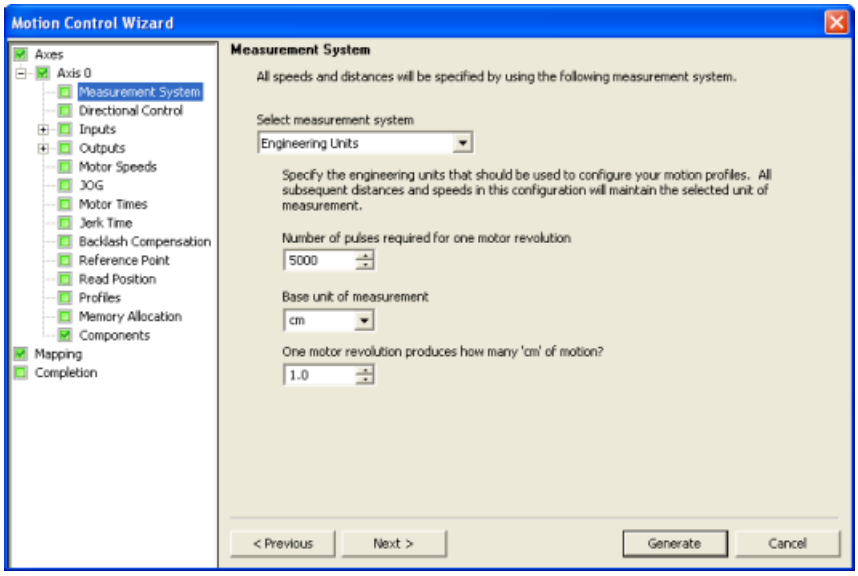
Encoder resolution

The "r" value is the resolution of the absolute encoder in your servo motor. As stated above, the encoder resolution of SIMOTICS S-1FL6 servo motors installed with absolute encoders is 2^20 pulses per revolution or "1048576". When the V90 servo drive is paired with a motor containing an absolute encoder, the drive automatically detects the encoder type and obtains its resolution. However, in your program, you must specify this resolution value in the AXISx_ABSPOS subroutine's "Res" input parameter and also in one of the engineering unit conversion factor calculations shown below.

Measurement system settings in the Motion wizard

When using the STEP 7-Micro/WIN SMART V3 Motion wizard to configure the measurement system for a CPU Axis of Motion, you must assign the three conversion settings:

- First setting: Relates CPU pulses to motor revolutions
- Second setting: Establishes the base engineering unit name
- Third setting: Relates motor revolutions to load movement



Setting 1: "Number of pulses required for one motor revolution"

This setting defines the relationship between CPU pulses and motor revolutions. The relevant equation that yields the correct value for this setting follows:

(1) Number of pulses required for one motor revolution = r * (b / a)

where, "r" = encoder resolution, expressed as encoder pulses per motor revolution,
"a" and "b" = electronic gearing (E-gear) ratio parameters ("a" = value of V90 parameter "p29012[0]" and "b" = value of V90 parameter "p29013")

For example, if the desired E-gear ratio is "128:1" and the motor's absolute encoder resolution is 2^20 or "1048576", then:

"Number of pulses required for one motor revolution" = **1048576 * (1 / 128) = 8192**

Setting 2: "Base unit of measurement"

This setting establishes the base engineering unit name for speed and distance settings throughout the Motion wizard. To avoid confusion, the selection should match the engineering unit relevant at the load. For example, if load movement and speed is to be expressed in "cm" and "cm / second", then the "cm" selection should be chosen for this setting.

Setting 3: "One motor revolution produces how many "xxx" of motion?"

This setting defines the relationship between motor revolutions and load movement in the defined engineering unit (for example, cm and degrees). The relevant equation that yields the correct value for this setting is as follows:

(2) One motor revolution produces how many "xxx" of motion = c * (m / n)

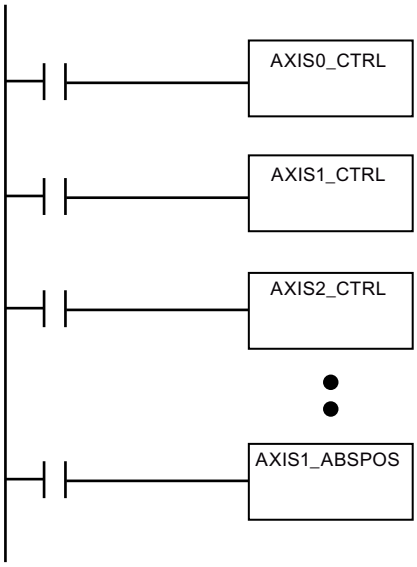
where, "c" = load movement (in the defined engineering unit) per load revolution,
"m/n" = external gearing ratio expressed as load revolutions per motor revolution

For example, if the mechanical gear ratio is "1:2" and the load movement per load revolution is 10 cm, then:

"One motor revolution produces how many cm of motion" = **10 * (1 / 2) = 5**

16.4.4.4 Important facts to know

- Do not call the AXISx_ABSPOS subroutine from within an interrupt routine or from a subroutine called within an interrupt routine.
- If you have configured multiple Axes of Motion in your CPU project, ensure that the AXISx_CTRL subroutines for all axes are executed prior to executing the first AXISx_ABSPOS subroutine for any axis. The AXISx_CTRL subroutine contains code to initialize the V memory area used commonly by all instances of the AXISx_ABSPOS subroutine in your program to manage the communications with the servo drive.



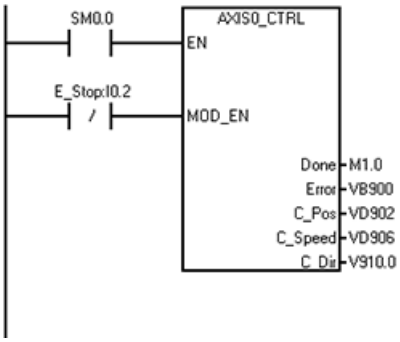
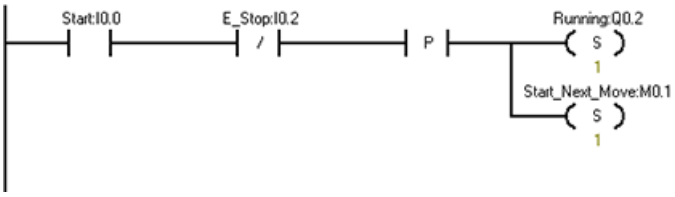
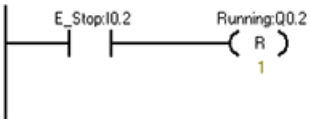
- If you configure your motion axis measurement system to the "relative pulses" setting instead of the "engineering units" setting, you can still use the AXISx_ABSPOS subroutine to return position information from the V90 servo drive. Note, however, that the position value returned in the "D_pos" parameter of the subroutine will then be of type DINT and is the actual position value reported by the servo drive (there are no engineering unit conversions performed on the value).

16.4.5 Axis of Motion example programs

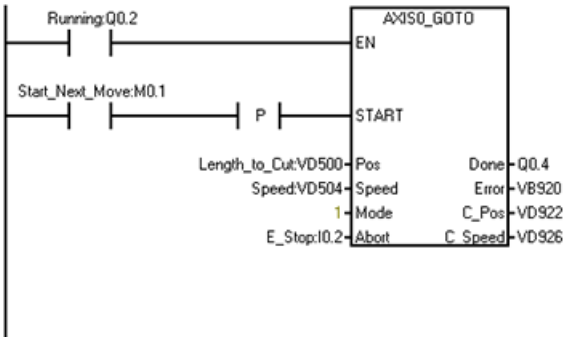
16.4.5.1 Axis of Motion simple relative move (cut-to-length application) example

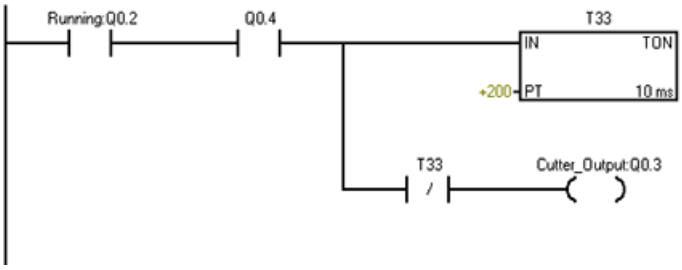
The example program shows a simple relative move that uses the AXISx_CTRL and AXISx_GOTO subroutines to perform a cut-to-length operation. This program does not require an RP seek mode or a motion profile, and the length can be measured in either pulses or engineering units. Enter the length (VD500) and target speed (VD504). When I0.0 (Start) turns on, the machine starts. When I0.1 (Stop) turns on, the machine finishes the current operation and stops. When I0.2 (E_Stop) turns on, the machine aborts any motion and immediately stops.

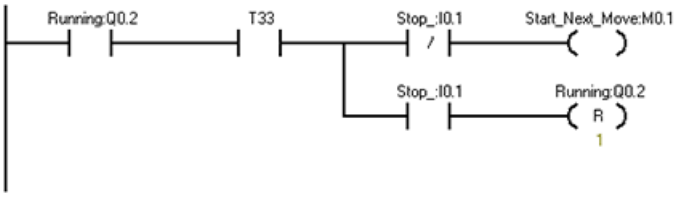
Table 16-33 Example: Axis of Motion simple relative move (cut-to-length application)

LAD/FBD	Description	STL
Network 1:  	Control instruction	LD SM0.0 = L60.0 LDN I0.2 = L63.7 LD L60.0 CALL AXIS0_CTRL, L63.7, M1.0, VB900, VD902, VD906, V910.0
Network 2:  	Start puts machine into automatic mode.	LD I0.0 AN I0.2 EU S Q0.2, 1 S M0.1, 1
Network 3:  	E_Stop: Stops immediately and turns off automatic mode.	LD I0.2 R Q0.2, 1

LAD/FBD	Description	STL

Network 4:
 
 	
	1. Move to a certain point:
	2. Enter the length to cut.
	3. Enter the target speed into Speed.
	4. Set the mode to 1 (Relative mode).
		`LD Q0.2`
		`= L60.0`
		`LD M0.1`
		`EU`
		`= L63.7`
		`LD L60.0`
		`CALL AXIS0_GOTO, L63.7, VD500, VD504, 1, I0.2, Q0.4, VB920, VD922, VD926`

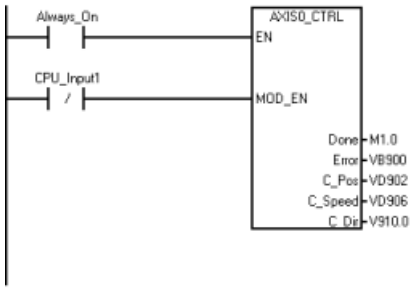
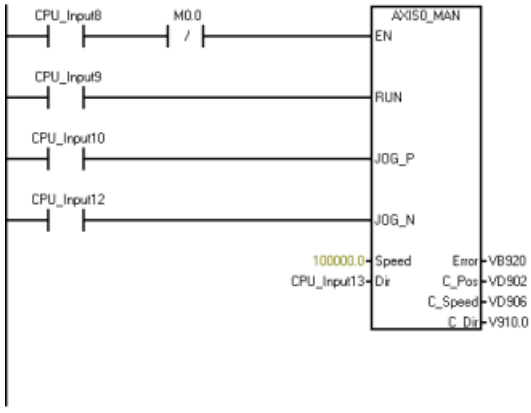
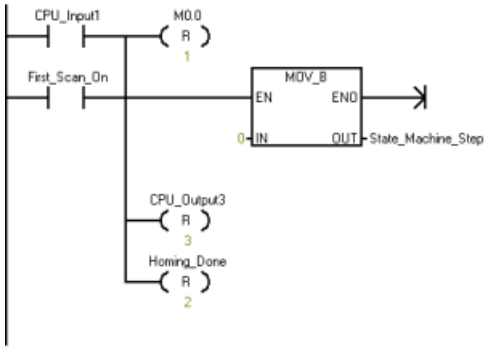
Network 5:
 
 	
	When in position, turn on the cutter for 2 seconds to finish the cut.	`LD Q0.2`
		`A Q0.4`
		`TON T33, +200`
		`AN T33`
		`= Q0.3`

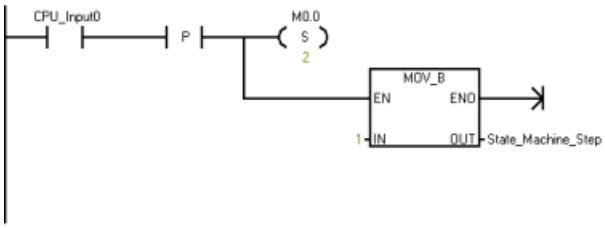
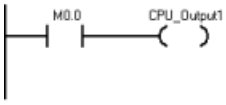
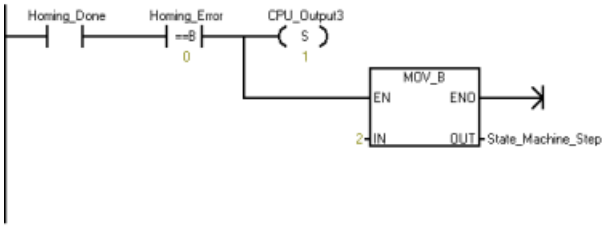
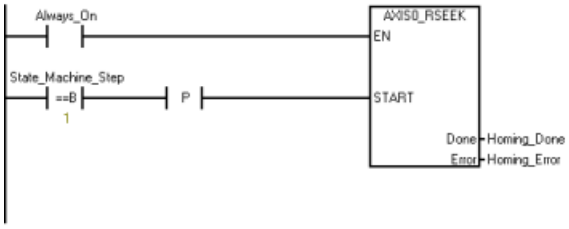
Network 6:
 
 	
	When the cut is finished, then restart unless the Stop is active.	`LD Q0.2`
		`A T33`
		`LPS`
		`AN I0.1`
		`= M0.1`
		`LPP`
		`A I0.1`
		`R Q0.2, 1`

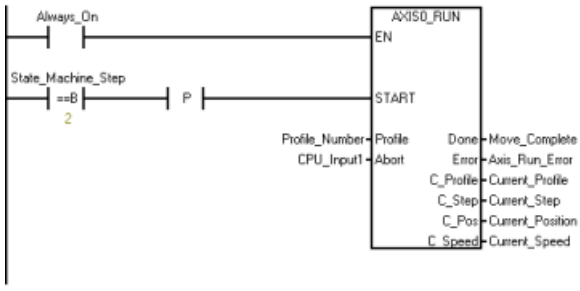
16.4.5.2 Axis of Motion AXISx_CTRL, AXISx_RUN, AXISx_SEEK, and AXISx_MAN example

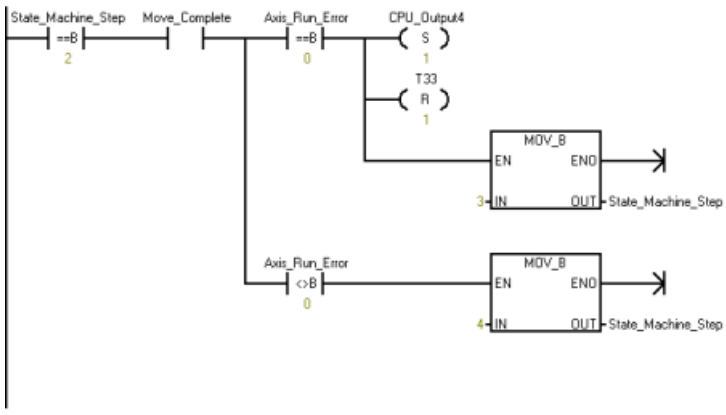
This program provides an example of the AXISx_CTRL, AXISx_RUN, AXISx_RSEEK, and AXISx_MAN subroutines. You must configure the RP seek mode and a motion profile.

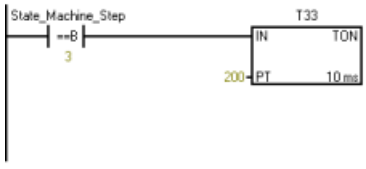
Table 16-34 Example: Axis of Motion AXISx_CTRL, AXISx_RUN, AXISx_SEEK, and AXISx_MAN subroutines application

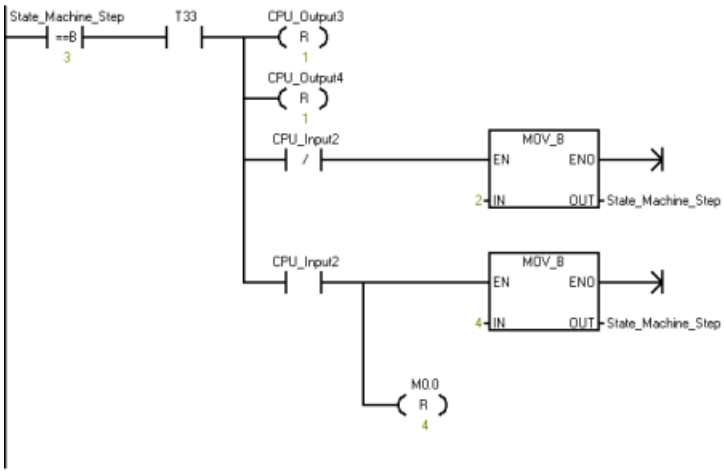
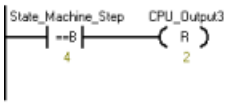
LAD/FBD	Description	STL
Network 1  	Enable the axis by turning CPU_Input1 off. Symbol and Address: [1] • Always_On = SM0.0 • AXIS0_CTRL = SBR1 • CPU_Input1 = I0.1	LD Always_On = L60.0 LDN CPU_Input1 = L63.7 LD L60.0 CALL AXIS0_CTRL, L63.7, M1.0, VB900, VD902, VD906, V910.0
Network 2  	Move the axis to a known position using the Jog command. You can now move the axis manually. Symbol and Address: [1] • AXIS0_MAN = SBR2 • CPU_Input10 = I1.2 • CPU_Input12 = I1.4 • CPU_Input13 = I1.5 • CPU_Input8 = I1.0 • CPU_Input9 = I1.1	LD CPU_Input8 AN M0.0 = L60.0 LD CPU_Input9 = L63.7 LD CPU_Input10 = L63.6 LD CPU_Input12 = L63.5 LD L60.0 CALL AXIS0_MAN, L63.7, L63.6, L63.5, 100000.0, CPU_Input13, VB920, VD902, VD906, V910.0
Network 3  	Reset the process. Set the initial step to "0". Symbol and Address: [1] • CPU_Input1 = I0.1 • CPU_Output3 = Q0.3 • First_Scan_On = SM0.1 • Homing_Done = M1.1 • State_Machine_Step = VB1500	LD CPU_Input1 O First_Scan_On R M0.0, 1 MOVB 0, State_Machine_Step R CPU_Output3, 3 R Homing_Done, 2

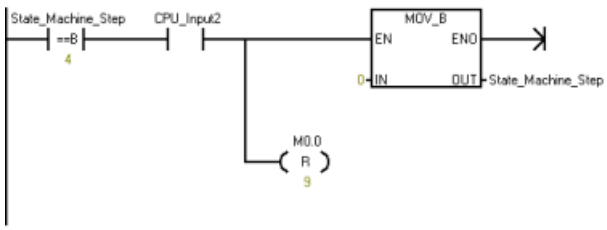
LAD/FBD	Description	STL
Network 4  	Start the process. When CPU_Input0 is toggled from off to on, the State_Machine_Step is set to "1". Symbol and Address: [1] • CPU_Input0 = I0.0 • State_Machine_Step = VB1500	**LD CPU_Input0** **EU** **S M0.0, 2** **MOVB 1, State_Machine_Step**
Network 5  	This network turns CPU_Input1 on. Symbol and Address: [1] • CPU_Input1 = I0.1	**LD M0.0** **= CPU_Input1**
Network 6  	When the homing is complete, move to the next step. Symbol and Address: [1] • CPU_Output3 = Q0.3 • Homing_Done = M1.1 • Homing_Error = VB930 • State_Machine_Step = VB1500	**LD Homing_Done** **AB= Homing_Error** **S CPU_Output3** **MOVB 2, State_Machine_Step**
Network 7  	When the state machine is in Step 1, the system automatically homes the axis. If there is a Homing error, the Homing_Error output displays the error code. Symbol and Address: [1] • Always_On = SM0.0 • AXIS0_RSEEK = SBR5 • Homing_Done = M1.1 • Homing_Error = VB930 • State_Machine_Step = VB1500	**LD Always_On** **= L60.0** **LDB State_Machine_Step, 1** **EU** **= L63.7** **LD L60.0** **CALL AXIS0_RSEEK, L63.7, Homing_Done, Homing_Error**

LAD/FBD	Description	STL
Network 8  	When the State Machine enters Step 2, it executes a move of the selected profile. Symbol and Address: [1] • Always_On = SM0.0 • AXIS0_RUN = SBR4 • Axis_Run_Error = VB940 • CPU_Input1 = I0.1 • Current_Position = VD914 • Current_Profile = VB941 • Current_Speed = VD948 • Current_Step = VB942 • Move_Complete = M1.2 • Profile_Number = VB228 • State_Machine_Step = VB1500	**LD Always_On = L60.0 LDB= State_Machine_Step, 2 EU = L63.7 LD L60.0 CALL AXIS0_RUN, L63.7, Profile_Number, CPU_Input1, Move_Complete, Axis_Run_Error, Current_Profile, Current_Step, Current_Position, Current_Speed**

Network 9
 
 	
	When the State Machine is in Step 2 and completes the move, you evaluate the error status. If there is no error, the State Machine transitions to Step 3. If there is an error, the State Machine transitions to Step 4 for error handling.
	
	Symbol and Address: [1]
	• Axis_Run_Error = VB940
	• CPU_Output4 = Q0.4
	• Move_Complete = M1.2
	• State_Machine_Step = VB1500
		**LDB= State_Machine_Step, 2
		A Move_Complete
		LPS
		AB= Axis_Run_Error, 0
		S CPU_Output4, 1
		R T33, 1
		MOVB 3, State_Machine_Step
		LPP
		AB<> Axis_Run_Error, 0
		MOVB 4, State_Machine_Step**

Network 10  	Wait for Step 3. Symbol and Address: [1] • State_Machine_Step = VB1500	**LDB= State_Machine_Step, 3 TON T33, 200**

LAD/FBD	Description	STL
Network 11  	If the State Machine moves to Step 3, wait for 2 s. Then, evaluate the status of the switches, and restart the move or stop. Symbol and Address: [1] • CPU_Input2 = I0.2 • CPU_Output3 = Q0.3 • CPU_Output4 = Q0.4 • State_Machine_Step = VB1500	**LDB= State_Machine_ Step, 3** **A T33** **LPS** **R CPU_Output3, 1** **R CPU_Output4, 1** **AN CPU_Input2** **MOVB 2, State_Machine_ Step** **LPP** **A CPU_Input2** **MOVB 4, State_Machine_ Step** **R M0.0, 4**
Network 12  	If the State Machine moves to Step 4, clear the outputs. Symbol and Address: [1] • CPU_Output3 = Q0.3 • State_Machine_Step = VB1500	**LDB= State_Machine_ Step, 4** **R CPU_Output3, 2**

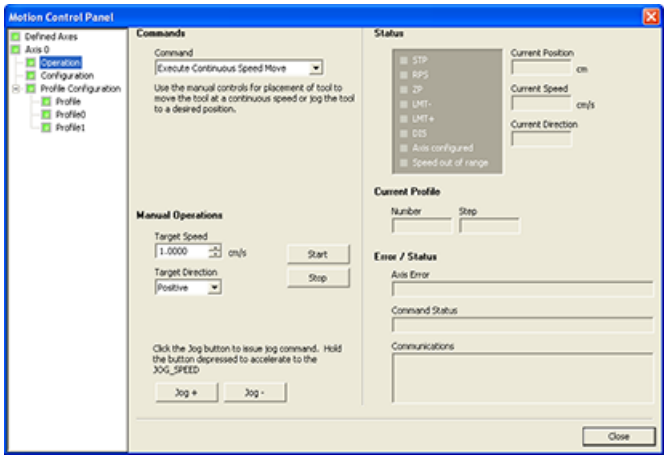
LAD/FBD	Description	STL
Network 13 	If the State Machine is in Step 4, flash output 5 to indicate an error. Symbol and Address: [1] • Clock_1s = SM0.5 (Clock pulse that is ON for 0.5 s and OFF for 0.5 s for a duty cycle time of 1 s.) • CPU_Output5 = Q0.5 • State_Machine_Step = VB1500	**LDB= State_Machine_ Step, 4** **A Clock_1s** **= CPU_Output5**
Network 14  	If the State Machine is in Step 4, you must acknowledge the error by toggling Input I0.2. This action resets the state to Step 0. Symbol and Address: [1] • CPU_Input2 = I0.2 • State_Machine_Step = VB1500	**LDB= State_Machine_ Step, 4** **A CPU_Input2** **MOVB 0, State_Machine_ Step** **R M0.0, 9**

[1] The program addresses shown are example addresses. Your program addresses could vary.

16.4.6 Monitoring the Axis of Motion

16.4.6.1 Displaying and controlling the operation of the Axis of Motion

In the Operation node, you can interact with the operations of the Axis of Motion. The control panel displays the current speed, the current position, and the current direction of the Axis of Motion. You can also see the status of the input and output LEDs (excluding the Pulse LEDs).



The control panel allows you to interact with the Axis of Motion by changing the speed and direction, by stopping and starting the motion, and by jogging the tool (if the CPU is stopped).

Note

You cannot execute a motion command while the CPU is running. The CPU must be in STOP mode in order to change the speed and direction, stop and start the motion, and use the jog tool.

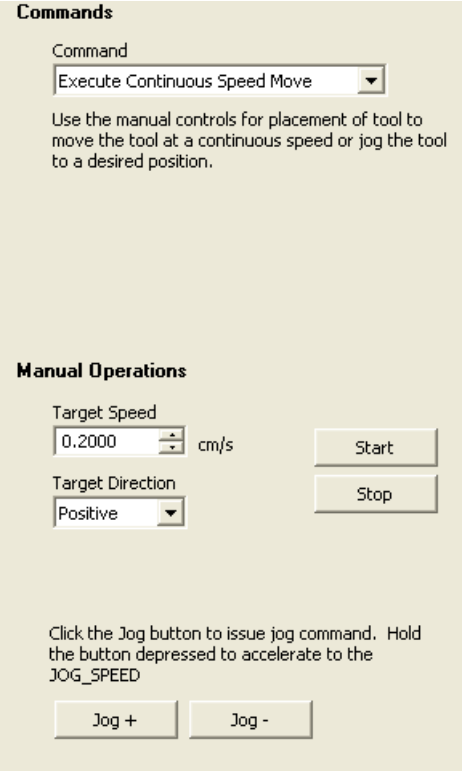
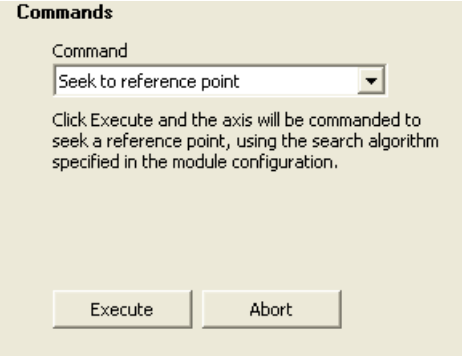
Note

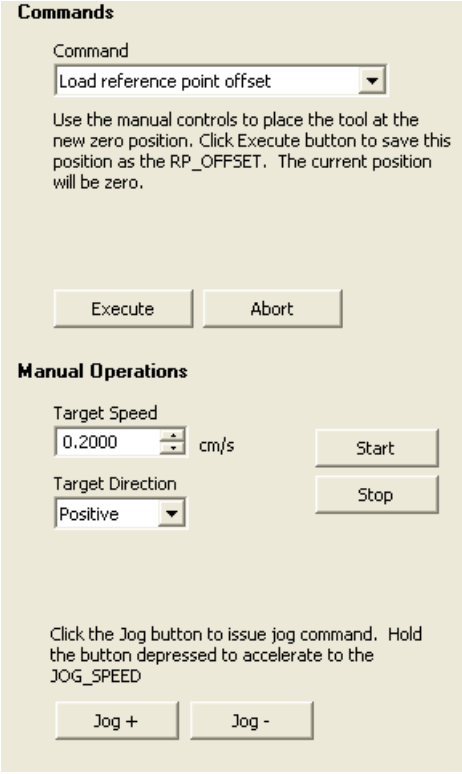
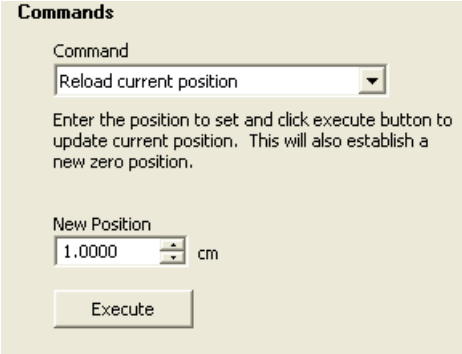
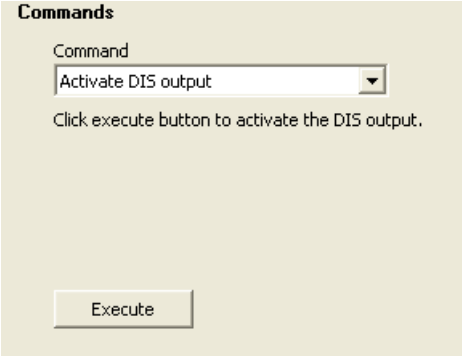
Exiting the Motion control panel or a loss of communications while a motion command is active causes the axis to stop its motion after a 5 second timeout.

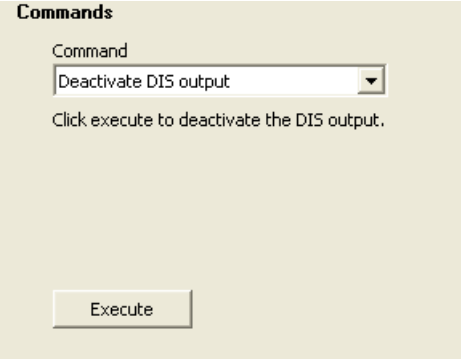
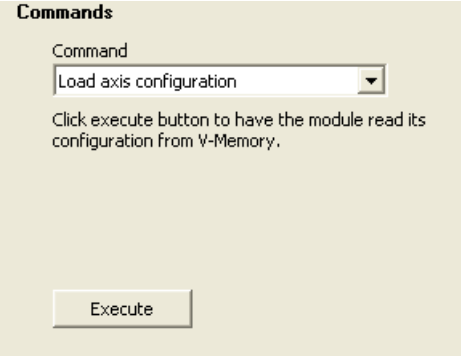
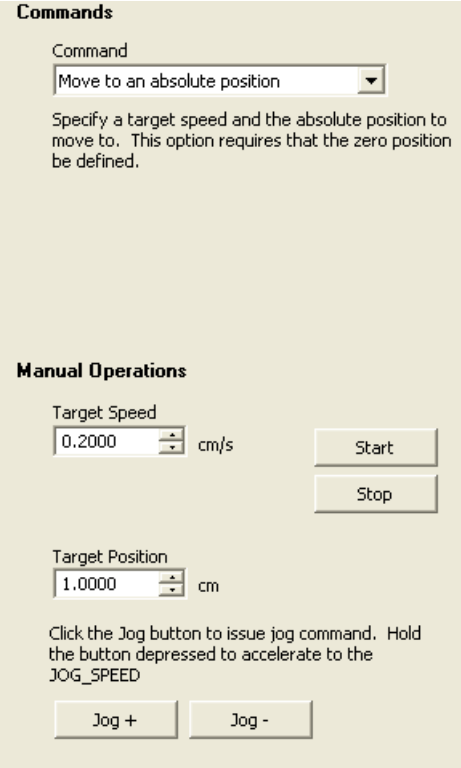
Motion commands

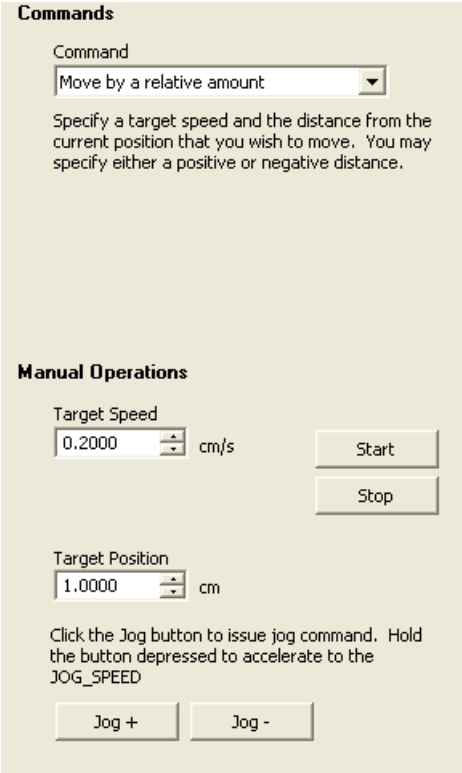
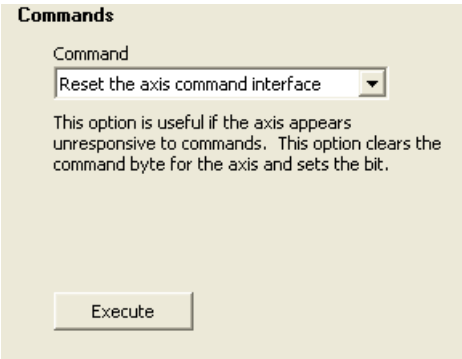
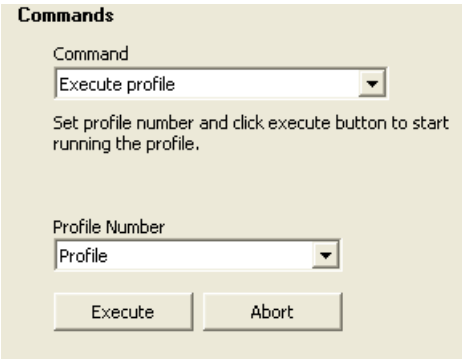
You can also generate the following motion commands:

Table 16-35 Motion control panel commands

Command	Description
	Execute continuous speed move: This command allows you to use the manual controls for positioning the tool. Enter "Target Speed" and "Direction", and click "Start" to execute continuous move. Motion will continue until "Stop" is clicked (or error condition).
	Seek to a reference point: This command finds the reference point by using the configured search mode. Click "Execute", and the axis will command a "Seek to Reference Point", using the search algorithm specified in the axis configuration. Click "Abort" to stop a seek process before the reference point has been found.

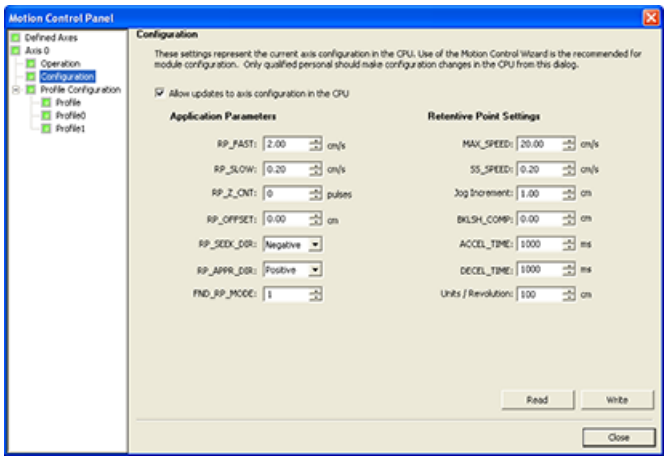
Command	Description
	Load reference point offset: After you use the manual controls to jog the tool to the new position, you then load the "Reference Point Offset". Use the manual controls to place the tool at the new position. Click "Execute" to save this position as the "RP_OFFSET". The current position will be set to zero.
	Reload current position: This command updates the current position value and establishes a new zero position. Enter the position to set and click "Execute" to update the current position. This will also establish a new zero position.
	Activate the DIS output: This command turns the DIS output of the Axis of Motion on. Click "Execute" to activate the DIS output.

Command	Description
	De-activate the DIS output: This command turns the DIS output of the Axis of Motion off. Click "Execute" to de-activate the DIS output.
	Load axis configuration: This command loads a new configuration by commanding the Axis of Motion to read the configuration block from the V memory of the CPU. Click "Execute" to have the axis read its configuration from V memory.
	Move to an absolute position: This command allows you to move to a specified position at a target speed. Before using this command, you must have already established the zero position. Assign a "Target Speed" and the "Absolute Position" to move to. This option requires that the zero position be defined.

Command	Description
	Move by a relative amount: This command allows you to move a specified distance from the current position at a target speed. You can enter a positive or negative distance. Assign a "Target Speed" and a "Target Position" to move to.
	Reset the axis command interface: This command clears the axis command interface for the Axis of Motion and sets the "Done" bit. Use this command if the Axis of Motion appears to not be responding to commands.
	Execute profile: This command allows you to select a profile to be executed. The control panel displays the status of the profile which is being executed by the Axis of Motion. Select the profile that is to be executed, then click "Execute", and the axis will command the profile. Note: This command is only available if a profile has been defined in the Motion wizard.

16.4.6.2 Displaying and modifying the configuration of the Axis of Motion

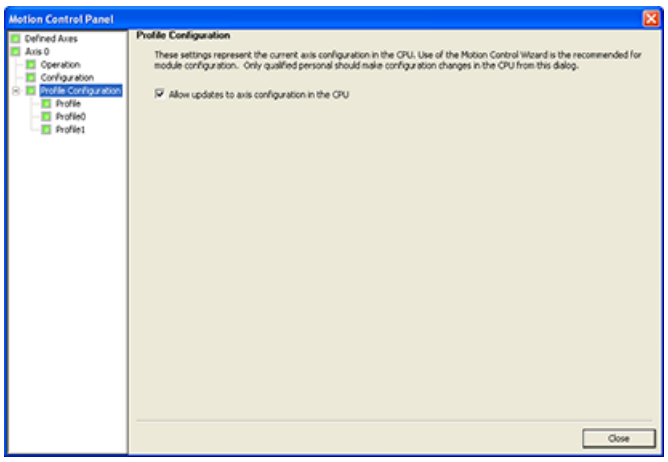
In the Configuration node, you can view and modify the configuration settings for the Axis of Motion that is stored in the data block of the CPU.



After you modify the configuration settings, you simply click the write button to send the data values to the CPU. These data values are not saved in your STEP 7-Micro/WIN SMART project. You must manually make changes to your project that reflects the final values of these fields.

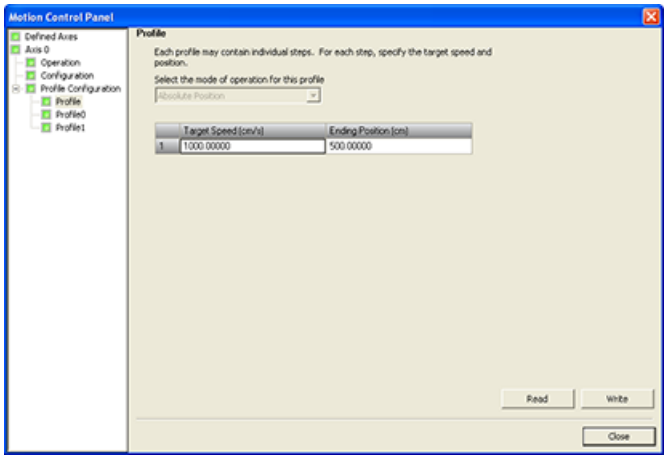
16.4.6.3 Displaying the profile configuration for the Axis of Motion

In the Profile Configuration node, you can view the configuration of each profile for the Axis of Motion.



Click each profile to view its mode of operation and data values.

Some data values of the profile can be modified in this dialog. After you modify the configuration settings, you simply click the write button to send the data values to the CPU. These data values are not saved in your STEP 7-Micro/WIN SMART project. You must manually make changes to your project that reflects the final values of these fields.



16.4.6.4 Error codes for the Axis of Motion (WORD at SMW620, SMW670, SMW720 or SMW770)

Table 16-36 Axis of Motion error codes

Error code	Description
0	No error
1	Reserved
2	Configuration block not present
3	Configuration block pointer error
4	Size of configuration block exceeds available V memory
5	Illegal configuration block format
6	Too many profiles specified
7	Illegal STP_RSP specification
8	Illegal LIM- specification
9	Illegal LIM+ specification
10	Illegal FILTER_TIME specification
11	Illegal MEAS_SYS specification
12	Illegal RP_CFG specification
13	Illegal PLS/REV value
14	Illegal UNITS/REV value
15	Illegal RP_ZP_CNT value
16	Illegal JOG_INCREMENT value
17	Illegal MAX_SPEED value
18	Illegal SS_SPD value
19	Illegal RP_FAST value
20	Illegal RP_SLOW value
21	Illegal JOG_SPEED value

Error code	Description
22	Illegal ACCEL_TIME value
23	Illegal DECEL_TIME value
24	Illegal JERK_TIME value
25	Illegal BKLSH_COMP value
26	AXIS not available
27	Invalid LMT+ location
28	Invalid LMT- location
29	Invalid STP location
30	Invalid RPS location
31	Invalid ZP location
32	Illegal output phase
33	No RPS input defined (If Homing is defined, then RPS must be defined also.)
34	Invalid TRIG location
35	Reserved
36	No ZP input defined (If Homing mode 3 or 4 is defined, then ZP must be defined also.)
37	Phase A (P0) output conflict
38	Phase B (P1) output conflict
39	DIS output conflict
40	Reserved
41	Invalid SDB0 record size
42	Illegal SDB0 format
43 to 127	Reserved

To verify that the Axis of Motion is wired correctly, to adjust the configuration data, and to test each movement profile, use the Motion control panel.

If additional changes need to be made in the Axis of Motion, go to the Motion wizard.

16.4.6.5 Error codes for the Axis of Motion instruction (seven LS bits of SMB634, SMB684, SMB734 or SMB784))

In the SM table for each axis there is a byte reserved to display the result of the motion instruction (Offset 34). This byte indicates when an instruction is complete and if there was an error in the instruction.

Table 16-37 Motion instruction error codes

Error code	Description
0	No error
1	Aborted by user
2	Configuration error (This error occurs if there is an error in the SDB0 configuration.)
3	Illegal command

Error code	Description
4	Aborted due to no valid configuration (This error occurs if there is an error in the configuration table.)
5	Reserved
6	Aborted due to no defined reference point
7	Aborted due to STP input active
8	Aborted due to LMT- input active
9	Aborted due to LMT+ input active
10	Aborted due to problem executing motion
11	No profile block configured for specified profile
12	Illegal operation mode
13	Operation mode not supported for this command
14	Illegal number of steps in profile block
15	Illegal direction change
16	Illegal distance
17	RPS/TRIG trigger occurred before target speed reached
18	Insufficient RPS active region width
19	Speed out-of-range
20	Insufficient distance to perform desired speed change
21	Illegal position
22	Zero position unknown
23	No DIS output is defined
24	Reserved
25	Aborted due to CPU going to stop
26	Aborted due to expiration of Motion control panel heartbeat
27	Illegal operation mode in AXIS_GOTO override function
28	Single axis movement is blocked as Axes group is running
29 to 127	Reserved
128	Axis of Motion cannot process this instruction: either the Axis of Motion is busy with another instruction, or there was no Start pulse on this instruction.
129	Reserved
130	Axis of Motion is not enabled
131	Reserved
132	Reserved
133	Illegal profile specified. The AXISx_RUN and AXISx_CACHE instructions profile number range must be between 0 - 31.
134	Illegal mod specified in AXISx_GOTO instruction

To verify that the Axis of Motion is wired correctly, to adjust the configuration data, and to test each movement profile, use the Motion control panel.

If additional changes need to be made in the Axis of Motion, go to the Motion wizard.

16.4.7 Advanced topics

16.4.7.1 Understanding the configuration/profile table for the Axis of Motion

Overview

The Motion wizard has been developed to make motion applications easy by automatically generating the configuration and profile information based upon the answers you give about your motion control system. Configuration/profile table information is provided for advanced users who want to create their own motion control routines.

The configuration/profile table is located in the V memory area of the S7-200 SMART CPU. As shown in the table below, the configuration settings are stored in the following types of information:

- **Configuration block:** Contains information used to setup the Axis of Motion in preparation for executing position commands
- **Interactive block:** Supports direct setup of position parameters by the user program
- **Profile block:** Describes a pre-defined move operation to be performed by the Axis of Motion. You can configure up to 32 profile blocks.

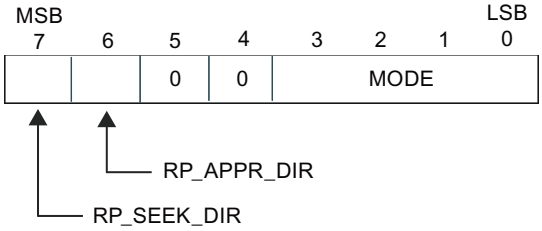
Note

The profile block of the configuration/profile table can contain up to 32 motion profiles. To create more than 32 move profiles, you can exchange configuration/profile tables by changing the value stored in the configuration/profile table pointer.

Table 16-38 Configuration/Profile table: Configuration block

Configuration/Profile table			
Byte off-set	**Name**	**Function description**	**Type**
Configuration block			
0	MOD_ID	Axis of Motion identification field	--
5	CB_LEN	Length of the configuration block in bytes (1 byte)	--
6	IB_LEN	Length of the interactive block in bytes (1 byte)	--
7	PF_LEN	Length of a single profile in bytes (1 byte)	--
8	STP_LEN	Length of a single step in bytes (1 byte)	--
9	STEPS	Number of steps allowed per profile (1 byte)	--
10	PROFILES	Number of profiles from 0 to 32 (1 byte)	--
11	--	Reserved: Set to 0	--
13	--	Reserved: Set to 0	--
14	STOP_RSP	Specifies the drive's response to the STP input (1 byte): • 0: No action. Ignore input condition. • 1: Decelerate to a stop and indicate STP input active. • 2: Terminate pulses and indicate STP input. • 3 to 255: Reserved (error if specified)	--

Configuration/Profile table			
Byte off-set	**Name**	**Function description**	**Type**
Configuration block			
15	LMT-_RSP	Specifies the drive's response to the negative limit input (1 byte): • 0: No action. Ignore input condition. • 1: Decelerate to a stop and indicate limit reached. • 2: Terminate pulses and indicate limit reached. • 3 to 255: Reserved (error if specified)	--
16	LMT+_RSP	Specifies the drive's response to the positive limit input (1 byte): • 0: No action. Ignore input condition. • 1: Decelerate to a stop and indicate limit reached. • 2: Terminate pulses and indicate limit reached. • 3 to 255: Reserved (error if specified)	--
17	--	Reserved: Set to 0	--
18	MEAS_SYS	Specifies the measurement system used to describe moves (1 byte): • 0: Pulses (speed measured in pulses/sec and position values measured in pulses; values are double integer) • 1: Engineering units (speed measured in units/sec and position values measured in units; values are single precision real) • 2 to 255: Reserved (error if specified)	--
19	--	Reserved: Set to 0	--
20	PLS/REV	Specifies the number of pulses per revolution of the motor, (only applicable when MEAS_SYS is set to 1) - (4 bytes)	DInt
24	UNITS/REV	Specifies the engineering units per revolution of the motor, (only applicable when MEAS_SYS is set to 1) - (4 bytes)	Real
28	UNITS	Reserved for STEP 7-Micro/WIN SMART to store a custom units string (4 bytes)	--

Configuration/Profile table			
Byte offset	**Name**	**Function description**	**Type**
		Configuration block	
32	RP_CFG	Specifies the reference point search configuration (1 byte):  RP_SEEK_DIR: This bit specifies the starting direction for a reference point search (0 - positive direction, 1 - negative direction). RP_APPR_DIR: This bit specifies the approach direction for terminating the reference point search (0 - positive direction, 1 - negative direction). MODE: Specifies the reference point search method '0000': Reference point search disabled. '0001': The reference point is where the RPS input goes active. '0010': The reference point is centered within the active region of the RPS input. '0011': The reference point is outside the active region of the RPS input. '0100': The reference point is within the active region of the RPS input. '0101' to '1111': Reserved (error if selected)	--
33	--	Reserved: Set to 0	--
34	RP_Z_CNT	Number of pulses of the ZP input used to define the reference point (4 bytes)	DInt
38	RP_FAST	Fast speed for the RP seek operation: MAX_SPD or less (4 bytes)	DInt/Real
42	RP_SLOW	Slow speed for the RP seek operation: Maximum speed from which the motor can instantly go to a stop or less (4 bytes)	DInt/Real
46	SS_SPEED	Start/Stop Speed. (4 bytes): The starting speed is the maximum speed to which the motor can instantly go from a stop and the maximum speed from which the motor can instantly go to a stop. Operation below this speed is allowed, but the acceleration and deceleration times do not apply.	DInt/Real
50	MAX_SPEED	Maximum operating speed of the motor (4 bytes)	DInt/Real
54	JOG_SPEED	Jog speed (4 bytes): MAX_SPEED or less (4 bytes)	DInt/Real

Configuration/Profile table			
Byte offset	**Name**	**Function description**	**Type**
Configuration block			
58	JOG_INCREMENT	Jog increment value: The distance (or number of pulses) to move in response to a single jog pulse (4 bytes).	DInt/Real
62	ACCEL_TIME	Time required to accelerate from minimum to maximum speed in msec (4 bytes)	DInt
66	DECEL_TIME	Time required to decelerate from maximum to minimum speed in msec (4 bytes)	DInt
70	BKLSH_COMP	Backlash compensation: The distance used to compensate for the system backlash on a direction change (4 bytes).	DInt/Real
74	JERK_TIME	Time during which jerk compensation is applied to the beginning and ending portions of an acceleration/deceleration curve (S-curve). Specifying a value of 0 disables jerk compensation. The jerk time is given in milliseconds (Range: 0 ms to 32000 ms. (4 bytes)	DInt

Table 16-39 Configuration/Profile table: Interactive block

Configuration/Profile table			
Byte offset	**Name**	**Function description**	**Type**
Interactive block			
78	MOVE_CMD	Selects the mode of operation (1 byte): • 0: Absolute position • 1: Relative position • 2: Single-speed, continuous positive rotation • 3: Single-speed, continuous negative rotation • 4: Manual speed control, positive rotation • 5: Manual speed control, negative rotation • 6: Single-speed, continuous positive rotation with triggered stop (Activation of RPS triggers the stop; TARGET_POS contains distance to travel after signal) • 7: Single-speed, continuous negative rotation with triggered stop (Activation of RPS triggers the stop; TARGET_POS contains distance to travel after signal) • 8 to 255: Reserved (error if specified)	--
79	--	Reserved: Set to 0	--
80	TGT_POS	Target position to go to in this move (4 bytes)	DInt/Real
84	TGT_SPEED	Target speed for this move (4 bytes)	DInt/Real
88	RP_OFFSET	Absolute position of the reference point (4 bytes)	DInt/Real

Table 16-40 Configuration/Profile table: Profile block 0

Configuration/Profile table			
Byte offset	**Name**	**Function description**	**Type**
		Profile block 0	
92(+0)	STEPS	Number of steps in this move sequence (1 byte)	--
93(+1)	MODE	Selects the mode of operation for this profile block (1 byte): • 0: Absolute position • 1: Relative position • 2: Single-speed, continuous positive rotation • 3: Single-speed, continuous negative rotation • 4: Reserved (error if specified) • 5: Reserved (error if specified) • 6: Single-speed, continuous positive rotation with triggered stop (RPS input signals stop) • 7: Single-speed, continuous negative rotation with triggered stop (RPS input signals stop) • 8: Two-speed, continuous positive rotation RPS selects speed) • 9: Two-speed, continuous negative rotation (RPS selects speed) • 10: Two-speed, continuous positive rotation with triggered stop (RPS selects speed, TRIG input signals stop) • 11: Two-speed, continuous negative rotation with triggered stop (RPS selects speed, TRIG input signals stop) • 12 to 255: Reserved (error if specified)	--
94(+2)	Step 0: POS	Position to go to in move step 0 (4 bytes)	DInt/Real
98(+6)	Step 0: SPEED	Target speed for move step 0 (4 bytes)	DInt/Real
102(+10)	Step 1: POS	Position to go to in move step 1 (4 bytes)	DInt/Real
106(+14)	Step 1: SPEED	Target speed for move step 1 (4 bytes)	DInt/Real
110(+18)	Step 2: POS	Position to go to in move step 2 (4 bytes)	DInt/Real
114(+22)	Step 2: SPEED	Target speed for move step 2 (4 bytes)	DInt/Real
118(+26)	Step 3: POS	Position to go to in move step 3 (4 bytes)	DInt/Real
122(+30)	Step 3: SPEED	Target speed for move step 3 (4 bytes)	DInt/Real
...	Step ...: POS	Position to go to in move step ... (4 bytes)	DInt/Real
...	Step ...: SPEED	Target speed for move step ... (4 bytes)	DInt/Real
214(+122)	Step 15: POS	Position to go to in move step 15 (4 bytes)	DInt/Real
218(+126)	Step 15: SPEED	Target speed for move step 15 (4 bytes)	DInt/Real

Note

You can have between 1 and 16 steps in the Configuration/Profile table, Profile block 0.

Table 16-41 Configuration/Profile table: Profile block 1

Configuration/Profile table			
Byte offset	**Name**	**Function description**	**Type**
Profile block 1			
X[1]	STEPS	Number of steps in this move sequence (1 byte) **Note**: There can be up to 16 steps.	--
(X + 1)	MODE	Selects the mode of operation for this profile block (1 byte)	--
(X + 2)	Step 0: POS	Position to go to in move step 0 (4 bytes)	DInt/Real
(X + 4)	Step 0: SPEED	The target speed for move step 0 (4 bytes)	DInt/Real
...	...	...	...

[1] The offset of Profile block 1 and subsequent blocks is variable and dependent upon the number of steps configured in the largest profile. The offset is determined by the following formula:
Offset of Profile block x = CB_LEN + IB_LEN + (x * PF_LEN)

Table 16-42 Profile detail for Mode 0 (Absolute position)

Byte offset from start of profile	Step number	Name	Field size	Value
+0		STEPS	byte	n = Number of steps configured in this profile
+1		MODE	byte	0 = Absolute position
+2	0	POS	dint/fp	Destination position in step 0
+6		SPEED	dint/fp	Target speed for step 0
● ● ●				
(4 * n) + 2	n	POS	dint/fp	Destination position in step n
($ * N) + 6		SPEED	dint/fp	Target speed for step n

Table 16-43 Profile detail for Mode 1 (Relative position)

Byte offset from start of profile	Step number	Name	Field size	Value
+0		STEPS	byte	n = Number of steps configured in this profile
+1		MODE	byte	0 = Relative position
+2	0	POS	dint/fp	Distance to travel in step 0
+6		SPEED	dint/fp	Target speed for step 0
● ● ●				

Byte offset from start of profile	Step number	Name	Field size	Value
(4 * n) + 2	n	POS	dint/fp	Distance to travel in step n
($ * N) + 6		SPEED	dint/fp	Target speed for step n

Table 16-44 Profile detail for Mode 2 (Single-speed, continuous positive rotation) and Mode 3 (Single-speed, continuous negative rotation)

Byte offset from start of profile	Step number	Name	Field size	Value
+0		STEPS	byte	1
+1		MODE	byte	2= Single-speed, continuous positive rotation or 3 = Single-speed, continuous negative rotation
+2	0	POS	dint/fp	n.a. (must be set to 0)
+6		SPEED	dint/fp	Target speed

Table 16-45 Profile detail for Mode 6 (Single-speed, continuous positive rotation with triggered stop) and Mode 7 (Single-speed, continuous negative rotation with triggered stop)

Byte offset from start of profile	Step number	Name	Field size	Value
+0		STEPS	byte	1
+1		MODE	byte	6 = Single-speed, continuous positive rotation with triggered stop or 7 = Single-speed, continuous negative rotation with triggered stop
+2	0	POS	dint/fp	Distance to travel after activation of RPS signal (value must be positive)
+6		SPEED	dint/fp	Target speed

Table 16-46 Profile detail for Mode 8 (Two-speed, continuous positive rotation) and Mode 9 (Two-speed, continuous negative rotation)

Byte offset from start of profile	Step number	Name	Field size	Value
+0		STEPS	byte	2
+1		MODE	byte	8 = Two-speed, continuous positive rotation or 9 = Two-speed, continuous negative rotation
+2	0	POS	dint/fp	n.a. (must be set to 0)

Byte offset from start of profile	Step number	Name	Field size	Value
+6		SPEED	dint/fp	Target speed if RPS signal is inactive
+10	1	POS	dint/fp	n.a. (must be set to 0)
+14		SPEED	dint/fp	Target speed if RPS signal is active

Table 16-47 Profile detail for Mode 10 (Two-speed, continuous positive rotation with triggered stop) and Mode 11 (Two-speed, continuous negative rotation with triggered stop)

Byte offset from start of profile	Step number	Name	Field size	Value
+0		STEPS	byte	2
+1		MODE	byte	10 = Two-speed, continuous positive rotation with triggered stop or 11 = Two-speed, continuous negative rotation with triggered stop
+2	0	POS	dint/fp	Distance to travel after activation of TRIG signal (value must be positive)
+6		SPEED	dint/fp	Target speed if RPS signal is inactive
+10	1	POS	dint/fp	n.a. (must be set to 0)
+14		SPEED	dint/fp	Target speed if RPS signal is active

16.4.7.2 Special memory (SM) locations for the Axis of Motion

The CPU allocates 50 bytes of special memory (SM) to each Axis of Motion. (See the following table.) When the Axis of Motion detects an error condition or a change in status of the data, the Axis of Motion updates these SM locations. The first Axis of Motion updates SMB600 through SMB649 as required to report the error condition, the second Axis of Motion updates SMB650 through SMB699, and so on.

Table 16-48 Special memory bytes SMB600 to SMB799

SM bytes for Axes of Motion:			
Axis of Motion 0	**Axis of Motion 1**	**Axis of Motion 2**	**Axis of Motion 3**
SMB600 to SMB649	SMB650 to SMB699	SMB700 to SMB749	SMB750 to SMB799

The following table shows the structure of the SM data area allocated for an Axis of Motion. The definition uses Axis 0 as an example.

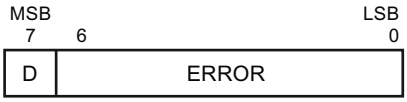
Table 16-49 Special memory area definition for the Axis of Motion 0

SM address	Description
SMB600 to SMB615	Axis name (16 ASCII characters). SMB600 is the first character: "Axis 0"
SMB616 to SMB619	Reserved
SMW620	Axis 0: Error code (See "Axis of Motion error codes" (Page 938) list.)

<table>
<tr><th>SM address</th><th>Description</th></tr>
<tr><td>SMB622</td><td>Axis 0: Input/output status: Reflects the status of the inputs and outputs
<table>
<tr><td>MSB
7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>LSB
0</td></tr>
<tr><td>DIS</td><td>0</td><td>TRIG</td><td>STP</td><td>LMT-</td><td>LMT+</td><td>RPS</td><td>ZP</td></tr>
</table>

DIS (Disable outputs):
– 0 = No current flow
– 1 = Current flow
TRIG (Stop input):
– 0 = No current flow
– 1 = Current flow
STP (Stop input):
– 0 = No current flow
– 1 = Current flow
LMT- (Negative travel limit input):
– 0 = No current flow
– 1 = Current flow
LMT+ (Positive travel limit input):
– 0 = No current flow
– 1 = Current flow
RPS (Reference point switch input):
– 0 = No current flow
– 1 = Current flow
ZP (Zero pulse input):
– 0 = No current flow
– 1 = Current flow
</td></tr>
<tr><td>SMB623</td><td>Axis 0 Instantaneous status: Reflects the status of the configuration and rotation direction status
<table>
<tr><td>MSB
7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>LSB
0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>OR</td><td>R</td><td>CFG</td></tr>
</table>

OR (Target speed out of range):
– 0 = In range
– 1 = Out of range
R (Direction of rotation):
– 0 = Positive rotation
– 1 = Negative rotation
CFG (Module configured):
– 0 = Not configured
– 1 = Configured
</td></tr>
</table>

SM address	Description
SMB624	Axis 0: CUR_PF is a byte that indicates the profile currently being executed.
SMB625	Axis 0: CUR_STP is a byte that indicates the step currently being executed in the profile.
SMD626	Axis 0: CUR_POS is a double-word value that indicates the current position of the Axis of Motion.
SMD630	Axis 0: CUR_SPD is a double-word value that indicates the current speed of the Axis of Motion.
SMB634	Axis 0: Result of the instruction. Error conditions above 127 are generated by the instruction subroutines created by the Motion wizard.  • D (Done bit): – 0= Operation in progress – 1= Operation complete (set by the Axis of Motion during initialization) • ERROR: (See "Motion instruction error codes" (Page 939) list.)
SMB635 to SMB645	Reserved
SMD646	Axis 0: Pointer to the V memory location of the configuration/profile table. A pointer value to an area other than V memory is not valid. The Axis of Motion monitors this location until it receives a non-zero pointer value.

16.4.8 Understanding the RP Seek modes of the Axis of Motion

The following figures provide diagrams of the different options for each RP Seek mode:

- RP Seek: Mode 1 shows two of the options for RP Seek mode 1. This mode locates the RP where the RPS input goes active on the approach from the work zone side.
- RP Seek: Mode 2 shows two of the options for RP Seek mode 2. This mode locates the RP in the center within the active region of the RPS input.
- RP Seek: Mode 3 shows two of the options for RP Seek mode 3. This mode locates the RP a specified number of zero pulses (ZP) outside the active region of the RPS input.
- RP Seek: Mode 4 shows two of the options for RP Seek mode 4. This mode locates the RP a specified number of zero pulses (ZP) within the active region of the RPS input.

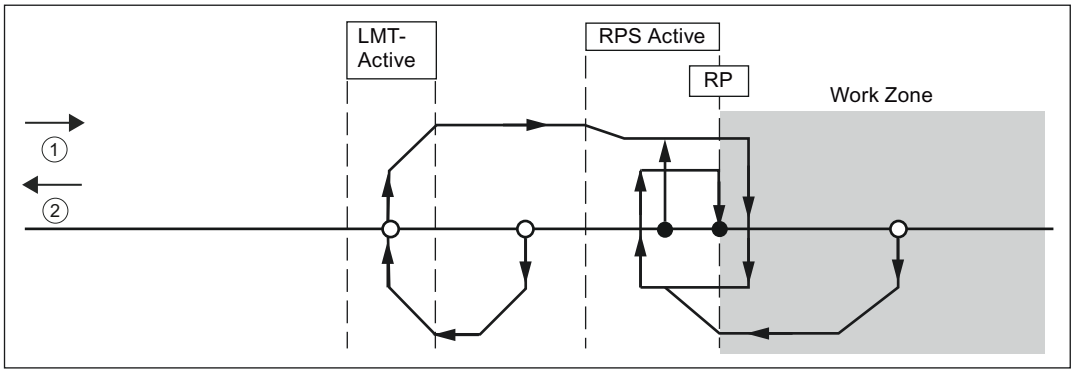
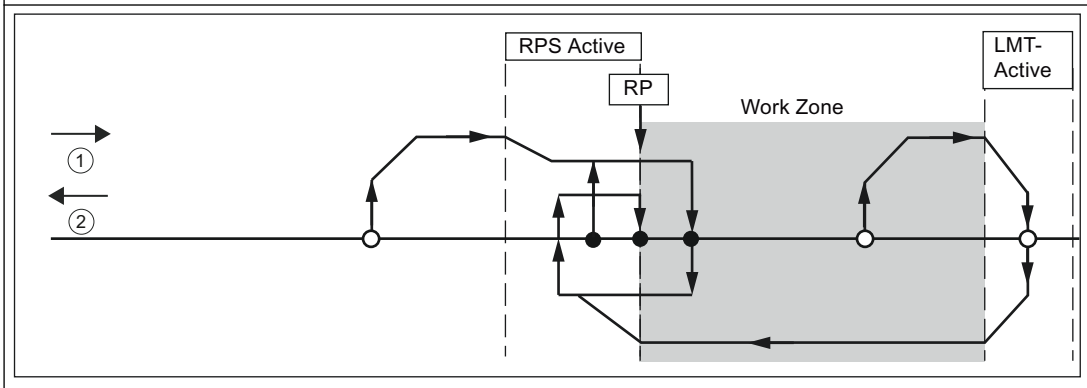
For each mode, there are four combinations of RP Seek direction and RP Approach direction. (Only two of the combinations are shown.) These combinations determine the pattern for the RP Seek operation. For each of the combinations, there are also four different starting points:

The work zones for each diagram have been located so that moving from the reference point to the work zone requires movement in the same direction as the RP Approach Direction.

By selecting the location of the work zone in this way, all the backlash of the mechanical gearing system is removed for the first move to the work zone after a reference point seek.

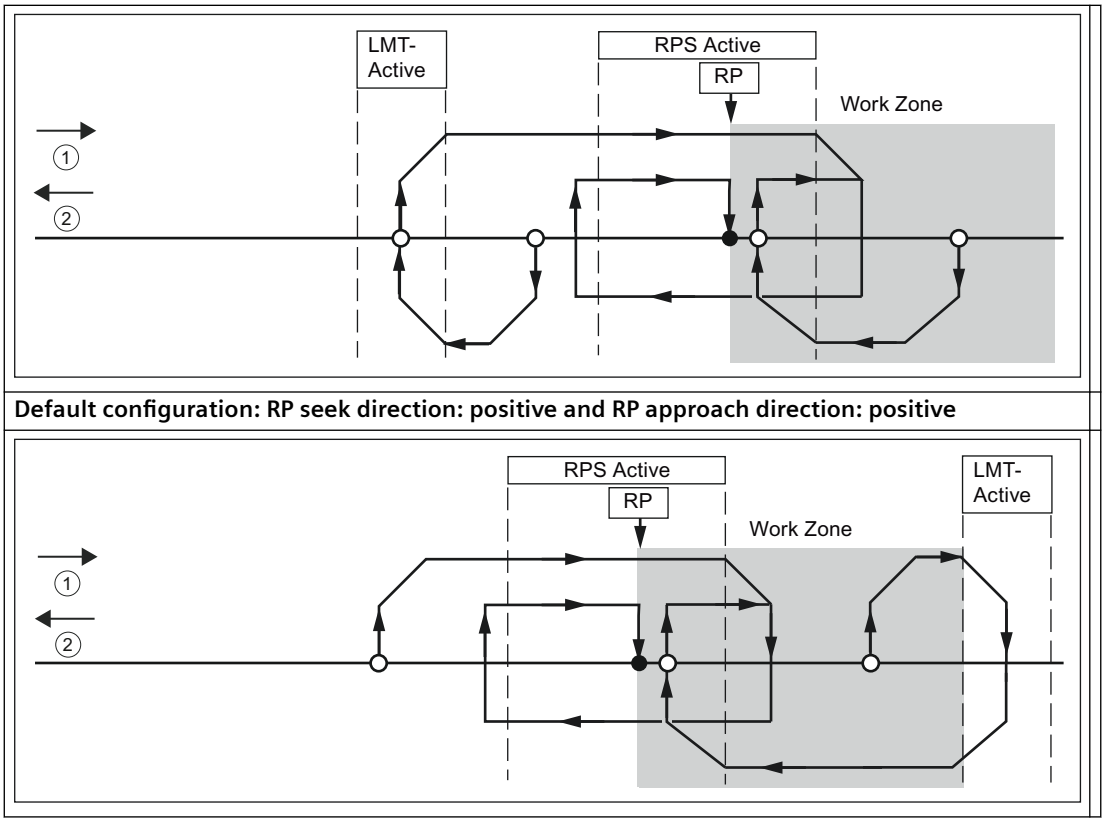
Note

The RPS input must be enabled to use the RP Seek functionality. The ZP input must also be enabled if RP Seek modes 3 or 4 are to be used, unless you configure the number of ZP pulses to be received after entering the RPS active region to a value of "0".

RP seek mode 1
Default configuration: RP seek direction: negative and RP approach direction: positive

Default configuration: RP seek direction: positive and RP approach direction: positive


①: Positive motion
②: Negative motion

RP seek mode 2
Default configuration: RP seek direction: negative and RP approach direction: positive

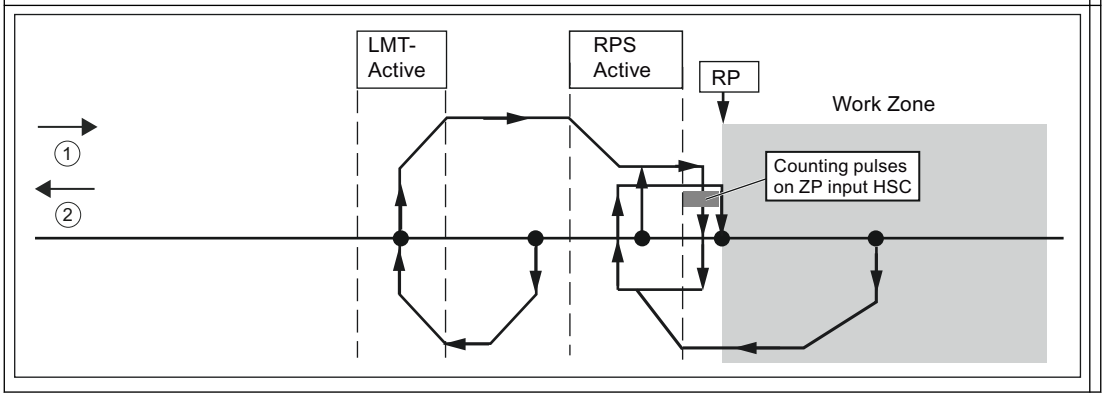


Default configuration: RP seek direction: positive and RP approach direction: positive

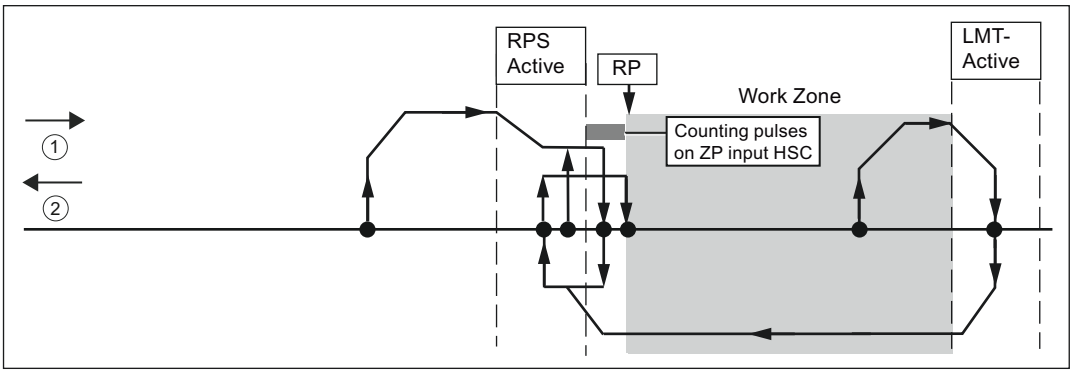
①: Positive motion
②: Negative motion

RP seek mode 3

Default configuration: RP seek direction: negative and RP approach direction: positive



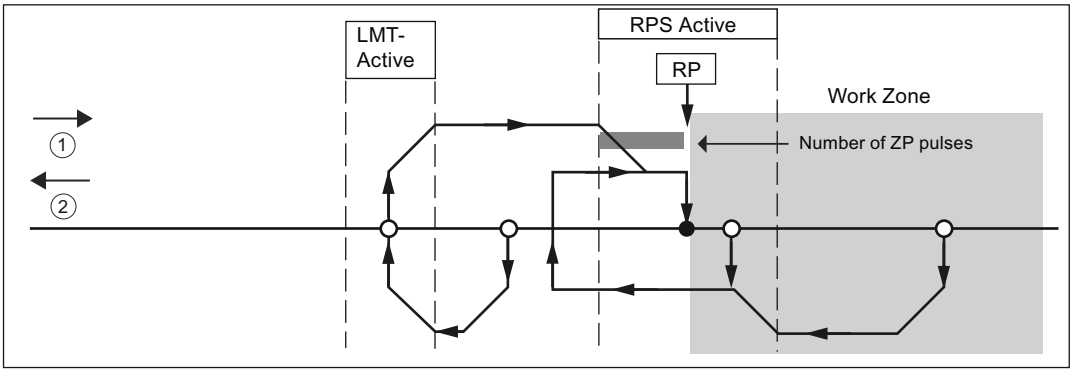
Default configuration: RP seek direction: positive and RP approach direction: positive



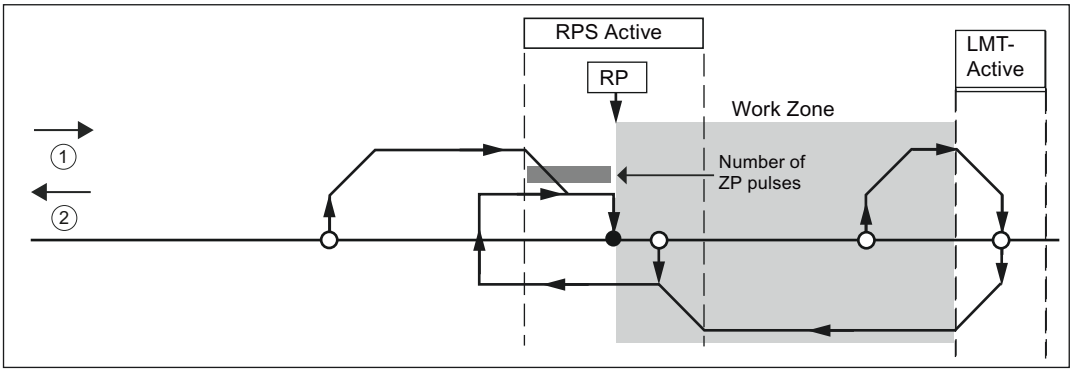
①: Positive motion
②: Negative motion

RP seek mode 4

Default configuration: RP seek direction: negative and RP approach direction: positive



Default configuration: RP seek direction: positive and RP approach direction: positive



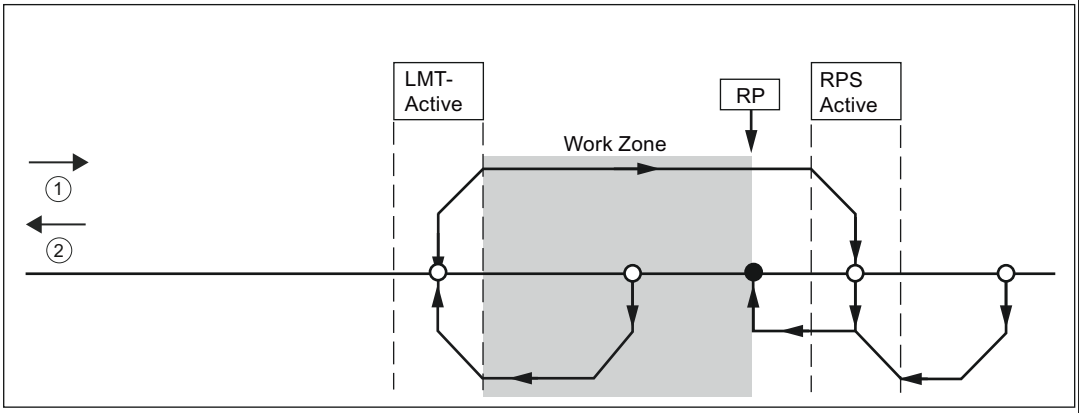
①: Positive motion
②: Negative motion

16.4.8.1 Selecting the work zone location to eliminate backlash

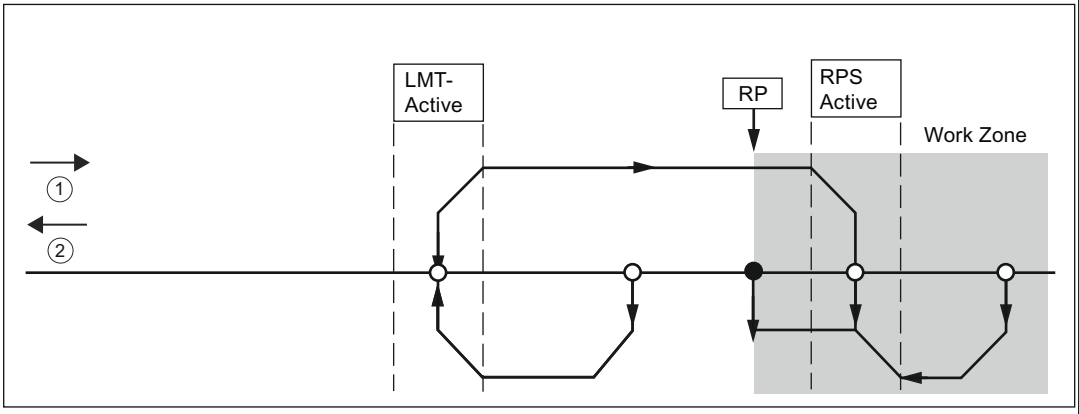
The following figure shows the work zone in relationship to the reference point (RP), the RPS Active zone, and the limit switches (LMT+ and LMT-) for an approach direction that eliminates the backlash. The second part of the illustration places the work zone so that the backlash is not eliminated. The following figure shows RP seek mode 3. A similar placement of the work zone is possible, although not recommended, for each of the search sequences for each of the other RP seek modes.

Selecting the work zone location to eliminate backlash

Backlash is eliminated: RP seek direction: negative and RP approach direction: negative



Backlash is not eliminated: RP seek direction: negative and RP approach direction: negative



①: Positive motion
② Negative motion

16.5 Using Axes group of Motion

16.5.1 Programming an Axes group of Motion

STEP 7-Micro/WIN SMART V3 provides easy-to-use tools for configuring and programming the Axes group of Motion. Simply follow these steps:

1. Configure the Axes group of Motion: STEP 7-Micro/WIN SMART V3 provides a Motion wizard for the configuration of Axes group and move path. See Configuring the Axes group of Motion (Page 955) for information about configuring the Axes group of Motion.
2. Create the program to be executed by the CPU: The Motion wizard automatically generates the motion instructions that you can call in your program to implement axes group functions. See Subroutines created by the Motion wizard for the Axes group of Motion (Page 962) for information about the motion instructions. Call the following instructions into your program:
   - To enable the Axes group of Motion, insert an AXISx_CTRL instruction. Use SM0.0 (Always On) to ensure that this instruction is executed every scan.
   - To command the motor to do linear interpolation movement to a specific location, use the GRP0_2D_MOVELINEAR or GRP0_3D_MOVELINEAR instruction.
   - To move the motor according to a specified move path, use the GRP0_MOVEPATH instruction.
3. Compile your program and download the system block, data block, and program block to the CPU.

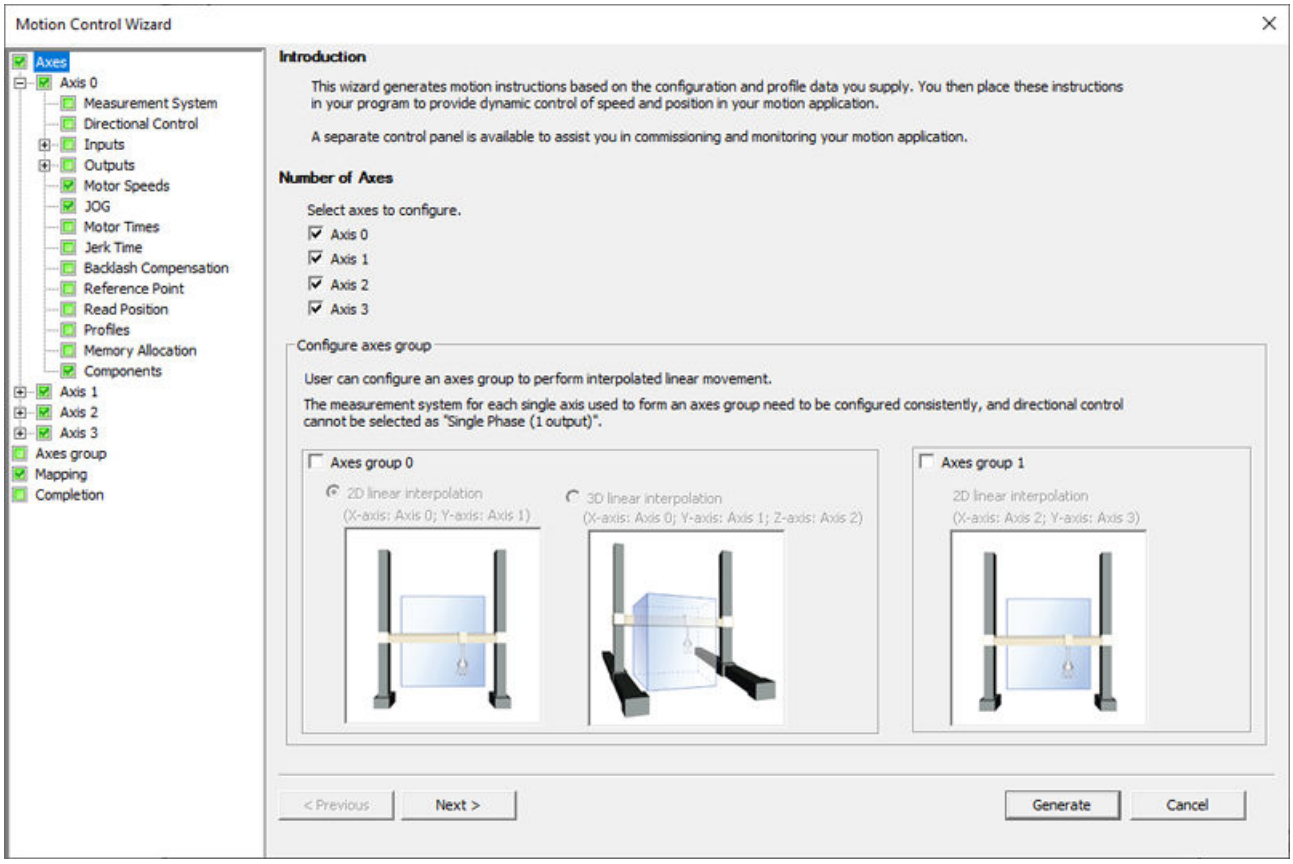
Note

Make sure to match the measurement system configuration to the pulses/revolution or distance/revolution specifications of your stepper/servo motor controller system.

16.5.2 Configuring an Axes group of Motion

S7-200 SMART CPU and STEP 7-Micro/WIN SMART V3 provide motion wizard for user to configure an Axes group of Motion. Refer to Configuring an Axis of Motion (Page 889) for information about how to configure an Axis of Motion.

Currently two axes group can be enabled for a project.



When an axes group is enabled, you can rename the axes group or keep the default name. Below table shows enabling corresponding single axes to configure axes group for 2D or 3D linear interpolation movement.

Linear interpolation movement	Enabling axes
2D movement	• X-axis: Axis 0 • Y-axis: Axis 1
	• X-axis: Axis 2 • Y-axis: Axis 3
3D movement	• X-axis: Axis 0 • Y-axis: Axis 1 • Z-axis: Axis 2

Note

Measurement and phasing for single axis.

The measurement system and the base unite of measurement for all axes in the axes group need to be configured consistently, and the phasing for directional control cannot be selected as "Single Phase (1 output)", otherwise, the axes group cannot get enabled.

Path planning

STEP 7-Micro/WIN SMART V3 supports user to plan multiple segments of paths for the Axes group of motion control.

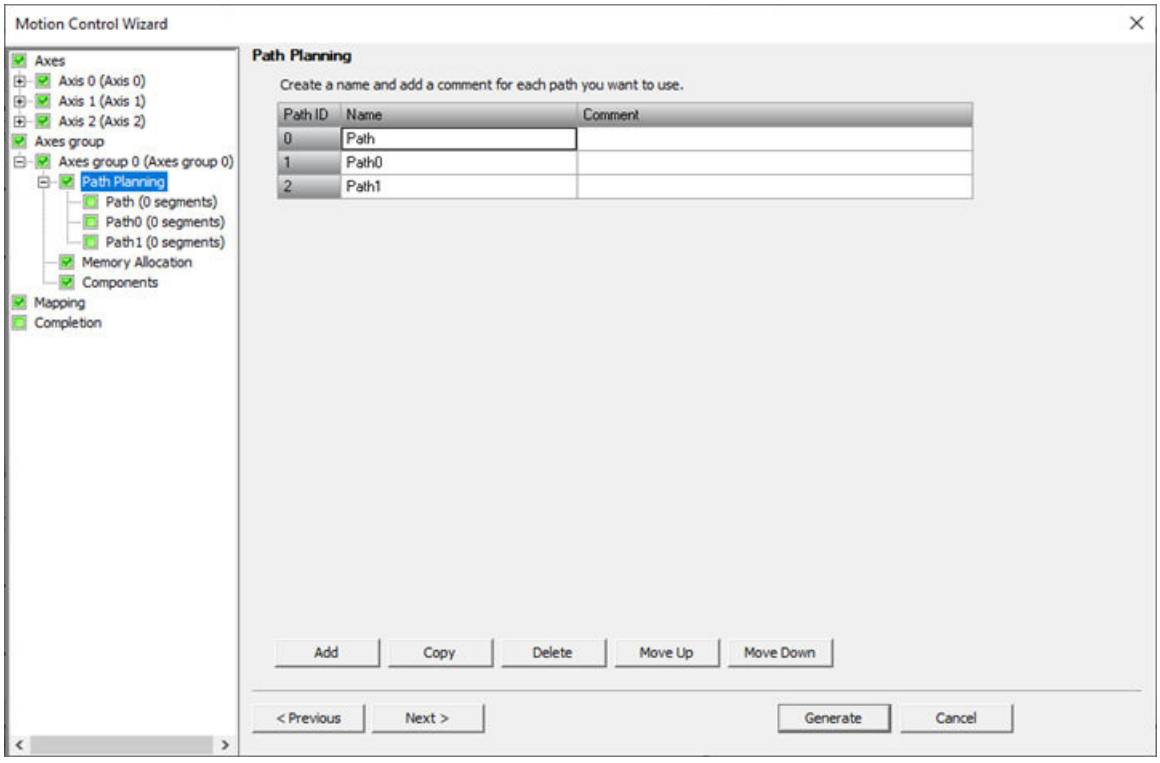
A path is a pre-defined motion description of a space curve that the tool center point (TCP) of an axes group moves along. It consists of a set of continuous positions and orientation information in 2/3D axes group movement space.

Note

The CPU only supports absolute position linear interpolation movement in path planning function for axes group..

Follow below steps to plan path:

1. Add paths according to the linear interpolation movement you want the axes group to run. You can create a name and add a comment for each path. You can add 128 paths at most.

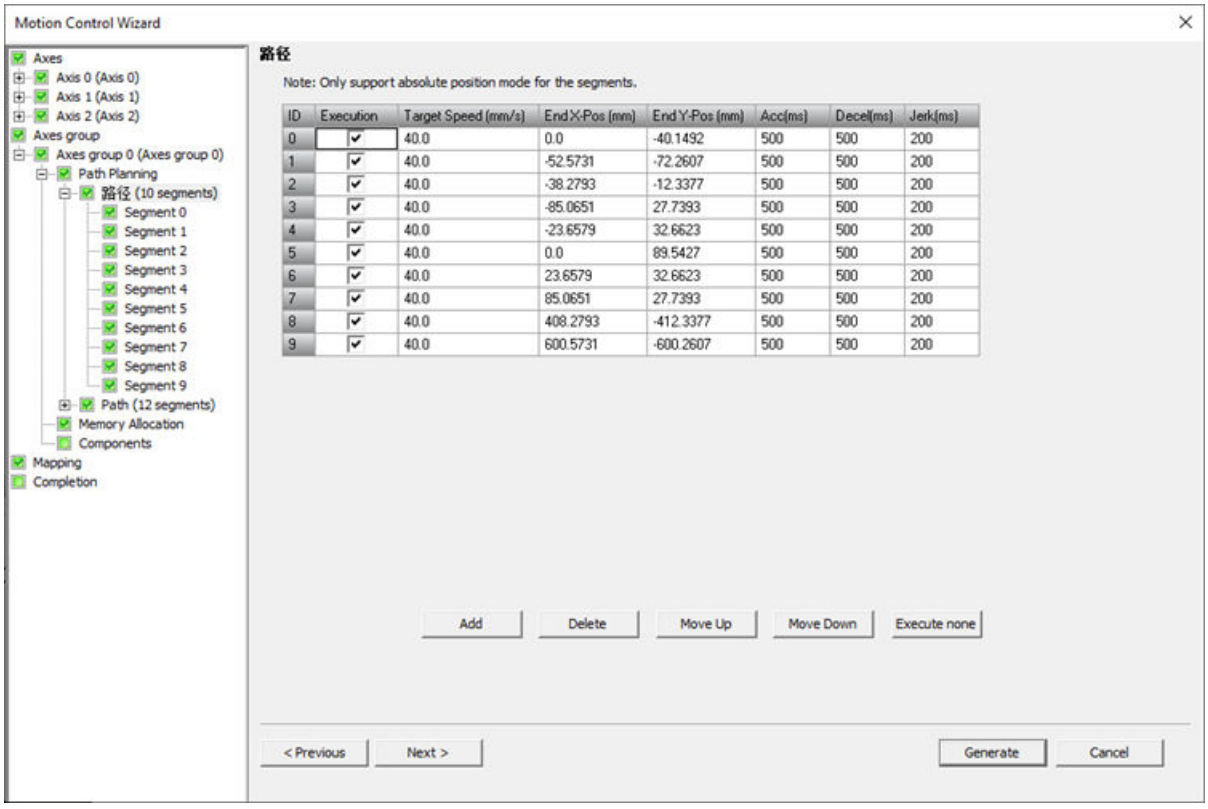


The motion wizard supports the following operations:

– Copy the selected path

– Change the sequence of the selected path by moving up or down the path table

2. Configure multiple segments for each path. You need configure in the segment table for the target speed, end X-Position, end Y-position, end Z-position (if the Axis 2 is enabled for 3D linear interpolation movement), acceleration time, deceleration time and jerk time.



Below are some regulations for segment table input. Input will be underlined in red if the input is not consistent with regulations.

Segment configuration	Input regulation
Execution	You can execute the movement of a segment by selecting the checkbox in the column "Execution", or don't execute the movement of a segment by deselecting the checkbox in the column "Execution". **Note:** • Segment 0 must be selected. • The segments must be selected consecutively.
Target speed	• Input > 0 • Input should be integer if the configured measure system in the motion wizard is relative pulses; Input should be floating if the configured measure system is engineering units
End position of axis X/Y/Z	• Input should be integer if the configured measure system in the motion wizard is relative pulses; Input should be floating if the configured measure system is engineering units
Acceleration time	• Input should be integer • 20 ms ≤ input ≤ 32000 ms
Deceleration time	• Input should be integer • 20 ms ≤ input ≤ 32000 ms
Jerk time	• 0 ms ≤ input ≤ 32000 ms

The motion wizard supports the following operations for configuring segments:

– Use shortcut `Ctrl+C` to copy the selected segment values and use shortcut `Ctrl+V` to paste into the selected line

– Change the sequence of the selected segment by moving up or down the table

– Copy the table of segment configuration with all the required values from an Excel spreadsheet and use shortcut `Ctrl+V` to paste into the segment configuration table in the Motion wizard

Note

Segment amount limit

- At least 1 segment shall be created in a path. Otherwise, a message pops up for user to delete the path containing no segment.
- The total number of segments under all paths cannot exceed 128.

Note

Paste to the segment table from Excel

- Each cell in the segment table supports input of 10 characters at most. If user try to paste item exceeding 10 characters, only content with the front 10 characters will be pasted into the cell.
- Invalid items pasted into the segment table are underlined in red as a reminder for user to do further check.

3. Configure buffer mode and trigger for start for each segment in the segment configuration page.
Buffer mode
Buffer mode defines the chronological sequence of the function block relative to the previous block.

– Buffered: the next function block controls the axes group as soon as the previous segment movement is completed.

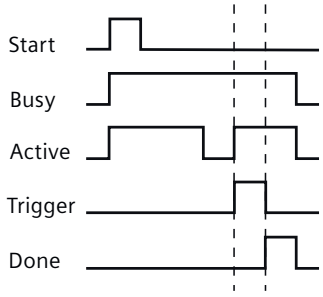
– Aborting: the next function block controls an ongoing segment to decelerate to stop and controls the new segment to accelerate to full execution.

For the first segment of the planned path, you can choose Aborting or Buffered as the buffer mode; for segments other than the first segment, only Buffered mode is supported.
When the first segment of the path is configured as Aborting, triggering GRP0_MOVEPATH will make the CPU abort the linear interpolation movement in execution and start to execute the planned path movement.
If the first segment in the current path is configured as Buffered, only when the last two segments of the previous path are under execution or finishes execution, user can trigger the execution of the current path.
Trigger for start
Segments other than the first one support the triggering function.
The first segment starts to run once GRP0_MOVEPATH (Page 965) subroutine is triggered.
Segments other than the first one start to run in below conditions:

– If the current segment is not configured with a trigger for start, it starts to run only when the previous segment finishes running.

– If the current segment is configured with a trigger for start, it starts to run only when the previous segment finishes running and the trigger rising edge comes for the current segment.

Below sequence diagram shows an example of the output status for the trigger for start setting in the axes group movement. In this case, 3 segments are configured for the path. No trigger for start is configured for the first two segments, while the third segment is configured with a trigger for start.



Allocating memory

Specify a starting address where the configuration will be placed in the data block. The wizard can also suggest an address that represents an unused block of V memory of the correct size.

Generating the configuration

1. After you have configured the Axes group of Motion, you simply click "Generate".
   The Motion wizard performs the following tasks:
   - Inserts the axes group configuration and the path and segment table into the system block and data block for your CPU program
   - Creates a global symbol table for the motion parameters
   - Adds the motion instruction subroutines into the project program block for you to use in your application
2. You can open the Motion wizard again to modify any configuration or path information.

Note

Because the Motion wizard makes changes to the program block, the data block, and the system block, you must download all three blocks to the CPU. Otherwise, the Axes group of Motion will not have all the program components that it needs for proper operation.

Changing the segment parameters defined in symbol table dynamically

You can change each segment parameters defined in the symbol table.

You can change the parameters of movement path segments dynamically as follows:

1. Create symbols for the following parameters of each segment and path in the symbol table:

Segment parameters	Symbols
Execution	AXESGRPx_Px_Sx_EXE
End position for axis X	AXESGRPx_Px_Sx_XPOS
End position for axis Y	AXESGRPx_Px_Sx_YPOS
End position for axis Z	AXESGRPx_Px_Sx_ZPOS
Target speed	AXESGRPx_Px_Sx_SPD
Acceleration time	AXESGRPx_Px_Sx_ACC
Deceleration time	AXESGRPx_Px_Sx_DEC
Jerk time	AXESGRPx_Px_Sx_JERK

2. Change value of each symbol as you need.

Note

Rules for multiple segments

- If you disable one segment among the consecutive segments, the segment you have disabled and the later segments will not run.
- If one segment is running or has completed running, and you modify the value of the bit "Execution" for this segment, the modification will not take effect until the next time you run the path.

16.5.3 Subroutines created by the Motion wizard for the Axes group of Motion

There are 4 Axes group of motion subroutines.

Motion subroutine	Description
GRPx_2/3D_MOVELINEAR (Page 963)	Command an interpolated linear 2/3D movement on an axes group from the actual position to a target absolute or relative position
GRPx_MOVEPATH (Page 965)	Command an axes group to move according to the path specified in multi-segment configuration
GRPx_RESET (Page 966)	Command an axes group to transit from ErrorStop state to the state of getting ready for linear interpolation movement.
GRPx_STOP (Page 968)	Command an axes group to stop the movement.

16.5.3.1 Guidelines for using the Motion subroutines

Some guidelines when using Axes group of motion subroutines are as follows:

- More than one Axes group of motion subroutines can be activated simultaneously. CPU can buffer 32 linear interpolation movement subroutines at most.
- Insert the AXISx_CTRL subroutine in your program to enable and initialize the axis of motion in the Axes group.
- Use GRP0_2/3D_MOVELINEAR or GRP0_MOVEPATH to command the CPU for Axes group movement.
- If the LMT+/LMT-/STP signal is triggered for single axis in the Axes group, or the MOD_EN parameter is turned off in AXISx_CTRL instruction, the running Axes group linear interpolation movement will stop immediately or decelerate to stop.
- When the CPU is in absolute mode, before triggering the Axes group movement, you need to use AXISx_RSEEK/AXISx_LDPOS to establish zero position for the Axes group.
- Use GRP0_RESET subroutine to command the Axes group to transit from ErrorStop state (Page 966) to the state of getting ready for linear interpolation movement.

16.5.3.2 GRPx_2/3D_MOVELINEAR subroutine

Table 16-50 GRPx_2/3D_MOVELINEAR

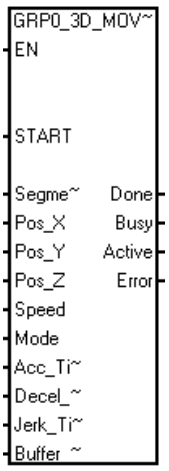
LAD / FBD	STL	Description
	**CALL GRPx_2D_MOVELINEAR, GRPx_3D_MOVELINEAR**	The GRPx_3D_MOVELINEAR subroutine commands the configured axes group on the linear interpolation movement from the actual position of the TCP (Tool Center point) to a target absolute or relative position. For axes group configured with 2 singles axes, the CPU will generate GRPx_2D_MOVELINEAR subroutine instruction.

Table 16-51 Parameters for the GRPx_2/3D_MOVELINEAR subroutine

Inputs/ Outputs	Data type	Operands	Description
EN	BOOL	I, Q, V, M, SM, S, T, C, L	Enable the subroutine. Ensure that the EN bit stays on until the Done bit signals that the execution of the subroutine has completed.
START	BOOL	I, Q, V, M, SM, S, T, C, L, Power Flow	Send a GRPx_2/3D _MOVELINEAR command to CPU.

Inputs/ Outputs	Data type	Operands	Description
Segment_ID	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD, Constant	A user specified unique parameter for each function block, which stands for the corresponding segment in the planned path.
Pos_X, Pos_Y, Pos_Z	DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *AC, *LD, Constant	Target X/Y/Z position parameter you set for this function block. Based upon the unit of measurement, the value is either a number of pulses/second (DINT) or the engineering units/second (REAL). Pos_Z parameter only exists for 3D linear interpolation movement.
Speed	DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *AC, *LD, Constant	Specify the maximum compound speed of movement, not necessarily reached. Based upon the unit of measurement, the value is either a number of pulses/second (DINT) or the engineering units/second (REAL).
Mode	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD, Constant	Specify the mode of the movement: • 0: Absolute mode • 1: Relative mode
Acc_Time	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *AC, *LD, Constant	Specify acceleration time (ms) of the movement.
Decel_Time	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *AC, *LD, Constant	Specify deceleration time (ms) of the movement.
Jerk_Time	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *AC, *LD, Constant	Specify jerk time (ms) of the movement.
BufferMode	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD, Constant	Define the chronological sequence of the function block relative to the previous block. BufferMode to select: • 0: Aborting • 1: Buffered
Done	BOOL	I, Q, V, M, SM, S, T, C, L	Parameter turns on when the commanded action has been completed successfully.
Busy	BOOL	I, Q, V, M, SM, S, T, C, L	Parameter turns on when the function block is triggered but not finished.
Active	BOOL	I, Q, V, M, SM, S, T, C, L	Parameter turns on when the function block is outputting pulse.
Error	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD	Certain error has occurred on the instruction execution.

Note

START parameter

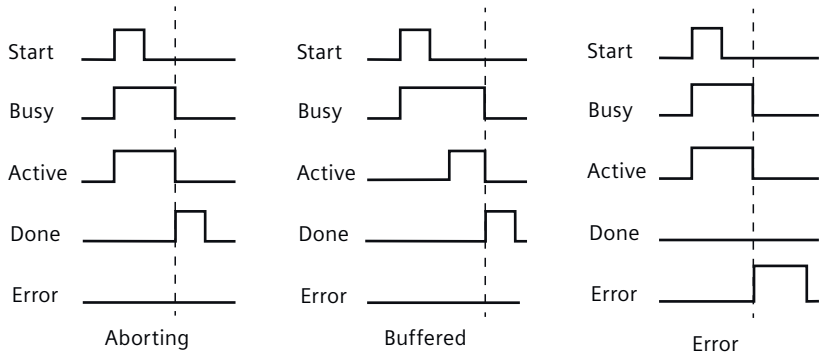
Use edge signal as the trigger signal for the linear interpolation movement subroutine execution to ensure the same linear interpolation movement subroutine is triggered only once.

Note

Acc_Time/Decel_Time/Jerk_Time parameter

The time related parameters configured in this subroutine work for Axes group linear interpolation movement, while time related parameters configured for each single Axis in the Motion Wizard don't work for Axes group linear interpolation movement.

The Done, Active and Busy output status in the axes group is displayed as below:



Segment_ID

Segment_ID parameter helps user to form a clear movement path, and it represents each corresponding line segment in the movement path. Follow below tips when using Segment_ID:

- Segment_ID doesn't represent segment execution sequence.
- Segment_ID ranges from 0 to 127.
- Segment_ID can be duplicate in different move linear subroutines in the same program.
  In some application scenarios, it is desirable to program a complex process with several sub-processes. In order to visualize the order of execution of sub-processes, the Segment_ID can be named in numerical order in the sub-process instructions. In this case, the Segment_ID can be repeated in different sub-processes.
  If the CPU buffers multiple segments with the same Segment_ID in the program, then these duplicate Segment_ID share the same running status, that is, Done, Active, Busy, Error status.

16.5.3.3 GRPx_MOVEPATH subroutine

Table 16-52 GRPx_MOVEPATH

LAD / FBD	STL	Description
	CALL GRPx_MOVEPATH	The GRPx_MOVEPATH subroutine commands an axes group to move according to the path specified in multiple segment configuration.

Table 16-53 Parameters for the GRPx_MOVEPATH subroutine

Inputs/ Outputs	Data type	Operands	Description
EN	BOOL	I, Q, V, M, SM, S, T, C, L	Enable the subroutine. Ensure that the EN bit stays on until the Done bit signals that the execution of the subroutine has completed.
START	BOOL	I, Q, V, M, SM, S, T, C, L, Power Flow	Send a GRPx_MOVEPATH command to CPU.
Path_ID	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD, Constant	Specify ID of path. The ID input number must be between 0 and 127. Otherwise, the subroutine returns an error.
Done	BOOL	I, Q, V, M, SM, S, T, C, L	Parameter turns on when the commanded action has been completed successfully.
Busy	BOOL	I, Q, V, M, SM, S, T, C, L	Parameter turns on when the function block is triggered but not finished.
Active	BOOL	I, Q, V, M, SM, S, T, C, L	Parameter turns on when the function block is outputting pulse.
Error	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD	Certain error has occurred on the instruction execution.
Err_Segment	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD	Specify the ID of the segment that encounters execution error.
Cur_Segment	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD	Specify the ID of the current segment under execution.

Note

START parameter

Use edge signal as the trigger signal for the linear interpolation movement subroutine execution to ensure the same linear interpolation movement subroutine is triggered only once.

16.5.3.4 GRPx_RESET subroutine

Table 16-54 GRPx_MOVEPATH

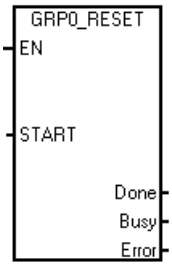
LAD / FBD	STL	Description
	`CALL GRPx_RESET`	The GRPx_RESET subroutine commands an axes group to transit from ErrorStop state to the state of getting ready for linear interpolation movement.

Table 16-55 Parameters for the GRPx_RESET subroutine

Inputs/ Outputs	Data type	Operands	Description
EN	BOOL	I, Q, V, M, SM, S, T, C, L	Enable the subroutine. Ensure that the EN bit stays on until the Done bit signals that the execution of the subroutine has completed.
START	BOOL	I, Q, V, M, SM, S, T, C, L, Power Flow	Send a GRPx_RESET command to CPU.
Done	BOOL	I, Q, V, M, SM, S, T, C, L	Parameter turns on when the commanded action has been completed successfully.
Busy	BOOL	I, Q, V, M, SM, S, T, C, L	Parameter turns on when the function block is triggered but not finished.
Error	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD	Certain error has occurred on the instruction execution.

Note

START parameter

Use edge signal as the trigger signal for the linear interpolation movement subroutine execution to ensure the same linear interpolation movement subroutine is triggered only once.

Axes group in ErrorStop state

Calling GRPx_RESET instruction doesn't affect the output of other FB instances. GRPx_RESET instruction takes into effect only when the axes group is in ErrorStop state. If calling this FB when the axes group is not in ErrorStop state, this FB will indicate error. Under the following conditions, axes group enters ErrorStop state after axes group is stopped:

- Triggering axes group linear interpolation movement instruction (GRPx_2/3D_MOVELINEAR or GRPx_MOVEPATH) with invalid parameters, for example, invalid operation mode, invalid speed, invalid acceleration time and so on.
- Encountering LMT+/LMT-/STP signal during axes group linear interpolation movement.
- Triggering axes group linear interpolation movement under following invalid conditions:
  - Triggering axes group linear interpolation movement in absolute mode without setting zero position.
  - Buffering axes group linear interpolation movement when exceeding the motion buffer limit of 32.
  - Triggering GRPx_2/3D_MOVELINEAR instruction when the motion buffer queue is busy. The motion buffer queue is busy when executing GRPx_MOVEPATH instruction and not all segments of the path have been added into the motion queue.
  - Not all axes in the axes group are in ready state.
  - At least one axis in the axes group has invalid configuration.

16.5.3.5 GRPx_STOP subroutine

Table 16-56 GRPx_STOP

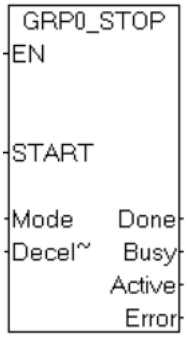
LAD/FBD	STL	Description
	**Call GRPx_STOP, START, Mode, Decel_Time, Done, Busy, Active, Error**	The **GRPx_STOP** subroutine commands an axes group to stop the movement.

Table 16-57 Parameters for the GRPx_STOP subroutine

Inputs/ Outputs	Data type	Operands	Description
EN	BOOL	I, Q, V, M, SM, S, T, C, L	Enable the subroutine. Ensure that the EN bit stays on until the Done bit signals that the execution of the subroutine has completed.
START	BOOL	I, Q, V, M, SM, S, T, C, L, Power Flow	Send a GRPx_STOP command to CPU.
Mode	BYTE	IB, QB, VB, MB, SMB, SB, LB,AC, *VD, *AC, *LD, Constant	Specify the stop mode of axes group: • 0: Decelerated stop The axes group movement decelerates and stops within the specific deceleration time. • 1: Immediate stop The axes group movement stops immediately.
Decel_Time	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *AC, *LD, Constant	Specify deceleration time (ms) of the stop movement. **Note**: The deceleration time range is from 20 ms to 32000 ms. **Note**: You need to configure the parameter Decel_Time if you select Mode 0.
Done	BOOL	I, Q, V, M, SM, S, T, C, L	Parameter turns on when the commanded action has been completed successfully.
Busy	BOOL	I, Q, V, M, SM, S, T, C, L	Parameter turns on when the function block is triggered but not finished.
Active	BOOL	I, Q, V, M, SM, S, T, C, L	Parameter turns on when the function block is stopping the axes group movement.
Error	BYTE	IB, QB, VB, MB, SMB, SB, LB,AC, *VD, *AC, *LD	Certain error has occurred on the instruction execution.

Note

If an axes group is configured with the LMT+/LMT-/STP input signal or is executing the GRPx_STOP instruction, an on-going decelerated stop will be interrupted by an immediate stop command.

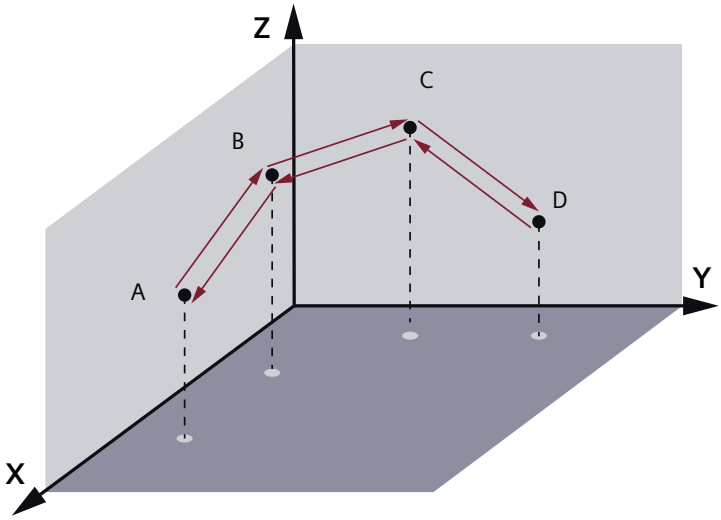
Note

After triggering the GRPx_STOP command, the axes group stops in the straight line where the current movement is.

16.5.4 Axes group of Motion example programs

The example program shows a simple absolute move that uses the AXISx_CTRL, AXISx_RSEEK, and GRPx_3D_MOVELINEAR subroutines to perform a pick-and-place operation. This program requires RP seek for axes in the axes group, and the moving distance can be measured in engineering units or relative pulses. When I0.0 (Start) turns on, the machine starts. When I0.2 (E_Stop for Axis X), or I0.3 (E_Stop for Axis Y), or I0.4 (E_Stop for Axis Z) turns on, the machine aborts any motion and decelerates to stop.

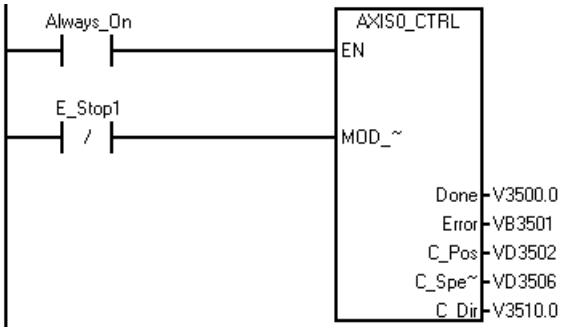
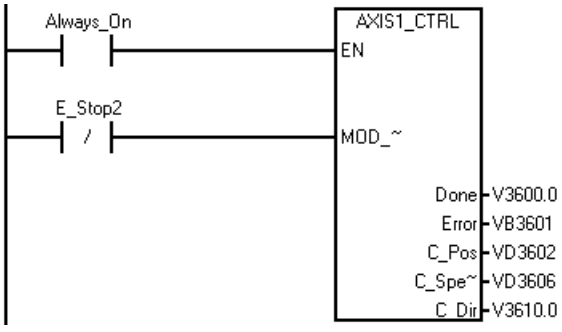
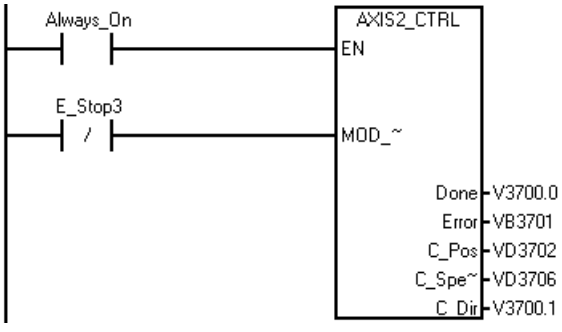
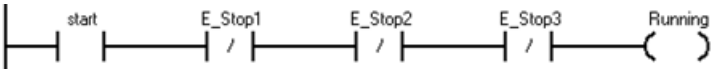
Below is the moving path of the tong on the pick-and-place machine:

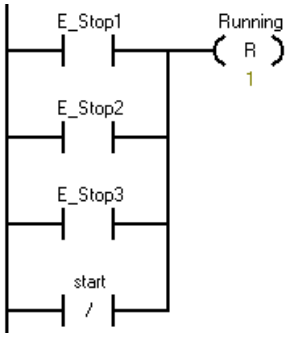
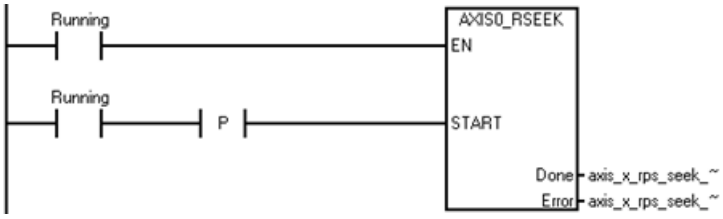
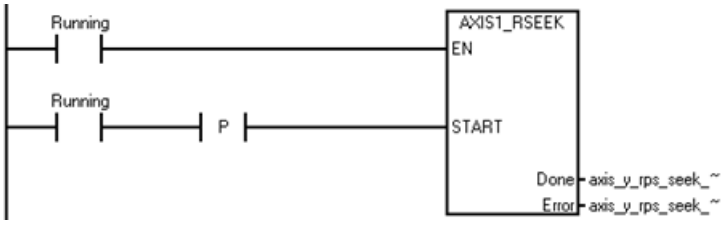
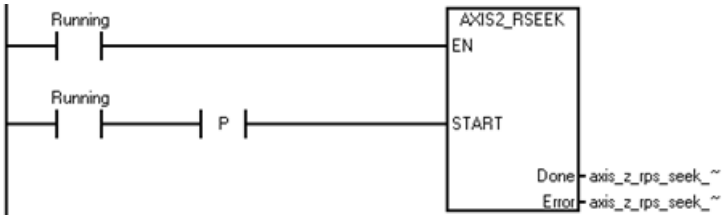


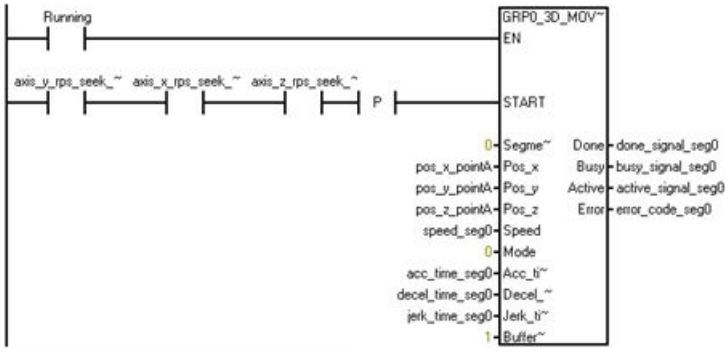
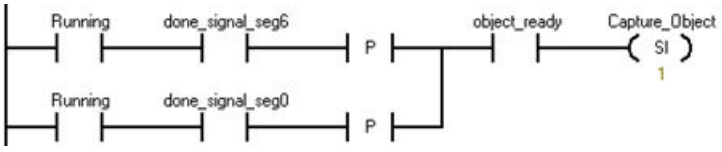
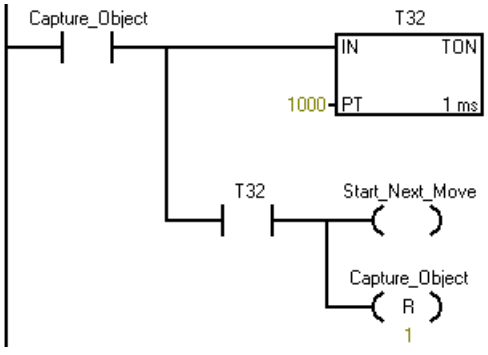
When the machine starts, the system seeks for zero position point first, and the tong moves to the pick position A. The system waits for the signal I0.1 that indicates the capture object is ready. The tong picks up the object by turning on Q1.1 (turn on for 1 second) and follow the track 1, 2 and 3 to position D. When the tong reaches to position D, it releases the object by

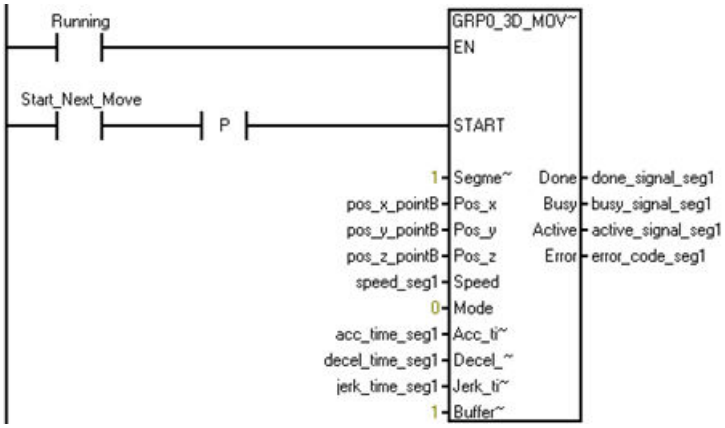
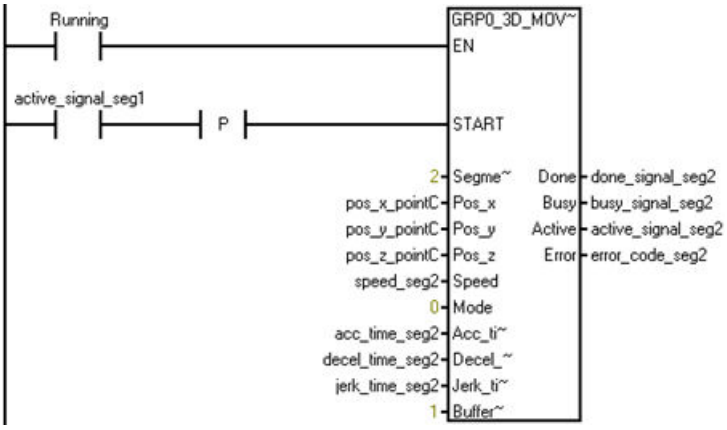
turning on Q1.2 (turn on for 1 second). The tong moves back to position A by following the track 4, 5 and 6 and waits for next capture.

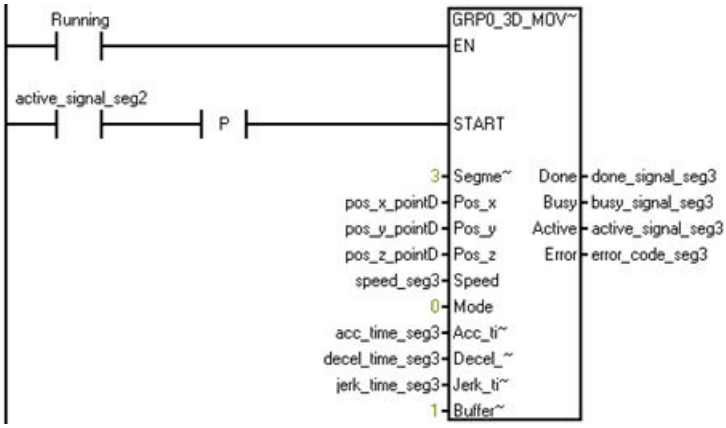
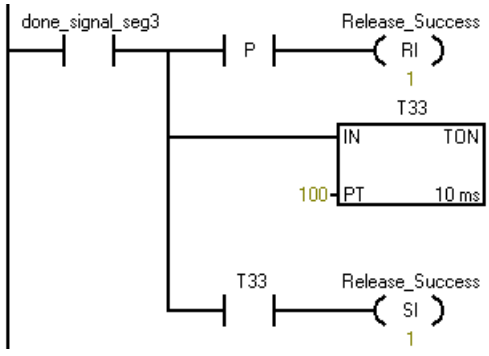
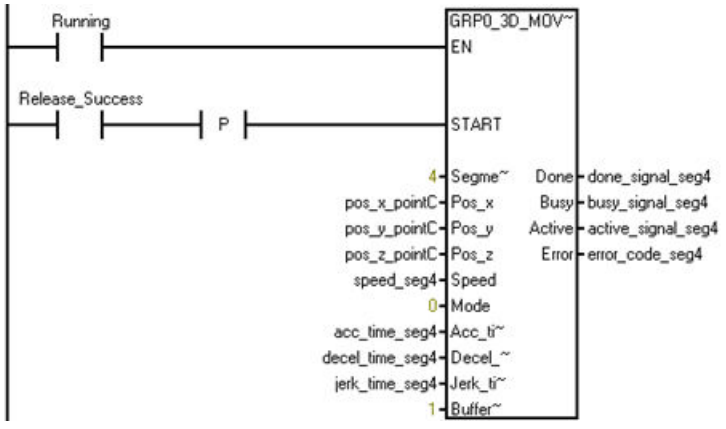
Table 16-58 Example: Axes group of Motion simple absolute move (pick-and-place application)

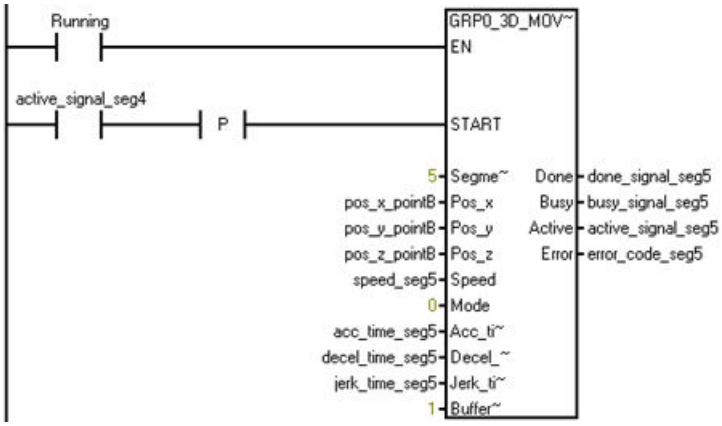
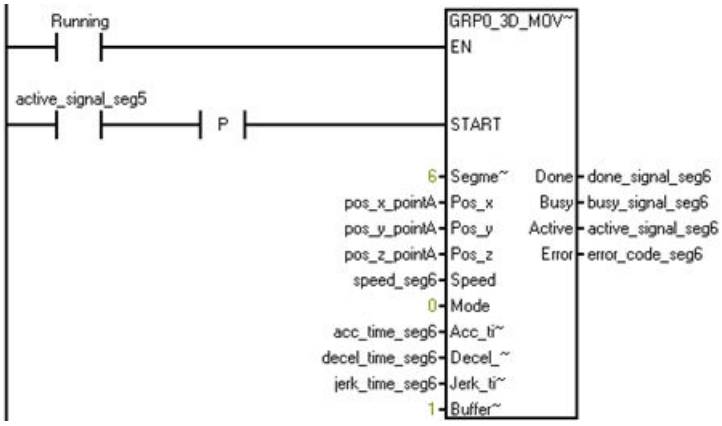
LAD/FBD	Description	STL
Network 1:  	Initialize Axis X.	**LD Always_On** **= L60.0** **LDN E_Stop1** **= L63.7** **LD L60.0** **CALL AXIS0_CTRL, L63.7, V3500.0, VB3501, VD3502, VD3506, V3510.0**
Network 2:  	Initialize Axis Y.	**LD Always_On** **= L60.0** **LDN E_Stop2** **= L63.7** **LD L60.0** **CALL AXIS1_CTRL, L63.7, V3600.0, VB3601, VD3602, VD3606, V3610.0**
Network 3:  	Initialize Axis Z.	**LD Always_On** **= L60.0** **LDN E_Stop3** **= L63.7** **LD L60.0** **CALL AXIS2_CTRL, L63.7, V3700.0, VB3701, VD3702, VD3706, V3700.1**
Network 4:  	When the start signal is ON and the stop signal is OFF, the system is ready to run.	**LD start** **AN E_Stop1** **AN E_Stop2** **AN E_Stops3** **= Running**

LAD/FBD	Description	STL
Network 5:  	When the start signal is OFF and the start signal is ON, the system stops.	**LD start** **O E_Stop1** **O E_Stop2** **ON start** **= Running, 1**
Network 6:  	When the start signal is ON, the system starts to seek zero position. This network is used for Axis X to seek zero position.	**LD Running** **= L60.0** **EU** **= L63.7** **LD L60.0** **CALL AXIS0_RSEEK, L63.7, axis_x_rps_seek_done, axis_x_rps_seek_error**
Network 7:  	When the start signal is ON, the system starts to seek zero position. This network is used for Axis Y to seek zero position.	**LD Running** **= L60.0** **EU** **= L63.7** **LD L60.0** **CALL AXIS1_RSEEK, L63.7, axis_y_rps_seek_done, axis_y_rps_seek_error**
Network 8:  	When the start signal is ON, the system starts to seek zero position. This network is used for Axis Z to seek zero position.	**LD Running** **= L60.0** **EU** **= L63.7** **LD L60.0** **CALL AXIS0_RSEEK, L63.7, axis_zx_rps_seek_done, axis_z_rps_seek_error**

LAD/FBD	Description	STL
Network 9:  	When Axis X, Axis Y and Axis Z return to the zero position, the tong moves to the position over point A.	**LD Running** **= L60.0** **LD axis_y_rps_seek_done** **A axis_x_rps_seek_done** **A axis_z_rps_seek_done** **EU** **= L63.7** **LD L60.0** **CALL GRP0_3D_MOVELINEAR, L63.7, 0, pos_x_pointA, pos_y_pointA, pos_z_pointA, speed_seg0, 0, acc_time_seg0, decel_time_seg0, jerk_time_seg0, 1, done_signal_seg0, busy_signal_seg0, active_signal_seg0, error_code_seg0**
Network 10:  	Picking starts when the capture object is ready.	**LD Running** **A done_signal_seg6** **EU** **LD Running** **A done_signal_seg0** **EU** **OLD** **A object_ready** **SI Capture_Object, 1**
Network 11:  	Picking ends 1 second later.	**LD Capture_Object** **TON T32, 1000** **A T32** **= Start_Next_Move** **R Capture_Object, 1**

LAD/FBD	Description	STL
Network 12:  	The tong moves from point A to point B in the 3D space after the picking is finished. The moving path number equals to the Segment_ID, 1.	**LD Running** **= L60.0** **LD Start_Next_Move** **EU** **= L63.7** **LD L60.0** **CALL GRP0_3D_MOVELINEAR, L63.7, 1, pos_x_pointB, pos_y_pointB, pos_z_pointB, speed_seg1, 0, acc_time_seg1, decel_time_seg1, jerk_time_seg1, 1, done_signal_seg1, busy_signal_seg1, active_signal_seg1, error_code_seg1**
Network 13:  	The tong moves from point B to point C in the 3D space. The moving path number equals to the Segment_ID, 2.	**LD Running** **= L60.0** **LD active_signal_seg1** **EU** **= L63.7** **LD L60.0** **CALL GRP0_3D_MOVELINEAR, L63.7, 2, pos_x_pointC, pos_y_pointC, pos_z_pointC, speed_seg2, 0, acc_time_seg2, decel_time_seg2, jerk_time_seg2, 1, done_signal_seg2, busy_signal_seg2, active_signal_seg2, error_code_seg2**

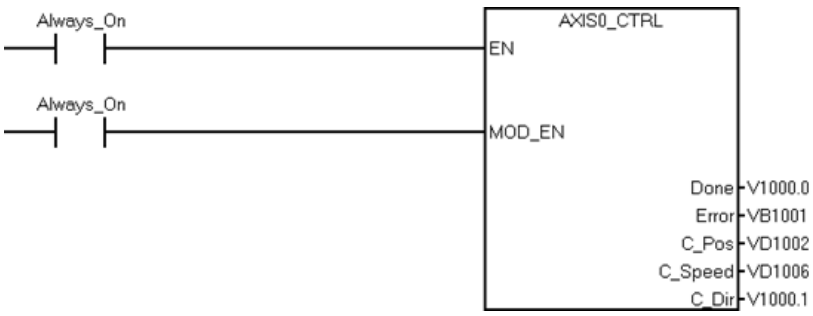
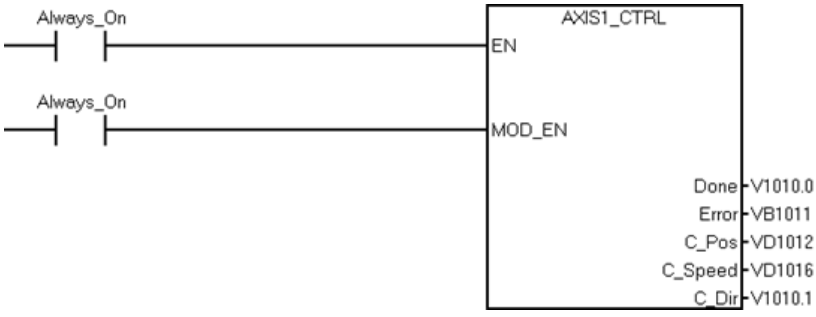
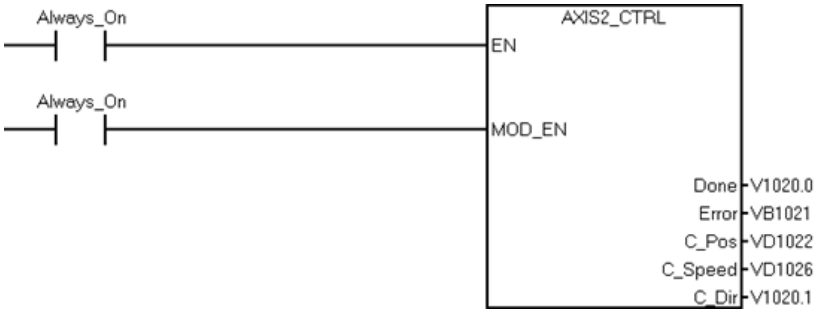
LAD/FBD	Description	STL
Network 14:  	The tong moves from point C to point D in the 3D space. The moving path number equals to the Segment_ID, 3.	**LD Running** **= L60.0** **LD active_signal_seg2** **EU** **= L63.7** **LD L60.0** **CALL GRP0_3D_MOVELINEAR, L63.7, 3, pos_x_pointD, pos_y_pointD, pos_z_pointD, speed_seg3, 0, acc_time_seg3, decel_time_seg3, jerk_time_seg3, 1, done_signal_seg3, busy_signal_seg3, active_signal_seg3, error_code_seg3**
Network 15:  	The tong starts to place the captured object. Placing ends 1 second later.	**LD done_signal_seg3** **LPS** **EU** **RI Release_Success, 1** **LRD** **TON T33, 100** **LPP** **A T33** **SI Release_Success, 1**
Network 16:  	The tong moves from point D to point C in the 3D space after the placing is finished. The moving path number equals to the Segment_ID, 4.	**LD Running** **= L60.0** **LD Release_Success** **EU** **= L63.7** **LD L60.0** **CALL GRP0_3D_MOVELINEAR, L63.7, 4, pos_x_pointC, pos_y_pointC, pos_z_pointC, speed_seg4, 0, acc_time_seg4, decel_time_seg4, jerk_time_seg4, 1, done_signal_seg4, busy_signal_seg4, active_signal_seg4, error_code_seg4**

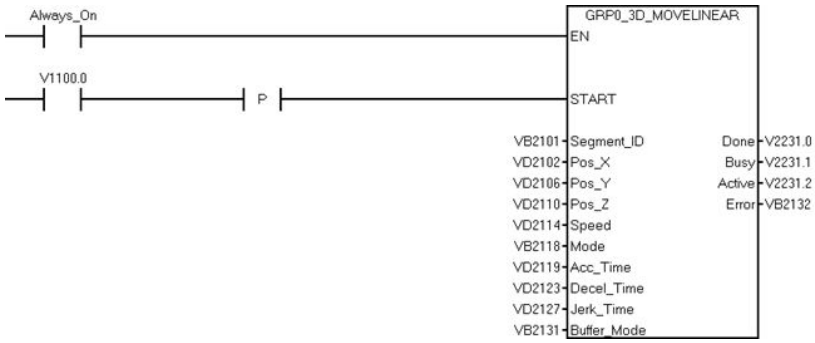
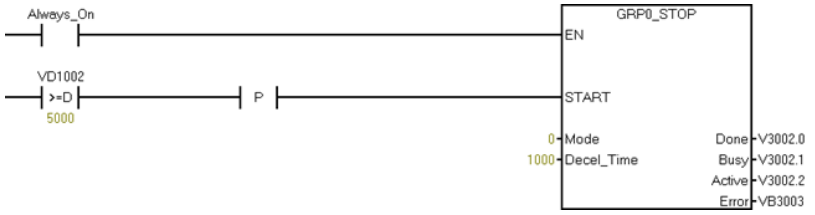
LAD/FBD	Description	STL
Network 17:  	The tong moves from point C to point B in the 3D space. The moving path number equals to the Segment_ID, 5.	**LD Running** **= L60.0** **LD active_signal_seg4** **EU** **= L63.7** **LD L60.0** **CALL GRP0_3D_MOVELINEAR, L63.7, 5, pos_x_pointB, pos_y_pointB, pos_z_pointB, speed_seg5, 0, acc_time_seg5, decel_time_seg5, jerk_time_seg5, 1, done_signal_seg5, busy_signal_seg5, active_signal_seg5, error_code_seg5**
Network 18:  	The tong moves from point B to point A in the 3D space, and gets ready to the next picking work. The moving path number equals to the Segment_ID, 6.	**LD Running** **= L60.0** **LD active_signal_seg5** **EU** **= L63.7** **LD L60.0** **CALL GRP0_3D_MOVELINEAR, L63.7, 6, pos_x_pointA, pos_y_pointA, pos_z_pointA, speed_seg6, 0, acc_time_seg6, decel_time_seg6, jerk_time_seg6, 1, done_signal_seg6, busy_signal_seg6, active_signal_seg6, error_code_seg6**

16.5.4.1 Example of using GPRx_STOP subroutine to stop axes group movement

The example program shows how to use GPR0_STOP subrountine to stop the axes group movement.

Table 16-59 Example of axes group stop

LAD/FBD	Description	STL
Network 1:  	Enable axis 0.	**LD Always_On:SM0.0** **= L60.0** **LD Always_On:SM0.0** **= L63.7** **LD L60.0** **CALL AXIS0_CTRL:SBR1, L63.7, axis0_ctl_done:V1000.0, VB1001, x_position:VD1002, VD1006, V1000.1**
Network 2:  	Enable axis 1.	**LD Always_On:SM0.0** **= L60.0** **LD Always_On:SM0.0** **= L63.7** **LD L60.0** **CALL AXIS1_CTRL:SBR12, L63.7, V1010.0, VB1011, VD1012, VD1016, V1010.1**
Network 3:  	Enable axis 2.	**LD Always_On:SM0.0** **= L60.0** **LD Always_On:SM0.0** **= L63.7** **LD L60.0** **CALL AXIS2_CTRL:SBR23, L63.7, V1020.0, VB1021, VD1022, VD1026, V1020.1**

LAD/FBD	Description	STL
Network 4:  	Trigger the 3D-move-linear movement of the axis group.	**LD Always_On:SM0.0** **= L60.0** **LD start_movement:V1100.0** **EU** **= L63.7** **LD L60.0** **CALL GRP0_3D_MOVELINEAR:SBR34, L63.7, segment:VB2101, Pos_x:VD2102, Pos_y:VD2106, Pos_z:VD2110, speed:VD2114, mode:VB2118, acc:VD2119, decel:VD2123, jerk:VD2127, buffer_mode:VB2131, done:V2231.0, busy:V2231.1, active:V2231.2, error:VB2132**
Network 5:  	When axis X moves for 5000 pulses, the command GPRO-STOP triggers a decelerated stop. The deceleration time is 1000 ms.	**LD Always_On:SM0.0** **= L60.0** **LDD>= x_position:VD1002, 5000** **EU** **= L63.7** **LD L60.0** **CALL GRP0_STOP:SBR36, L63.7, 0, 1000, stp_done:V3002.0, stp_busy:V3002.1, stp_active:V3002.2, stp_error:VB3003**

16.6 Stopping the Axes group movement

The running axes group can be stopped in below three methods:

- If any axis in the axes group has been configured with LMT+/LMT-/STP (Page 889) input signal, when the signal comes, based on the configuration, the running axes group will stop immediately or decelerate to stop in the planned move path.
- User can also turn off the MOD_EN parameter (Page 904) in the AXISx_CTRL instruction to abort the axes group motion instruction under execution.
- Use the GRPx_STOP (Page 968)subroutine.

16.7 Error codes for the Axes group of Motion (WORD at SMW820 or SMW890)

If there is any error in the axes group configuration after the configuration is downloaded into the CPU, it will appear in the WORD SMW820 for axes group 0 and in the WORD SMB890 for axes group 1. Error codes and meanings are as follows:

Error code	Description
43	Axes group configuration block pointer error
44	Axes group configuration block size error
45	Invalid SBD0 record size of axes group
46	Illegal SDB0 format of axes group
47	Illegal dimension of axes group
48	Illegal number of axes in the axes group dimension
49	Illegal axes group Kinematics model
50	The axis contained in axes group doesn't exist
51	Configuration block pointer error
52	Duplicate configuration of axis
53	Axes group not supported by the type of CPU
54 to 127	Reserved

16.8 Error codes for the Axes group of Motion instruction (seven LS bits of SMB823 or SMB893)

If the CPU detects parameter error in the triggered axes group of motion during linear interpolation movement, corresponding error code appears in the error parameter. Then the axes group of motion will be aborted immediately. Error code 66 appears for the running instruction and buffered instructions due to motion abort.

Error code	Description
50	Axes group motion instruction aborted by user
51	Invalid axes group configuration
52	Axes group instruction cannot be executed as a path instruction is under execution
53	Illegal operation mode
54	Illegal buffer mode
55	Illegal speed value
56	Illegal acceleration time value
57	Illegal deceleration time value
58	Illegal jerk time value
59	Not all axes in the axes group have established zero point when running in absolute mode
60	Illegal axes group moving distance
61	Buffer mode not supported
62	Insufficient distance to perform specified acceleration/deceleration

Error code	Description
63	The buffered linear interpolation movement has exceeded the supported maximum buffer number of 32.
64	Not all axes in the axes group are in ready status
65	Axes group speed out-of-range
66	Aborted due to other axes group instruction parameter error
67	Reserved
68	Segment ID out-of-range (0-127)
69	Path ID is not configured or out-of-range (0-127)
70	Axes group cannot run due to STOP signal on single axis is activated
71	Axes group cannot run due to positive limit on single axis is reached
72	Axes group cannot run due to negative limit on single axis is reached
73	Calling GRPx_RESET when the axes group is not in ErrorStop state.
74	A decelerated stop is under progress, and cannot be aborted by another decelerated stop command.
75	Mode for GRPx_STOP instruction is illegal. Only Mode 0 (Decelerated stop) and Mode 1 (Immediate stop) are supported.
76	The on-going movement is aborted by the GRPx_STOP instruction.
77	The decelerated stop triggered by the GRPx_STOP instruction is aborted by an immediate stop.

16.9 Interactions between Axis and Axis group

An axes group is constituted of 2 or 3 configured axes. Execution of single Axis and Axes group are mutually exclusive, while certain single Axis motion instructions can be called for Axes group movement preparation.

Instructions of single Axis and Axes group

Instructions of single Axis and Axes group are mutually exclusive when under execution.

When the GRP0_2/3D_MOVELINEAR or GRP0_MOVEPATH instruction is under execution, commanding a single axis movement instruction will not be accepted and performed by the CPU, and the corresponding error code will be returned and displayed in the single axis function block. The axes group linear interpolation movement will continue.

The reversed case is also applicable that the running single axis instruction will not be interrupted by the commanding of an axes group linear interpolation movement instruction.

Axes group movement

Follow below steps to program the axes group movement after configuration in the Motion Wizard.

1. Initialize axes group
   Use AXISx_CTRL (Page 904) instruction to initialize each axis, hence to activate the axes group.
2. Seek reference point position or establish a new position for axes group
   Use AXISx_RSEEK (Page 910) or AXISx_LDPOS (Page 912) instructions to seek reference point position or establish a new position for each axis, hence to work on the axes group.
3. Program axes group movement
   Use GRP0_2/3D_MOVELINEAR (Page 963) or GRP0_MOVEPATH (Page 965) to command the CPU for axes group movement.

Technical specifications

A

A.1 General specifications

A.1.1 General technical specifications

Standards compliance

The S7-200 SMART automation system complies with the following standards and test specifications. The test criteria for the S7-200 SMART automation system is based on these standards and test specifications.

CE approval

The S7-200 SMART Automation System satisfies requirements and safety related objectives according to the EC directives listed below, and conforms to the harmonized European standards (EN) for the programmable controllers listed in the Official Journals of the European Community.

- EC Directive 2006/95/EC (Low Voltage Directive) "Electrical Equipment Designed for Use within Certain Voltage Limits"
  - EN 61131-2: Programmable controllers - Equipment requirements and tests
- EC Directive 2004/108/EC (EMC Directive) "Electromagnetic Compatibility"
  - Emission standard
    EN 61000-6-4: AI: Industrial Environment
  - Immunity standard
    EN 61000-6-2: Industrial Environment

The CE Declaration of Conformity is held on file available to competent authorities at:

Siemens Aktiengesellschaft
DI FA
Gleiwitzer Str. 555
DE-90475 Nuremberg

UK Conformity Assessed marking

The S7-200 SMART Automation System complies with the designated British standards (BS) for programmable logic controllers published in the official consolidated list of the British

Government. The S7-200 SMART Automation System meets the requirements and protection targets of the following regulations and related amendments:

- Electrical Equipment (Safety) Regulations 2016 (Low-Voltage)
- Electromagnetic Compatibility Regulations 2016 (EMC)
- Regulations on the restriction o the use of certain hazardous substances in electrical and electronic equipment 2012 (RoHS)

UK Declarations of Conformity for the respective authorities are available from:

Siemens plc
Princess Road
Manchester
M20 2UR
United Kingdom

The UK Declaration of Conformity is also available for download from the Siemens Industry Online Support website under the keyword "Declaration and Conformity".

Industrial environments

The S7-200 SMART automation system is designed for use in industrial environments.

Table A-1 Industrial environments

Application field	Noise emission requirements	Noise immunity requirements
Industrial	EN 61000-6-4	EN 61000-6-2

Electromagnetic compatibility

Electromagnetic Compatibility (EMC) is the ability of an electrical device to operate as intended in an electromagnetic environment and to operate without emitting levels of electromagnetic interference (EMI) that may disturb other electrical devices in the vicinity.

Table A-2 Immunity per EN 61000-6-2

Electromagnetic compatibility - Immunity per EN 61000-6-2	
EN 61000-4-2 Electrostatic discharge	±2kV; ±4kV; ±8 kV air discharge to all surfaces ±4 kV contact discharge to exposed conductive surfaces
EN 61000-4-3 Radiated, radio-frequency, electromagnetic field immunity test	80 to 1000 MHz, 10 V/m, 80% AM at 1 kHz 1.4 to 6.0 GHz, 3 V/m, 80% AM a 1 kHz
EN 61000-4-4 Fast transient burst	±2 kV, 5 kHz and 100 kHz with coupling network to AC and DC system power ±2 kV, 5 kHz and 100 kHz coupling clamp to I/O
EN 61000-4-5 Surge immunity	AC systems ±2 kV common mode, ±1kV differential mode DC systems ±1 kV common mode, ±0.5kV differential mode
EN 61000-4-6 Immunity to conducted disturbances	150 kHz to 80 MHz, 10 V RMS, 80% AM at 1 kHz

Electromagnetic compatibility - Immunity per EN 61000-6-2	
EN 61000-4-11 Voltage dips	AC systems 0% for 1 cycle, 40% for 12 cycles and 70% for 30 cycles at 60 Hz 0% for 1 cycle, 40% for 10 cycles and 70% for 25 cycles at 50 Hz
EN 61000-4-29 Voltage dips	DC systems 10ms

Table A-3 Conducted and radiated emissions per EN 61000-6-4

Electromagnetic compatibility - Conducted and radiated emissions per EN 61000-6-4		
Conducted Emissions	0.15 MHz to 0.5 MHz	<79dB (µV) quasi-peak; <66 dB (µV) average
EN 55011, Class A, Group 1	0.5 MHz to 30 MHz	<73dB (µV) quasi-peak; <60 dB (µV) average
Radiated Emissions	30 MHz to 230 MHz	<50dB (µV/m) quasi-peak; measured at 3 m
EN 55011, Class A, Group 1	230 MHz to 1 GHz	<57dB (µV/m) quasi-peak; measured at 3 m
	1 GHz to 3 GHz	<76dB (µV/m) peak, 56dB(µV/m) rms; measured at 3 m
	3 GHz to 6 GHz	<80dB (µV/m) peak, 60dB(µV/m) rms; measured at 3 m

Environmental conditions

Table A-4 Transport and storage

Environmental conditions - Transport and storage	
EN 60068-2-2, Test Bb, Dry heat and EN 60068-2-1, Test Ab, Cold	-40 °C to +70 °C
EN 60068-2-30, Test Db, Damp heat	25 °C to 55 °C, 95% humidity
EN 60068-2-14, Test Na, temperature shock	-40 °C to +70 °C, dwell time 3 hours, 2 cycles
EN 60068-2-32, Free fall	0.3 m, 5 times, product packaging
Atmospheric pressure	1139 to 660 h Pa (corresponding to an altitude of -1000 to 3500 m)

Table A-5 Operating conditions

Environmental conditions - Operating	
Ambient temperature range (Inlet Air 25 mm below unit)	-20 °C to 60 °C horizontal mounting -20 °C to 50 °C vertical mounting 95% non-condensing humidity
Atmospheric pressure	1139 to 795 hPa (corresponding to an altitude of -1000 to 2000 m)
Concentration of contaminants	$S0_2$: < 0.5 ppm; H_2S: < 0.1 ppm; RH < 60% non-condensing
EN 60068-2-14, Test Nb, temperature change	5 °C to 60 °C, 3 K /minute
EN 60068-2-27 Mechanical shock	15 g, 11 ms pulse, 6 shocks in each of 3 axis
EN 60068-2-6 Sinusoidal vibration	DIN rail mount: 3.5 mm from 5-8.4 Hz, 1G from 8.4 - 150 Hz Panel mount: 7.0 mm from 5-8.5 Hz, 2G from 8.5 - 150 Hz

Table A-6 High potential isolation test

High potential isolation test	
24 V DC/ 5 V DC nominal circuits	707 V DC (type test of optical isolation boundaries)
230 V AC circuits to ground and 24 V DC / 5 V DC circuits	3000 V AC or 4250 V DC
Ethernet port to 24 V DC / 5 V DC circuits and ground[1]	1500 V AC (type test only)

[1] Ethernet port isolation is designed to limit hazard during short term network faults to hazardous voltages. It does not conform to safety requirements for routine AC line voltage isolation.

Insulation

The insulation is designed in accordance with the requirements of EN 61131-2.

Note

For modules with 24 V DC supply voltage, the electrical isolation is designed for max. 60 V AC / 75 V DC and basic insulation is designed according to EN 61131-2.

Contamination level/overvoltage category according to IEC 61131-2

- Pollution degree 2
- Overvoltage category: II

Protection class in accordance with IEC 61131-2

- Protection Class II according to EN 61131-2 (Protective conductor not required)

Degree of protection IP20

- IP20 Mechanical Protection, EN 60529
- Protects against finger contact with high voltage as tested by standard probe. External protection required for dust, dirt, water and foreign objects of < 12.5mm in diameter.

Rated voltages

Table A-7 Rated voltages

Module	Rated voltage	Tolerance
For DC/DC/DC modules	24 V DC	20.4 V DC to 28.8 V DC
For AC/DC/Relay modules	120/240 V AC	85 V AC to 264 V AC, 47 to 63 Hz
	110 V DC	77 V DC to 138 V DC

Note

When a mechanical contact turns on output power to the S7-200 SMART CPU, or any digital expansion module, it sends a "1" signal to the digital outputs for approximately 150 microseconds. This could cause unexpected machine or process operation which could result in death or serious injury to personnel and/or damage to equipment. You must plan for this, especially if you are using devices which respond to short duration pulses.

WARNING

Time duration for a mechanical contact to turn on output power

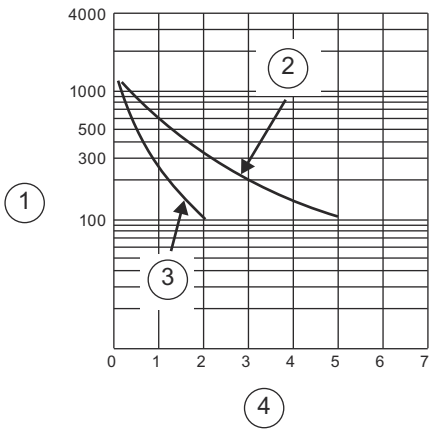
When a mechanical contact turns on output power to the S7-200 SMART CPU, or any digital expansion module, it sends a "1" signal to the digital outputs for approximately 150 microseconds.

This can cause unexpected machine or process operation which can result in death or serious injury to personnel and/or damage to equipment.

You must plan for this situation, especially, if you use devices which respond to short duration pulses.

Relay electrical service life

The typical performance data supplied by relay vendors is shown below. Actual performance may vary depending upon your specific application. An external protection circuit that is adapted to the load will enhance the service life of the contacts.



① Service life (x 10^3 operations)
② 250 V AC resistive load
30 V DC resistive load
③ 250 V AC inductive load (p.f=0.4)
30 V DC inductive load (L/R=7ms)
④ Rated Operating Current (A)

A.2 S7-200 SMART CPUs

A.2.1 CPU ST40 and CPU SR40

A.2.1.1 General specifications and features

General specifications and features of the CPU ST40 and CPU SR40

Table A-8 General specifications

Technical data	CPU ST40 DC/DC/DC	CPU SR40 AC/DC/Relay
Article number	6ES7288-1ST40-0AA2	6ES7288-1SR40-0AA2
Dimensions W x H x D (mm)	125 x 100 x 81	125 x 100 x 81
Weight	422.8 grams	493.2 grams
Power dissipation	24 W	28 W
Current available (EM bus)	1800 mA max. (5 V DC)	1800 mA max. (5 V DC)
Current available (24 V DC)	300 mA max. (sensor power)	300 mA max. (sensor power)
Digital input current consumption (24 V DC)	4 mA/input used	4 mA/input used

Table A-9 CPU features

Technical data		CPU ST40 DC/DC/DC	CPU SR40 AC/DC/Relay
User memory	Program	80 Kbytes	
	User-defined area	896 Kbytes	
	User data (V)	80 Kbytes	
	Retentive	40 Kbytes max.[1]	
On-board digital I/O		24 inputs/16 outputs	
Process image		584 bits of inputs (I) / 584 bits of outputs (Q)	
Analog image		266 words of inputs (AI) / 266 words of outputs (AQ)	
Bit memory (M)		512 bits	
Temporary (local) memory (L)		1024 bytes in the main program and 1024 bytes in each subroutine and interrupt routine 1008 bytes when programming in LAD or FBD (STEP 7-Micro/WIN reserves 4 bytes)	
Sequential control relays (S)		256 bits	
Expansion modules expansion		8 max.	
Signal board expansion		2 max.	
High-speed counters	Total	8	
	Single phase	8 at 200 kHz	
	A/B phase	8 at 200 kHz	

Technical data	CPU ST40 DC/DC/DC	CPU SR40 AC/DC/Relay
Pulse outputs [2]	5 at 200 kHz	No
Pulse catch inputs	16	
Cyclic interrupts	8 at 1 ms resolution	
Edge interrupts	4 rising and 4 falling (8 each when using optional signal boards)	
Memory card	microSD Card (maximum 128 GB)	
Real time clock accuracy	120 seconds/month	
Real time clock retention time	7 days typ./6 days min. at 25 °C	

[1] You can configure areas of V memory, M memory, C memory (current values), portions of T memory (current values) user-defined area on retentive, up to the specified maximum amount.

[2] The specified maximum pulse frequency is possible only for CPU models with transistor outputs. Pulse output operation is not recommended for CPU models with relay outputs.

Table A-10 PROFINET features

Description	CPU ST40 DC/DC/DC	CPU SR40 AC/DC/Relay
Support Controller function for PROFINET communication	Yes	
Support I-Device function for PROFINET communication	Yes	
Maximum number of PROFINET device	8	
Device number of PROFINET device	1 to 8	
Maximum input size of each PROFINET device	128 Bytes	
Maximum output size of each PROFINET device	128 Bytes	
Maximum input size of PROFINET I-Device	128 Bytes	
Maximum output size of PROFINET I-Device	128 Bytes	
Maximum number of modules	64	
Minimum cyclic update time of PROFINET device	The minimum value of the update time also depends on the communication component set for PROFINET, on the number of PROFINET devices, and the quantity of configured user data.	
CPU address range of PROFINET process image input register	I128.0 to I1279.7	
CPU address range of PROFINET process image output register	Q128.0 to Q1279.7	
CPU address of PROFINET process image input and output register for device #1	I128.0 to I255.7 Q128.0 to Q255.7	
CPU address of PROFINET process image input and output register for device #2	I256.0 to I383.7 Q256.0 to Q383.7	
CPU address of PROFINET process image input and output register for device #3	I384.0 to I511.7 Q384.0 to Q511.7	
CPU address of PROFINET process image input and output register for device #4	I512.0 to I639.7 Q512.0 to Q639.7	
CPU address of PROFINET process image input and output register for device #5	I640.0 to I767.7 Q640.0 to Q767.7	

Description	CPU ST40 DC/DC/DC	CPU SR40 AC/DC/Relay
CPU address of PROFINET process image input and output register for device #6	I768.0 to I895.7 Q768.0 to Q895.7	
CPU address of PROFINET process image input and output register for device #7	I896.0 to I1023.7 Q896.0 to Q1023.7	
CPU address of PROFINET process image input and output register for device #8	I1024.0 to I1151.7 Q1024.0 to Q1151.7	
CPU address of PROFINET process image input and output register for I-Device	I1152.0 to I1279.7 Q1152.0 to Q1279.7	

Table A-11 Performance

Type of instruction	Execution speed
Boolean	90 ns instruction
Move Word	0.7 µs/instruction
Real math	2.2 µs/instruction

Table A-12 User program elements supported

Element		Description
POUs	Type/quantity	Main program: 1 Subroutines: 128 (0 to 127) Function blocks: 128 (0 to 127) Interrupt routines: 128 (0 to 127)
	Nesting depth	From main program: 8 subroutine levels / function block levels From interrupt routine: 4 subroutine levels / function block levels Note: Function block nested function block instances follow the UDT instance nesting logic (maximum 4)
Accumulators	Quantity	4
Timers	Type/quantity	Non-retentive (TON, TOF): 320 Retentive (TONR): 192
Counters	Quantity	512

Table A-13 Communication

Technical data	CPU ST40 DC/DC/DC	CPU SR40 AC/DC/Relay
Number of ports	PROFINET (LAN): 2 Serial ports: 1 (RS485) Add-on serial ports: 2 (with optional RS232/485 signal board)	
HMI device	PROFINET (LAN): 8 connections Serial ports: 4 connections per port	
Programming device (PG)	PROFINET (LAN): 1 connection Serial ports: 1 connection	
CPUs (PUT/GET)	PROFINET (LAN): 8 client and 8 server connections	
Open user communication	PROFINET (LAN): 8 active and 8 passive connections	

Technical data	CPU ST40 DC/DC/DC	CPU SR40 AC/DC/Relay
Data rates	PROFINET (LAN): 100 Mb/s RS485 system protocols: 9600, 19200, and 187500 b/s RS485 freeport: 1200 to 115200 b/s	
Isolation (external signal to PLC logic)	PROFINET (LAN): Transformer isolated, 1500 V AC RS485: RS485 signal to chassis ground, 500 V AC/ 707 V DC; RS485 signal to CPU logic common, 500 V AC/ 707V DC;	
Cable type	PROFINET (LAN): CAT5e shielded RS485: PROFIBUS network cable	
PROFINET Communication		
PROFINET controller	Yes	
PROFINET device	Yes	
PROFINET controller		
Services		
PG/OP communication	Yes	
S7 routing	Yes	
Isochronous mode	No	
Open IE communication	Yes	
IRT	No	
MRP	No	
PROFIenergy	No	
Max. number of PROFINET devices that you can connect for RT	8	
Max. number of module	64	
Update times	The minimum value of the update time also depends on the communication component set for PROFINET, on the number of PROFINET devices, and the quantity of configured user data.	
With RT		
Send clock of 1 ms	1 ms to 512 ms	

Table A-14 Power supply

Technical data		CPU ST40 DC/DC/DC	CPU SR40 AC/DC/Relay	
Voltage range		20.4 to 28.8 V DC	85 to 264 V AC	77 to 138 V DC
Line frequency		--	47 to 63 Hz	--
Input current (max. load)	CPU only	320 mA at 24 V DC (without driving 300 mA sensor power) 620 mA at 24 V DC (with driving 300 mA sensor power)	150 mA at 120 V AC (without driving 300 mA sensor power) 270 mA at 120 V (with driving 300 mA sensor power) 100 mA at 240 V AC (without driving 300 mA sensor power) 170 mA at 240 V AC (with driving 300 mA sensor power)	155 mA at 110 VDC (with 300 mA power sensor output) 80 mA at 110 VDC (without 300 mA power sensor output)
	CPU with all expansion accessories	1 A at 24 V DC	420 mA at 120 V AC 260 mA at 240 V AC	240 mA at 110 VDC (with 300 mA power sensor output)

Technical data	CPU ST40 DC/DC/DC	CPU SR40 AC/DC/Relay	
Hold up time (loss of power)	10 ms at 24 V DC	30 ms at 120 V AC 200 ms at 240 V AC	30 ms at 110 V DC
Inrush current (max.)	14.9 A at 28.8 V DC	34 A at 264 V AC	
Isolation (input power to logic)	--	3000 V AC	
Ground leakage, AC line to functional earth	--	1.6 mA	
Internal fuse, not user replaceable	3 A, 250 V, slow blow	3 A, 250 V, slow blow	

Table A-15 Sensor power

Technical data	CPU ST40 DC/DC/DC	CPU SR40 AC/DC/Relay
Voltage range	19.2 to 28.8 V DC	20.4 to 28.8 V DC
Output current rating (max.)	300 mA	300 mA
Maximum ripple noise (<10 MHz)	< 1 V peak to peak	< 1 V peak to peak
Isolation (CPU logic to sensor power)	Not isolated	Not isolated

A.2.1.2 Digital inputs and outputs

Table A-16 Digital inputs

Technical data	CPU ST40 DC/DC/DC	CPU SR40 AC/DC/Relay
Number of inputs	24	
Type	Sink/Source (IEC Type 1 sink, except I0.0 to I0.7)	Sink/Source (IEC Type 1 sink)
Rated voltage	24 V DC at 4 mA, nominal	
Continuous permissible voltage	30 V DC, max.	
Surge voltage	35 V DC for 0.5 sec.	
Logic 1 signal (min.)	I0.0 to I0.7: 4 V DC at 8 mA Other inputs: 15 V DC at 2.5 mA	15 V DC at 2.5 mA
Logic 0 signal (max.)	I0.0 to I0.7: 1 V DC at 1 mA Other inputs: 5 V DC at 1 mA	5 V DC at 1 mA
Isolation (field side to logic)	500 V AC for 1 minute	
Isolation group	1	
Filter times	Individually selectable on each channel (points I0.0 to I1.7): µs: 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 12.8 ms: 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 12.8	
	Individually selectable on each channel (points I2.0 and greater): ms: 0, 6.4, 12.8	
HSC clock input rates (max.) (Logic 1 Level = 15 to 26 V DC)	Single phase: 8 HSCs at 200 kHz; A/B phase: 8 HSCs at 200 kHz;	
Number of inputs on simultaneously	24	

Technical data	CPU ST40 DC/DC/DC	CPU SR40 AC/DC/Relay
Cable length (max.), in meters	I0.0 to I0.7 and I1.0 to I1.7: Shielded (only): • 500 m normal (low-speed) inputs, • 50 m HSC (high-speed) inputs	All inputs: Shielded: • 500 m normal inputs • 50 m HSC inputs Unshielded: • 300 m normal inputs
	All other inputs: Shielded: • 500 m normal inputs Unshielded: • 300 m normal inputs	

Note

Improve high frequency square wave signal quality when frequency is above 20kHz

To achieve a clean square wave above 20 kHz, consider the following improvements:

- Minimize cable length
- Switch from a sink or source only driver to a push-pull driver
- Upgrade to a higher quality cable
- Consider adding an external load at the input

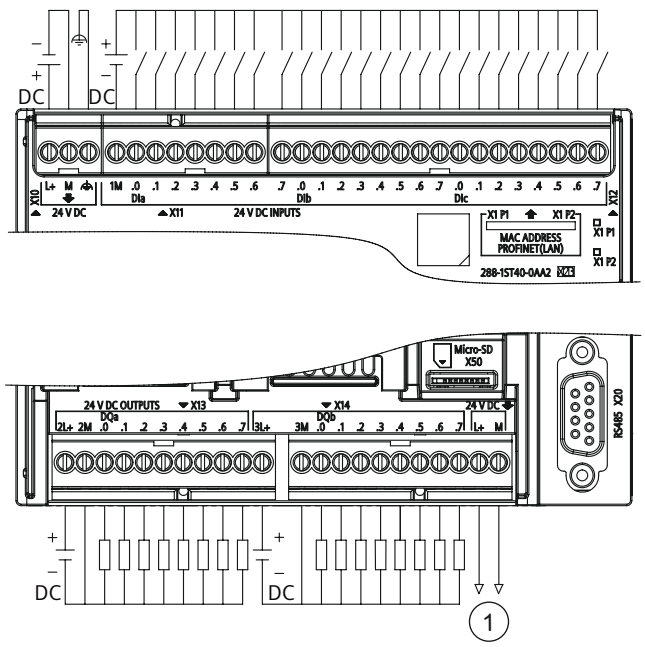
Table A-17 Digital outputs

Technical data	CPU ST40 DC/DC/DC	CPU SR40 AC/DC/Relay
Number of outputs	16	16
Type	Solid state - MOSFET (sourcing)	Relay, dry contact
Voltage range	20.4 to 28.8 V DC	5 to 30 V DC or 5 to 250 V AC
Logic 1 signal at max. current	20 V DC min.	--
Logic 0 signal with 10 KΩ load	0.1 V DC max.	--
Rated current per point (max.)	0.5 A	2 A
Rated current per common (max.)	6 A	10 A
Lamp load	5 W	30 W DC / 200 W AC
ON state resistance	0.6 Ω max.	0.2 Ω max. when new
Leakage current per point	10 µA max.	--
Surge current	8 A for 100 ms max.	7 A with contacts closed
Overload protection	No	No
Isolation (field side to logic)	500 V AC for 1 minute	3000 V AC for 1 minute (coil to contact) None (coil to logic)
Isolation resistance	--	100 MΩ min. when new
Isolation between open contact	--	750 V AC for 1 minute
Isolation groups	3	4
Inductive clamp voltage	L+ minus 48 V DC, 1 W dissipation	--
Switching delay (Qa.0 to Qa.7)	0.5 µs max., off to on 4.0 µs max., on to off	10 ms max.

Technical data	CPU ST40 DC/DC/DC	CPU SR40 AC/DC/Relay
Switching delay (Qb.0 to Qc.7)	50 µs max., off to on 200 µs max., on to off	10 ms max.
Maximum relay switching frequency	--	1 Hz
Lifetime mechanical (no load)	--	10,000,000 open/close cycles
Lifetime contacts at rated load	--	100,000 open/close cycles
Output state in STOP mode	Last value or substitute value (default value 0)	Last value or substitute value (default value 0)
Number of outputs on simultaneously	24	16
Cable length (max.), in meters	Shielded: 500 m Unshielded: 150 m	Shielded: 500 m Unshielded: 150 m

A.2.1.3 Wiring diagrams

Table A-18 Wiring diagram for the CPU ST40 DC/DC/DC (6ES7288-1ST40-0AA2)



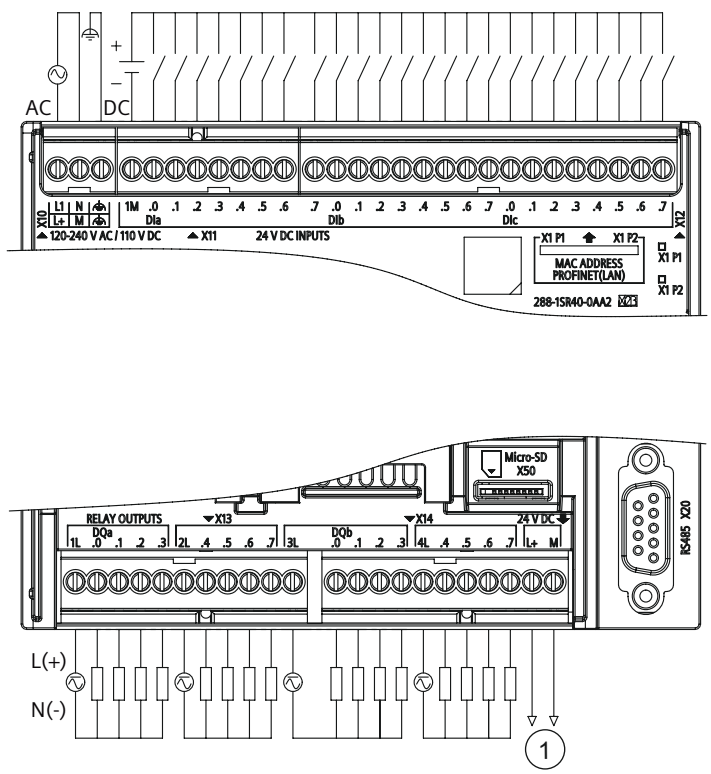
1 24 V DC Sensor Power Out

Table A-19 Connector pin locations for CPU ST40 DC/DC/DC (6ES7288-1ST40-0AA2)

Pin	X10	X11	X12	X13	X14
1	L+ / 24 V DC	1M	DI a.7	2L+	3M
2	M / 24 V DC	DI a.0	DI b.0	2M	DQ b.0
3	Functional Earth	DI a.1	DI b.1	DQ a.0	DQ b.1
4	--	DI a.2	DI b.2	DQ a.1	DQ b.2

Pin	X10	X11	X12	X13	X14
5	--	DI a.3	DI b.3	DQ a.2	DQ b.3
6	--	DI a.4	DI b.4	DQ a.3	DQ b.4
7	--	DI a.5	DI b.5	DQ a.4	DQ b.5
8	--	DI a.6	DI b.6	DQ a.5	DQ b.6
9	--	--	DI b.7	DQ a.6	DQ b.7
10	--	--	DI c.0	DQ a.7	L+ / 24 V DC Out
11	--	--	DI c.1	3L+	M / 24 V DC Out
12	--	--	DI c.2	--	--
13	--	--	DI c.3	--	--
14	--	--	DI c.4	--	--
15	--	--	DI c.5	--	--
16	--	--	DI c.6	--	--
17	--	--	DI c.7	--	--
18	--	--	--	--	--
19	--	--	--	--	--
20	--	--	--	--	--

Table A-20 Wiring diagram for the CPU SR40 AC/DC/Relay (6ES7288-1SR40-0AA2)



1 24 V DC Sensor Power Out

Table A-21 Connector pin locations for SR40 AC/DC/Relay (6ES7288-1SR40-0AA2)

Pin	X10	X11	X12	X13	X14
1	L1 / 120 - 240 V AC L+/ 110 V DC	1M	DI a.7	1L	DQ b.0
2	N / 120 - 240 V AC M/ 110 V DC	DI a.0	DI b.0	DQ a.0	DQ b.1
3	Functional Earth	DI a.1	DI b.1	DQ a.1	DQ b.2
4	--	DI a.2	DI b.2	DQ a.2	DQ b.3
5	--	DI a.3	DI b.3	DQ a.3	4L
6	--	DI a.4	DI b.4	2L	DQ b.4
7	--	DI a.5	DI b.5	DQ a.4	DQ b.5
8	--	DI a.6	DI b.6	DQ a.5	DQ b.6
9	--	--	DI b.7	DQ a.6	DQ b.7
10	--	--	DI c.0	DQ a.7	L+ / 24 V DC Out
11	--	--	DI c.1	3L	M / 24 V DC Out
12	--	--	DI c.2	--	--
13	--	--	DI c.3	--	--
14	--	--	DI c.4	--	--
15	--	--	DI c.5	--	--
16	--	--	DI c.6	--	--
17	--	--	DI c.7	--	--
18	--	--	--	--	--
19	--	--	--	--	--
20	--	--	--	--	--

A.2.2 CPU ST60 and CPU SR60

A.2.2.1 General specifications and features

Table A-22 General specifications

Technical data	CPU ST60 DC/DC/DC	CPU SR60 AC/DC/Relay
Article number	6ES7288-1ST60-0AA2	6ES7288-1SR60-0AA2
Dimensions W x H x D (mm)	175 x 100 x 81	175 x 100 x 81
Weight	546.2 grams	629.6 grams
Power dissipation	25 W	30 W
Current available (EM bus)	1800 mA max. (5 V DC)	1800 mA max. (5 V DC)

Technical data	CPU ST60 DC/DC/DC	CPU SR60 AC/DC/Relay
Current available (24 V DC)	300 mA max. (sensor power)	300 mA max. (sensor power)
Digital input current consumption (24 V DC)	4 mA/input used	4 mA/input used

Table A-23 CPU features

Technical data		CPU ST60 DC/DC/DC	CPU SR60 AC/DC/Relay
User memory	Program	100 Kbytes	
	User-defined area	896 Kbytes	
	User data (V)	100 Kbytes	
	Retentive	40 Kbytes max.[1]	
On-board digital I/O		36 inputs/24 outputs	
Process image		584 bits of inputs (I) / 584 bits of outputs (Q)	
Analog image		266 words of inputs (AI) / 266 words of outputs (AQ)	
Bit memory (M)		512 bits	
Temporary (local) memory (L)		1024 bytes in the main program and 1024 bytes in each subroutine and interrupt routine 1008 bytes when programming in LAD or FBD (STEP 7-Micro/WIN reserves 4 bytes)	
Sequential control relays (S)		256 bits	
Expansion modules expansion		8 max.	
Signal board expansion		2 max.	
High-speed counters	Total	8	
	Single phase	8 at 200 kHz	
	A/B phase	8 at 200 kHz	
Pulse outputs [2]		5 at 200 kHz	No
Pulse catch inputs		16	16
Cyclic interrupts		8 at 1 ms resolution	
Edge interrupts		4 rising and 4 falling (8 each when using optional signal boards)	
Memory card		microSD Card (maximum 128 GB)	
Real time clock accuracy		120 seconds/month	
Real time clock retention time		7 days typ./6 days min. at 25 °C	

1 You can configure areas of V memory, M memory, C memory (current values), portions of T memory (current values on retentive timers) and user-defined area to be retentive, up to the specified maximum amount.

2 The specified maximum pulse frequency is possible only for CPU models with transistor outputs. Pulse output operation is not recommended for CPU models with relay outputs.

Table A-24 PROFINET features

Description	CPU ST60 DC/DC/DC	CPU SR60 AC/DC/Relay
Support Controller function for PROFINET communication	Yes	
Support I-Device function for PROFINET communication	Yes	

Description	CPU ST60 DC/DC/DC	CPU SR60 AC/DC/Relay
Maximum number of PROFINET device	8	
Device number of PROFINET device	1 to 8	
Maximum input size of each PROFINET device	128 Bytes	
Maximum output size of each PROFINET device	128 Bytes	
Maximum input size of PROFINET I-Device	128 Bytes	
Maximum output size of PROFINET I-Device	128 Bytes	
Maximum number of modules	64	
Minimum cyclic update time of PROFINET device	The minimum value of the update time also depends on the communication component set for PROFINET, on the number of PROFINET devices, and the quantity of configured user data.	
CPU address range of PROFINET process image input register	I128.0 to I1279.7	
CPU address range of PROFINET process image output register	Q128.0 to Q1279.7	
CPU address of PROFINET process image input and output register for device #1	I128.0 to I255.7 Q128.0 to Q255.7	
CPU address of PROFINET process image input and output register for device #2	I256.0 to I383.7 Q256.0 to Q383.7	
CPU address of PROFINET process image input and output register for device #3	I384.0 to I511.7 Q384.0 to Q511.7	
CPU address of PROFINET process image input and output register for device #4	I512.0 to I639.7 Q512.0 to Q639.7	
CPU address of PROFINET process image input and output register for device #5	I640.0 to I767.7 Q640.0 to Q767.7	
CPU address of PROFINET process image input and output register for device #6	I768.0 to I895.7 Q768.0 to Q895.7	
CPU address of PROFINET process image input and output register for device #7	I896.0 to I1023.7 Q896.0 to Q1023.7	
CPU address of PROFINET process image input and output register for device #8	I1024.0 to I1151.7 Q1024.0 to Q1151.7	
CPU address of PROFINET process image input and output register for I-Device	I1152.0 to I1279.7 Q1152.0 to Q1279.7	

Table A-25 Performance

Type of instruction	Execution speed
Boolean	90 ns instruction
Move Word	0.7 µs/instruction
Real math	2.2 µs/instruction

Table A-26 User program elements supported

Element		Description
POUs	Type/quantity	Main program: 1 Subroutines: 128 (0 to 127) Function blocks: 128 (0 to 127) Interrupt routines: 128 (0 to 127)
	Nesting depth	From main program: 8 subroutine levels / function block levels From interrupt routine: 4 subroutine levels / function block levels Note: Function block nested function block instances follow the UDT instance nesting logic (maximum 4)
Accumulators	Quantity	4
Timers	Type/quantity	Non-retentive (TON, TOF): 320 Retentive (TONR): 192
Counters	Quantity	512

Table A-27 Communication

Technical data	CPU ST60 DC/DC/DC	CPU SR60 AC/DC/Relay
Number of ports	PROFINET (LAN): 2 Serial ports: 1 (RS485) Add-on serial ports: 2 (with optional RS232/485 signal board)	
HMI device	PROFINET (LAN): 8 connections Serial ports: 4 connections per port	
Programming device (PG)	PROFINET (LAN): 1 connection Serial ports: 1 connection	
CPUs (PUT/GET)	PROFINET (LAN): 8 client and 8 server connections	
Open user communication	PROFINET (LAN): 8 active and 8 passive connections	
Data rates	PROFINET (LAN): 100 Mb/s RS485 system protocols: 9600, 19200, and 187500 b/s RS485 freeport: 1200 to 115200 b/s	
Isolation (external signal to PLC logic)	PROFINET (LAN): Transformer isolated, 1500 V AC RS485: RS485 signal to chassis ground, 500 V AC/ 707 V DC; RS485 signal to CPU logic common, 500 V AC/ 707V DC;	
Cable type	PROFINET (LAN): CAT5e shielded RS485: PROFIBUS network cable	
PROFINET Communication		
PROFINET controller	Yes	
PROFINET device	Yes	
PROFINET controller		
Services		
PG/OP communication	Yes	
S7 routing	Yes	
Isochronous mode	No	
Open IE communication	Yes	

Technical data	CPU ST60 DC/DC/DC	CPU SR60 AC/DC/Relay
IRT	No	
MRP	No	
PROFIenergy	No	
Max. number of PROFINET devices that you can connect for RT	8	
Max. number of module	64	
Update times	The minimum value of the update time also depends on the communication component set for PROFINET, on the number of PROFINET devices, and the quantity of configured user data.	
With RT		
Send clock of 1 ms	1 ms to 512 ms	

Table A-28 Power supply

Technical data		CPU ST60 DC/DC/DC	CPU SR60 AC/DC/Relay	
Voltage range		20.4 to 28.8 V DC	85 to 264 V AC	77 to 138 V DC
Line frequency		--	47 to 63 Hz	--
Input current (max. load)	CPU only	350 mA at 24 V DC (without driving 300 mA sensor power) 650 mA at 24 V DC (with driving 300 mA sensor power)	190 mA at 120 V AC (without driving 300 mA sensor power) 310 mA at 120 V AC (with driving 300 mA sensor power) 120 mA at 240 V AC (without driving 300 mA sensor power) 190 mA at 240 V AC (with driving 300 mA sensor power)	110 mA at 110 VDC (with 300 mA power sensor output) 180 mA at 110 VDC (without 300 mA power sensor output)
	CPU with all expansion accessories	1.1 A at 24 V DC	460 mA at 120 V AC 280 mA at 240 V AC	270 mA at 110 VDC (with 300 mA power sensor output)
Hold up time (loss of power)		10 ms at 24 V DC	30 ms at 120 V AC 200 ms at 240 V AC	30 ms at 110 V DC
Inrush current (max.)		15 A at 28.8 V DC	34 A at 264 V DC	
Isolation (input power to logic)		None	3000 V AC	
Ground leakage, AC line to functional earth		--	1.6 mA	
Internal fuse, not user replaceable		3 A, 250 V, slow blow	3 A, 250 V, slow blow	

Table A-29 Sensor power

Technical data	CPU ST60 DC/DC/DC	CPU SR60 AC/DC/Relay
Voltage range	19.2 to 28.8 V DC	20.4 to 28.8 V DC
Output current rating (max.)	300 mA	300 mA
Maximum ripple noise (<10 MHz)	< 1 V peak to peak	< 1 V peak to peak
Isolation (CPU logic to sensor power)	Not isolated	Not isolated

A.2.2.2 Digital inputs and outputs

Table A-30 Digital inputs

Technical data	CPU ST60 DC/DC/DC	CPU SR60 AC/DC/Relay
Number of inputs	36	
Type	Sink/Source (IEC Type 1 sink, except I0.0 to I0.7)	Sink/Source (IEC Type 1 sink)
Rated voltage	24 V DC at 4 mA, nominal	
Continuous permissible voltage	30 V DC, max.	30 V DC, max.
Surge voltage	35 V DC for 0.5 sec.	35 V DC for 0.5 sec.
Logic 1 signal (min.)	• I0.0 to I0.7: 4 V DC at 8 mA • Other inputs: 15 V DC at 2.5 mA	15 V DC at 2.5 mA
Logic 0 signal (max.)	• I0.0 to I0.7: 1 V DC at 1 mA • Other inputs: 5 V DC at 1 mA	5 V DC at 1 mA
Isolation (field side to logic)	500 V AC for 1 minute	500 V AC for 1 minute
Isolation groups	1	
Filter times	Individually selectable on each channel (points I0.0 to I1.7): µs: 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 12.8 ms: 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 12.8	
	Individually selectable on each channel (points I2.0 and greater): ms: 0, 6.4, 12.8	
HSC clock input rates	Single phase: 8 HSCs at 200 kHz; A/B phase: 8 HSCs at 200 kHz;	
Number of inputs on simultaneously	36	
Cable length (max.), in meters	I0.0 to I0.7 and I1.0 to I1.7: Shielded (only): • 500 m normal (low-speed) inputs • 50 m HSC (high-speed) inputs	All inputs: Shielded: • 500 m normal inputs • 50 m HSC inputs Unshielded: • 300 m normal inputs
	All other inputs: Shielded: • 500 m normal inputs Unshielded: • 300 m normal inputs	

Note

Improve high frequency square wave signal quality when frequency is above 20kHz

To achieve a clean square wave above 20 kHz, consider the following improvements:

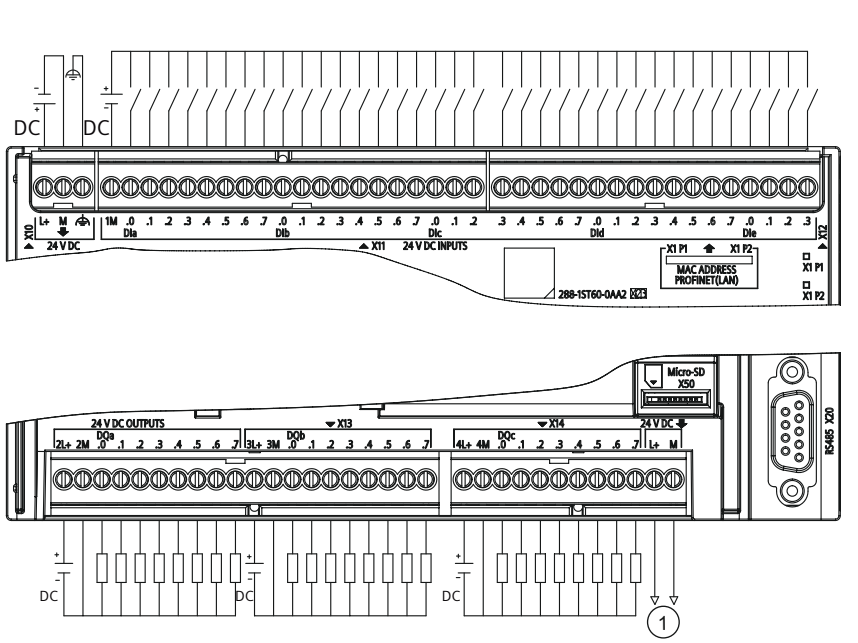
- Minimize cable length
- Switch from a sink or source only driver to a push-pull driver
- Upgrade to a higher quality cable
- Consider adding an external load at the input

Table A-31 Digital outputs

Technical data	CPU ST60 DC/DC/DC	CPU SR60 AC/DC/Relay
Number of outputs	24	24
Type	Solid state - MOSFET (sourcing)	Relay, dry contact
Voltage range	20.4 to 28.8 V DC	5 to 30 V DC or 5 to 250 V AC
Logic 1 signal at max. current	20 V DC min.	--
Logic 0 signal with 10 KΩ load	0.1 V DC max.	--
Rated current per point (max.)	0.5 A	2 A
Rated current per common (max.)	6 A	10 A
Lamp load	5 W	30 W DC / 200 W AC
ON state resistance	0.6 Ω max.	0.2 Ω max. when new
Leakage current per point	10 μA max.	--
Surge current	8 A for 100 ms max.	7 A with contacts closed
Overload protection	No	No
Isolation (field side to logic)	500 V AC for 1 minute	3000 V AC for 1 minute (coil to contact) None (coil to logic)
Isolation resistance	--	100 M Ω min. when new
Isolation between open contacts	--	750 V AC for 1 minute
Isolation groups	3	6
Inductive clamp voltage	L+ minus 48 V DC, 1 W dissipation	--
Switching delay (Qa.0 to Qa.7)	0.5 μs max., off to on 1 μs max., on to off	10 ms max.
Switching delay (Qb.0 to Qc.7)	50 μs max., off to on 200 μs max., on to off	10 ms max.
Maximum relay switching frequency	--	1 Hz
Lifetime mechanical (no load)	--	10,000,000 open/close cycles
Lifetime contacts at rated load	--	100,000 open/close cycles
Output behavior in STOP	Last value or substitute value (default value 0)	Last value or substitute value (default value 0)
Number of outputs on simultaneously	24	24
Cable length (max.), in meters	Shielded: 500 m Unshielded: 150 m	Shielded: 500 m Unshielded: 150 m

A.2.2.3 Wiring diagrams

Table A-32 Wiring diagram for the CPU ST60 DC/DC/DC (6ES7288-1ST60-0AA2)



1 24 V DC Sensor Power Output

Table A-33 Connector pin locations for CPU ST60 DC/DC/DC (6ES7288-1ST60-0AA2)

Pin	X10	X11	X12	X13	X14
1	L+ / 24 V DC	1M	DI c.3	2L+	4L+
2	M / 24 V DC	DI a.0	DI c.4	2M	4M
3	Functional Earth	DI a.1	DI c.5	DQ a.0	DQ c.0
4		DI a.2	DI c.6	DQ a.1	DQ c.1
5		DI a.3	DI c.7	DQ a.2	DQ c.2
6		DI a.4	DI d.0	DQ a.3	DQ c.3
7		DI a.5	DI d.1	DQ a.4	DQ c.4
8		DI a.6	DI d.2	DQ a.5	DQ c.5
9		DI a.7	DI d.3	DQ a.6	DQ c.6
10		DI b.0	DI d.4	DQ a.7	DQ c.7
11		DI b.1	DI d.5	3L+	L+ / 24 V DC Out
12		DI b.2	DI d.6	3M	M / 24 V DC Out
13		DI b.3	DI d.7	DQ b.0	--
14		DI b.4	DI e.0	DQ b.1	--
15		DI b.5	DI e.1	DQ b.2	--
16		DI b.6	DI e.2	DQ b.3	--
17		DI b.7	DI e.3	DQ b.4	--

Pin	X10	X11	X12	X13	X14
18		DI c.0		DQ b.5	--
19		DI c.1		DQ b.6	--
20		DI c.2		DQ b.7	--

Table A-34 Wiring diagram for the CPU SR60 AC/DC/Relay (6ES7288-1SR60-0AA2)

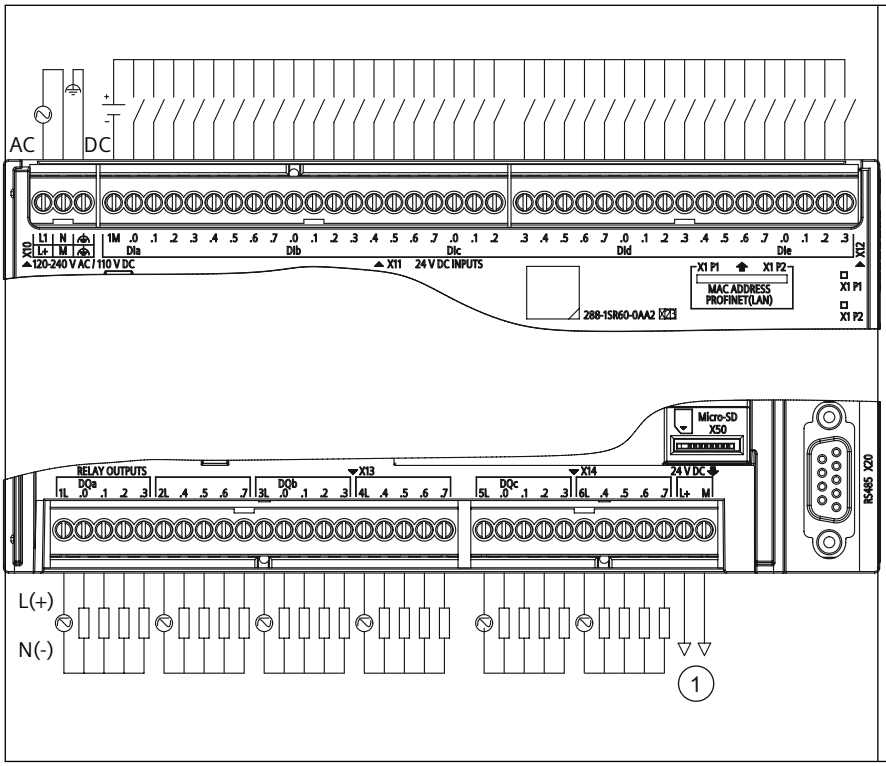


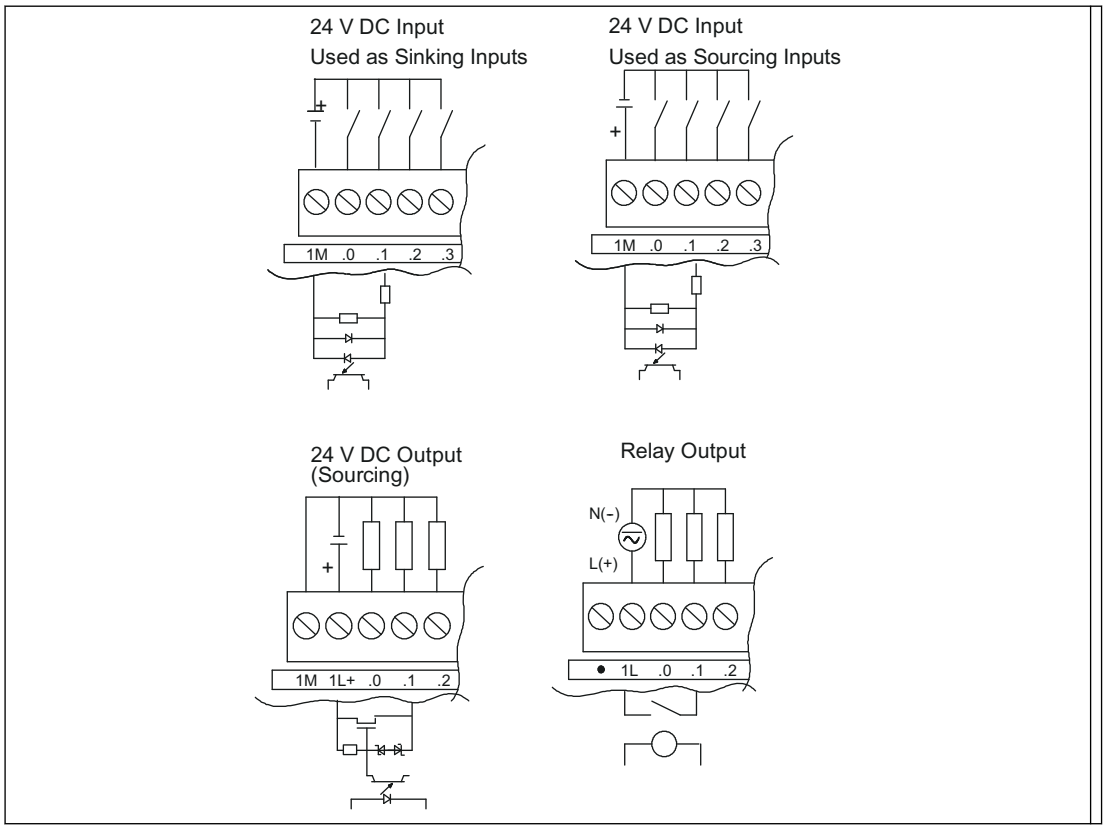
Table A-35 Connector pin locations for CPU SR60 AC/DC/Relay (6ES7288-1SR60-0AA2)

Pin	X10	X11	X12	X13	X14
1	L1 / 120 - 240 V AC L+/ 110 V DC	1M	DI c.3	1L	5L
2	N / 120 - 240 V AC M/ 110 V DC	DI a.0	DI c.4	DQ a.0	DQ c.0
3	Functional Earth	DI a.1	DI c.5	DQ a.1	DQ c.1
4		DI a.2	DI c.6	DQ a.2	DQ c.2
5		DI a.3	DI c.7	DQ a.3	DQ c.3
6		DI a.4	DI d.0	2L	6L
7		DI a.5	DI d.1	DQ a.4	DQ c.4
8		DI a.6	DI d.2	DQ a.5	DQ c.5
9		DI a.7	DI d.3	DQ a.6	DQ c.6

Pin	X10	X11	X12	X13	X14
10		DI b.0	DI d.4	DQ a.7	DQ c.7
11		DI b.1	DI d.5	3L	L+ / 24 V DC Out
12		DI b.2	DI d.6	DQ b.0	M / 24 V DC Out
13		DI b.3	DI d.7	DQ b.1	--
14		DI b.4	DI e.0	DQ b.2	--
15		DI b.5	DI e.1	DQ b.3	--
16		DI b.6	DI e.2	4L	--
17		DI b.7	DI e.3	DQ b.4	--
18		DI c.0		DQ b.5	--
19		DI c.1		DQ b.6	--
20		DI c.2		DQ b.7	--

A.2.3 Wiring diagrams for sink and source input, and relay output

Table A-36 Wiring diagrams for sink input, source input, and relay output



A.3 Digital inputs and outputs expansion modules (EMs)

A.3.1 EM DE08 and EM DE16 digital input specifications

Table A-37 General specifications

Model	EM Digital 8 x Inputs (EM DE08)	EM Digital 16 x Inputs (EM DE16)
Article number	6ES7288-2DE08-0AA0	6ES7288-2DE16-0AA0
Dimensions W x H x D (mm)	45 x 100 x 81	45 x 100 x 81
Weight	141.4 grams	176 grams
Power dissipation	1.5 W	2.3 W
Current consumption (SM Bus)	105 mA	105 mA
Current consumption (24 V DC)	4 mA / input used	4 mA / input used

Table A-38 Digital inputs

Model	EM Digital 8 x Inputs (EM DE08)	EM Digital 16 x Inputs (EM DE16)
Number of inputs	8	16
Type	Sink/Source (IEC Type 1 sink)	Sink/Source (IEC Type 1 sink)
Rated voltage	24 V DC at 4 mA, nominal	24 V DC at 4 mA, nominal
Continuous permissible voltage	30 V DC, max.	30 V DC, max.
Surge voltage	35 V DC for 0.5 sec.	35 V DC for 0.5 sec.
Logic 1 signal (min.)	15 V DC at 2.5 mA	15 V DC at 2.5 mA
Logic 0 signal (max.)	5 V DC at 1 mA	5 V DC at 1 mA
Isolation (field side to logic)	500 V AC for 1 minute	500 V AC for 1 minute
Isolation groups	2	4
Filter times	0.2, 0.4, 0.8, 1.6, 3.2, 6.4, and 12.8 ms (selectable in groups of 4)	0.2, 0.4, 0.8, 1.6, 3.2, 6.4, and 12.8 ms (selectable in groups of 4)
Number of inputs on simultaneously	8	16
Cable length (max.), in meters	Shielded: 500 m normal inputs Unshielded: 300 m normal inputs	Shielded: 500 m normal inputs Unshielded: 300 m normal inputs

Table A-39 Wiring diagram for the EM DE08 Digital 8 x Inputs (6ES7288-2DE08-0AA0) and EM DE16 Digital 16 x Input (6ES7288-2DE16-0AA0)

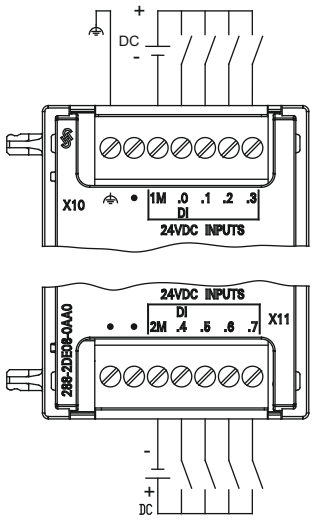
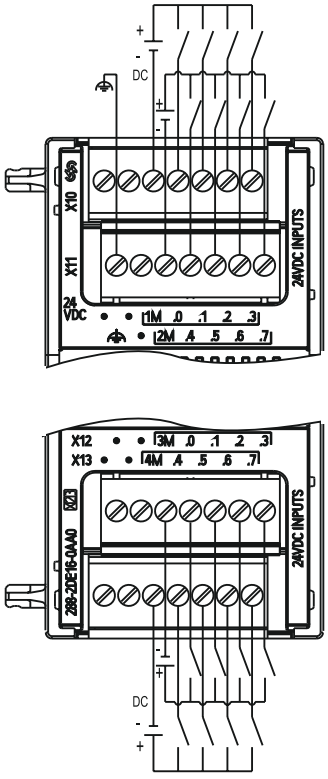
EM DE08 Digital 8 x Inputs (6ES7288-2DE08-0AA0)	**EM DE16 Digital 16 x Inputs (6ES7288-2DE16-0AA0)**
	

Table A-40 Connector pin locations for EM DE08 Digital 8 x Input (6ES7288-2DE08-0AA0)

Pin	X10	X11
1	Functional Earth	No connection
2	No connection	No connection
3	1M	2M
4	DI a.0	DI a.4
5	DI a.1	DI a.5
6	DI a.2	DI a.6
7	DI a.3	DI a.7

Table A-41 Connector pin locations for EM DE16 Digital 16 x Input (6ES7288-2DE16-0AA0)

Pin	X10	X11	X12	X13
1	No connection	Functional Earth	No connection	No connection
2	No connection	No connection	No connection	No connection
3	1M	2M	3M	4M
4	DI a.0	DI a.4	DI b.0	DI b.4
5	DI a.1	DI a.5	DI b.1	DI b.5
6	DI a.2	DI a.6	DI b.2	DI b.6
7	DI a.3	DI a.7	DI b.3	DI b.7

A.3.2 EM DT08, EM DR08, EM QR16, and EM QT16 digital output specifications

Table A-42 General specifications

Model	EM Digital 8 x Outputs (EM DT08)	EM Digital 8 x Outputs Relay (EM DR08)	EM Digital 16 x Outputs Relay (EM QR16)	EM Digital 16 x Outputs Transistor (EM QT16)
Article number	6ES7288-2DT08-0AA0	6ES7288-2DR08-0AA0	6ES7288-2QR16-0AA0	6ES7288-2QT16-0AA0
Dimensions W x H x D (mm)	45 x 100 x 81	45 x 100 x 81	45 x 100 x 81	45 x 100 x 81
Weight	147 grams	166.3 grams	221 grams	186 grams
Power dissipation	1.5 W	4.5 W	4.5 W	1.7 W
Current consumption (SM Bus)	120 mA	120 mA	110 mA, SM bus	120 mA, SM bus
Current consumption (24 V DC)	--	11 mA / relay coil used	150 mA, all relay are on	50 mA

Table A-43 Digital outputs

Model	EM Digital 8 x Outputs (EM DT08)	EM Digital 8 x Outputs Relay (EM DR08)	EM Digital 16 x Outputs Relay (EM QR16)	EM Digital 16 x Outputs Transistor (EM QT16)
Number of outputs	8	8	16	16
Type	Solid state - MOSFET (sourcing)	Relay, dry contact	Relay, dry contact	Solid state - MOSFET (sourcing)
Voltage range	20.4 to 28.8 V DC	5 to 30 V DC or 5 to 250 V AC	5 to 30 V DC or 5 to 250 V AC	20.4 to 28.8 V DC
Logic 1 signal at max. current	20 V DC	-	-	20 V DC
Logic 0 signal with 10 K Ω load	0.1 V DC	-	-	0.1 V DC
Rated current per point (max.)	0.75 A	2.0 A	2.0 A	0.75 A
Rated current per common (max.)	3 A	8 A	8 A	3 A
Lamp load	5 W DC	30 W DC / 200 W AC	30 W DC/200 W AC	5 W
ON state contact resistance	0.6 Ω	0.2 Ω max. when new	0.2 Ω max. when new	0.6 Ω max.

Model	EM Digital 8 x Outputs (EM DT08)	EM Digital 8 x Outputs Relay (EM DR08)	EM Digital 16 x Outputs Relay (EM QR16)	EM Digital 16 x Outputs Transistor (EM QT16)
Leakage current per point	10 µA	--	--	10 µA
Surge current	8 A for 100 ms max.	7 A with contacts closed	7 A with contacts closed	8 A for 100 ms max.
Overload protection	No	No	No	No
Isolation (field side to logic)	Optical, 500 V AC for 1 minute	1500 V AC for 1 minute (coil to contact) None (coil to logic)	1500 V AC for 1 minute (coil to contact) None (coil to logic)	500 V AC for 1 minute
Isolation resistance	-	100 M Ω min. when new	100 M Ω min. when new	-
Isolation between open contacts	-	750 V AC for 1 minute	750 V AC for 1 minute	-
Isolation groups	2	2	4	4
Inductive clamp voltage	Minus 48 V DC, 1 W dissipation	-	-	L+ minus 48 V, 1 W dissipation
Switching delay	Switch on less than 50 µs and switch off less than 200 µS	10 ms max.	10 ms max.	Switch on less than 50 µs and switch off less than 200 µS
Lifetime mechanical (no load)	--	10,000,000 open/close cycles	10,000,000 open/close cycles	-
Lifetime contacts at rated load	--	100,000 open/close cycles	100,000 open/close cycles	-
Output behavior in STOP	Last value or substitute value (default value 0)	Last value or substitute value (default value 0)	Last value or substitute value (default value 0)	Last value or substitute value (default value 0)
Number of outputs on simultaneously	8	8	16	16
Cable length (max.), in meters	Shielded: 500 m Unshielded: 150 m	Shielded: 500 m Unshielded: 150 m	Shielded: 500 m Unshielded: 150 m	Shielded: 500 m Unshielded: 150 m

Table A-44 Wiring diagrams for the EM DT08 Digital 8 x Outputs (6ES7288-2DT08-0AA0) and EM DR08 Digital 8 x Outputs x Relay (6ES7288-2DR08-0AA0)

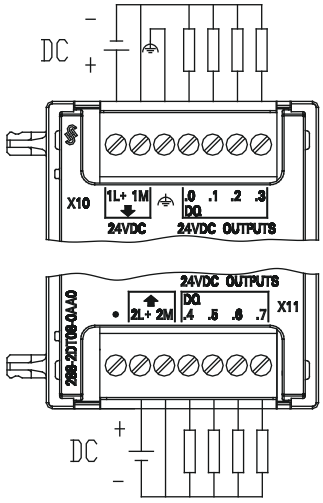
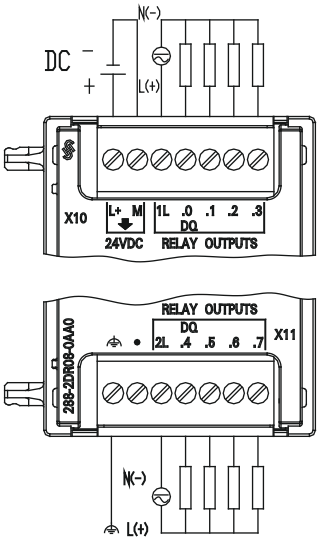
EM DT08 Digital 8 x Outputs (6ES7288-2DT08-0AA0)	**EM DR08 Digital 8 x Outputs x Relay (6ES7288-2DR08-0AA0)**
	

Table A-45 Connector pin locations for EM DT08 Digital 8 x Outputs (6ES7288-2DT08-0AA0

Pin	X10	X11
1	1L+ / 24 V DC	No connection
2	1M / 24 V DC	2L+ / 24 V DC
3	Functional Earth	2M / 24 V DC
4	DQ a.0	DQ a.4
5	DQ a.1	DQ a.5
6	DQ a.2	DQ a.6
7	DQ a.3	DQ a.7

Table A-46 Connector pin locations for EM DR08 Digital 8 x Outputs x Relay (6ES7288-2DR08-0AA0)

Pin	X10	X11
1	L+ / 24 V DC	Functional Earth
2	M / 24 V DC	No connection
3	1L	2L
4	DQ a.0	DQ a.4
5	DQ a.1	DQ a.5

Pin	X10	X11
6	DQ a.2	DQ a.6
7	DQ a.3	DQ a.7

Table A-47 Wiring diagrams for the EM QR16 Digital 16 x Outputs Relay (6ES7288-2QR16-0AA0) and EM QT16 Digital 16 x Outputs Transition (6ES7288-2QT16-0AA0)

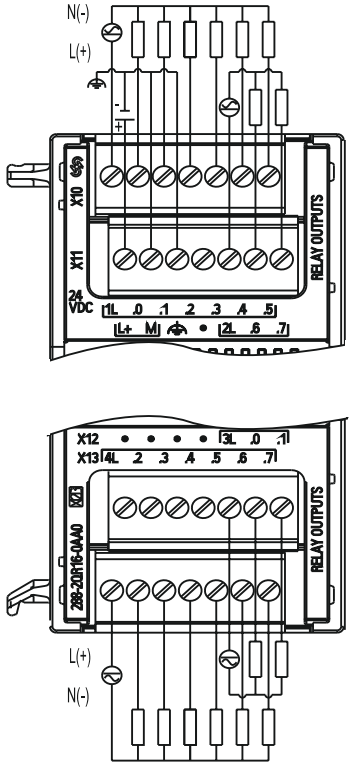
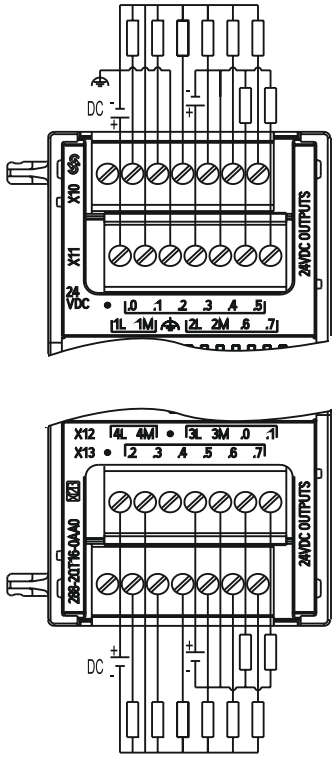
EM QR16 Digital 16 x Outputs Relay (6ES7288-2QR16-0AA0)	EM QT16 Digital 16 x Outputs Transition (6ES7288-2QT16-0AA0)
	

Table A-48 Connector pin locations for EM QR16 Digital 16 x Outputs Relay (6ES7288-2QR16-0AA0)

Pin	X10	X11	X12	X13
1	1L	L+ / 24 V DC	No connection	4L
2	DQ a.0	M / 24 V DC	No connection	DQ b.2
3	DQ a.1	Functional Earth	No connection	DQ b.3
4	DQ a.2	No connection	No connection	DQ b.4
5	DQ a.3	2L	3L	DQ b.5
6	DQ a.4	DQ a.6	DQ b.0	DQ b.6
7	DQ a.5	DQ a.7	DQ b.1	DQ b.7

Table A-49 Connector pin locations for EM QT16 Digital 16 x Outputs Transition (6ES7288-2QT16-0AA0)

Pin	X10	X11	X12	X13
1	No connection	1L / 24 V DC	4L / 24 V DC	No connection
2	DQ a.0	1M / 24 V DC	4M / 24 V DC	DQ b.2
3	DQ a.1	Functional Earth	No connection	DQ b.3
4	DQ a.2	2L / 24 V DC	3L / 24 V DC	DQ b.4
5	DQ a.3	2M / 24 V DC	3M / 24 V DC	DQ b.5
6	DQ a.4	DQ a.6	DQ b.0	DQ b.6
7	DQ a.5	DQ a.7	DQ b.1	DQ b.7

A.3.3 EM DT16, EM DR16, EM DT32, and EM DR32 digital input/output specifications

Table A-50 General specifications

Model	**EM Digital 8 x Inputs/ Digital 8 x Outputs (EM DT16)**	**EM Digital 8 x Inputs/ 8 x Relay Outputs (EM DR16)**	**EM Digital 16 x Inputs/ Digital 16 x Outputs (EM DT32)**	**EM Digital 16 x Inputs / 16 x Relay Outputs (EM DR32)**
Article number	6ES7288-2DT16-0AA0	6ES7288-2DR16-0AA0	6ES7288-2DT32-0AA0	6ES7288-2DR32-0AA0
Dimensions W x H x D (mm)	45 x 100 x 81	45 x 100 x 81	70 x 100 x 81	70 x 100 x 81
Weight	179.7g grams	201.9 grams	257.3 grams	295.4 grams
Power dissipation	2.5 W	5.5 W	4.5 W	10 W
Current consumption (SM Bus)	145 mA	145 mA	185 mA	180 mA
Current consumption (24 V DC)	4 mA / Input used 11 mA / Relay coil used	4 mA / Input used 11 mA / Relay coil used	4 mA / Input used	4 mA / Input used

Table A-51 Digital inputs

Model	**EM Digital 8 x Inputs/ Digital 8 x Outputs (EM DT16)**	**EM Digital 8 x Inputs/ 8 x Relay Outputs (EM DR16)**	**EM Digital 16 x Inputs/ Digital 16 x Outputs (EM DT32)**	**EM Digital 16 x Inputs / 16 x Relay Outputs (EM DR32)**
Number of inputs	8	8	16	16
Type	Sink/Source (IEC Type 1 sink)	Sink/Source (IEC Type 1 sink)	Sink/Source (IEC Type 1 sink)	Sink/Source (IEC Type 1 sink)
Rated voltage	24 V DC at 4 mA, nominal	24 V DC at 4 mA, nominal	24 V DC at 4 mA, nominal	24 V DC at 4 mA, nominal
Continuous permissible voltage	30 V DC max.	30 V DC max.	30 V DC max.	30 V DC max.
Surge voltage	35 V DC for 0.5 sec.	35 V DC for 0.5 sec.	35 V DC for 0.5 sec.	35 V DC for 0.5 sec.

Model	EM Digital 8 x Inputs/ Digital 8 x Outputs (EM DT16)	EM Digital 8 x Inputs/ 8 x Relay Outputs (EM DR16)	EM Digital 16 x Inputs/ Digital 16 x Outputs (EM DT32)	EM Digital 16 x Inputs / 16 x Relay Outputs (EM DR32)
Logic 1 signal (min.)	15 V DC	15 V DC	15 V DC	15 V DC
Logic 0 signal (max.)	5 V DC	5 V DC	5 V DC	5 V DC
Isolation (field side to logic)	500 V AC for 1 minute	500 V AC for 1 minute	500 V AC for 1 minute	500 V AC for 1 minute
Isolation groups	2	2	2	2
Filter times	0.2, 0.4, 0.8, 1.6, 3.2, 6.4, and 12.8 ms, selectable in groups of 4	0.2, 0.4, 0.8, 1.6, 3.2, 6.4, and 12.8 ms, selectable in groups of 4	0.2, 0.4, 0.8, 1.6, 3.2, 6.4, and 12.8 ms, selectable in groups of 4	0.2, 0.4, 0.8, 1.6, 3.2, 6.4, and 12.8 ms, selectable in groups of 4
Number of inputs on simultaneously	8	8	16	16
Cable length (max.), in meters	Shielded: 500 m normal inputs Unshielded: 300 m normal inputs	Shielded: 500 m normal inputs Unshielded: 300 m normal inputs	Shielded: 500 m normal inputs Unshielded: 300 m normal inputs	Shielded: 500 m normal inputs Unshielded: 300 m normal inputs

Table A-52 Digital outputs

Model	EM Digital 8 x Inputs/ Digital 8 x Outputs (EM DT16)	EM Digital 8 x Inputs/ 8 x Relay Outputs (EM DR16)	EM Digital 16 x Inputs/ Digital 16 x Outputs (EM DT32)	EM Digital 16 x Inputs/ 16 x Relay Outputs (EM DR32)
Number of outputs	8	8	16	16
Type	Solid state-MOSFET (sourcing)	Relay, dry contact	Solid state-MOSFET (sourcing)	Relay, dry contact
Voltage range	20.4 to 28.8 V DC	5 to 30 V DC or 5 to 250 V AC	20.4 to 28.8 V DC	5 to 30 V DC or 5 to 250 V AC
Logic 1 signal at max. current	20 V DC, min.	--	20 V DC, min.	--
Logic 0 signal with 10 KΩ load	0.1 V DC, max.	--	0.1 V DC, max.	--
Rated current per point (max.)	0.75 A	2 A	0.75 A	2 A
Rated current per common (max.)	3 A	8 A	6 A	8 A
Lamp load	5 W	30 W DC/200 W AC	5 W	30 W DC/200 W AC
ON state contact resistance	0.6 Ω max.	0.2 Ω max. when new	0.6 Ω max.	0.2 Ω max. when new
Leakage current per point	10 µA max.	--	10 µA max.	--
Surge current	8 A for 100 ms max.	7 A with contacts closed	8 A for 100 ms max.	7 A with contacts closed
Overload protection	No	No	No	No
Isolation (field side to logic)	500 V AC for 1 minute	1500 V AC for 1 minute (coil to contact) None (coil to logic)	500 V AC for 1 minute	1500 V AC for 1 minute (coil to contact) None (coil to logic)
Isolation resistance	--	100 M Ω min. when new	--	100 M Ω min. when new

Model	EM Digital 8 x Inputs/ Digital 8 x Outputs (EM DT16)	EM Digital 8 x Inputs/ 8 x Relay Outputs (EM DR16)	EM Digital 16 x Inputs/ Digital 16 x Outputs (EM DT32)	EM Digital 16 x Inputs/ 16 x Relay Outputs (EM DR32)
Isolation between open contacts	--	750 V AC for 1 minute	--	750 V AC for 1 minute
Isolation groups	2	2	3	4
Inductive clamp voltage	Minus 48 V	--	Minus 48 V	--
Switching delay	50 µs max., off to on 200 µs max., on to off	10 ms max.	50 µs max., off to on 200 µs max., on to off	10 ms max.
Lifetime mechanical (no load)	--	10,000,000 open/ close cycles	--	10,000,000 open/ close cycles
Lifetime contacts at rated load	--	100,000 open/close cycles	--	100,000 open/close cycles
Output behavior in STOP	Last value or substitute value (default value 0)	Last value or substitute value (default value 0)	Last value or substitute value (default value 0)	Last value or substitute value (default value 0)
Number of outputs on simultaneously	8	8	16	16
Cable length (max.), in meters	Shielded: 500 m Unshielded: 150 m	Shielded: 500 m Unshielded: 150 m	Shielded: 500 m Unshielded: 150 m	Shielded: 500 m Unshielded: 150 m

Table A-53 Wiring diagrams for the EM DT16 Digital 8 x Inputs / Digital 8 x Outputs (6ES7288-2DT16-0AA0) and EM DR16 Digital 8 x Inputs / 8 x Relay Outputs (6ES7288-2DR16-0AA0)

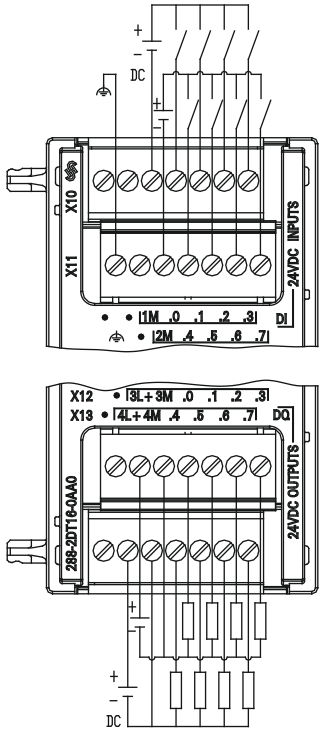
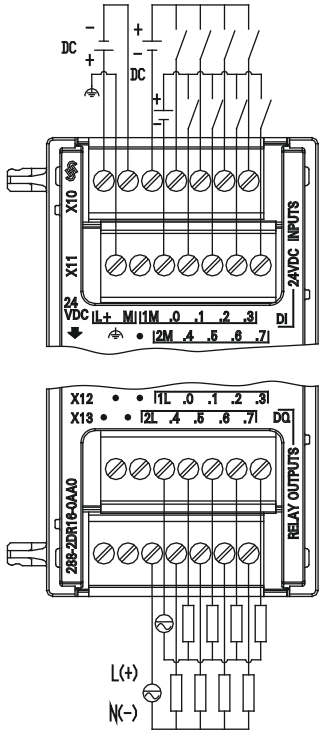
EM DT16 Digital 8 x Inputs / Digital 8 x Outputs (6ES7288-2DT16-0AA0)	EM DR16 Digital 8 x Inputs / 8 x Relay Outputs (6ES7288-2DR16-0AA0)
	

Table A-54 Connector pin locations for EM DT16 Digital 8 x Inputs / Digital 8 x Outputs (6ES7288-2DT16-0AA0)

Pin	X10	X11	X12	X13
1	No connection	Functional Earth	No connection	No connection
2	No connection	No connection	3L+ / 24 V DC	4L+ / 24 V DC
3	1M	2M	3M / 24 V DC	4M / 24 V DC
4	DI a.0	DI a.4	DQ a.0	DQ a.4
5	DI a.1	DI a.5	DQ a.1	DQ a.5
6	DI a.2	DI a.6	DQ a.2	DQ a.6
7	DI a.3	DI a.7	DQ a.3	DQ a.7

Table A-55 Connector pin locations for EM DR16 Digital 8 x Inputs / 8 x Relay Outputs (6ES7288-2DR16-0AA0)

Pin	X10	X11	X12	X13
1	L+ / 24 V DC	Functional Earth	No connection	No connection
2	M / 24 V DC	No connection	No connection	No connection
3	1M	2M	1L	2L
4	DI a.0	DI a.4	DQ a.0	DQ a.4
5	DI a.1	DI a.5	DQ a.1	DQ a.5
6	DI a.2	DI a.6	DQ a.2	DQ a.6
7	DI a.3	DI a.7	DQ a.3	DQ a.7

Table A-56 Wiring diagrams for the EM DT32 Digital 16 x Inputs / Digital 16 x Outputs (6ES7288-2DT32-0AA0 and EM DR32 Digital 16 x Inputs / 16 x Relay (6ES7288-2DR32-0AA0)

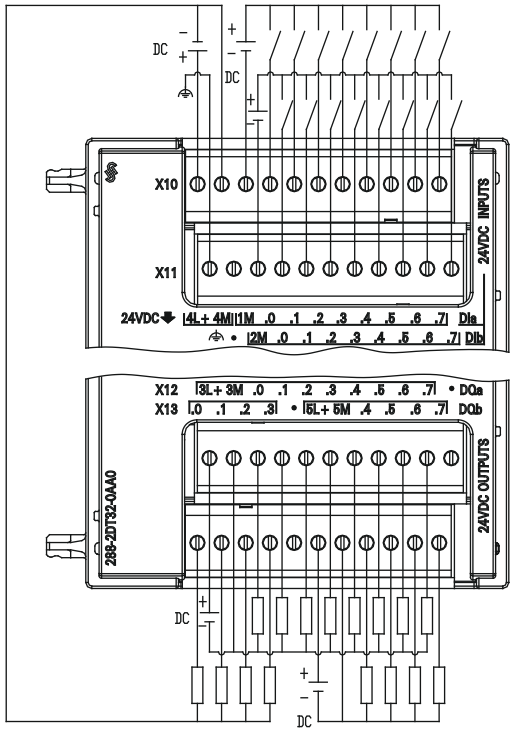
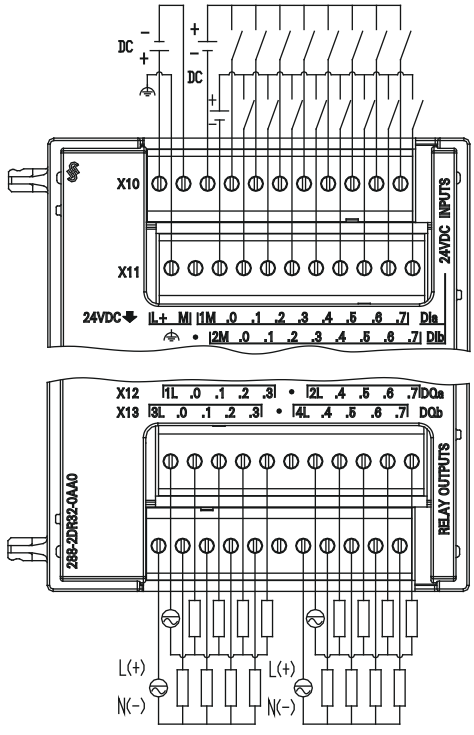
EM DT32 Digital 16 x Inputs / Digital 16 x Outputs (6ES7288-2DT32-0AA0)	EM DR32 Digital 16 x Inputs / 16 x Relay (6ES7288-2DR32-0AA0)
	

Table A-57 Connector pin locations for EM DT32 Digital 16 x Inputs / Digital 16 x Outputs (6ES7288-2DT32-0AA0)

Pin	X10	X11	X12	X13
1	4L+ / 24 V DC[1]	Functional Earth	3L+ / 24 V DC	DQ b.0[1]
2	4M / 24 V DC[1]	No connection	3M / 24 V DC	DQ b.1[1]
3	1M	2M	DQ a.0	DQ b.2[1]
4	DI a.0	DI b.0	DQ a.1	DQ b.3[1]
5	DI a.1	DI b.1	DQ a.2	No connection
6	DI a.2	DI b.2	DQ a.3	5L+ / 24 V DC
7	DI a.3	DI b.3	DQ a.4	5M / 24 V DC
8	DI a.4	DI b.4	DQ a.5	DQ b.4
9	DI a.5	DI b.5	DQ a.6	DQ b.5
10	DI a.6	DI b.6	DQ a.7	DQ b.6
11	DI a.7	DI b.7	No connection	DQ b.7

[1] In same isolation group.

Table A-58 Connector pin locations for EM DR32 Digital 16 x Inputs / 16 x Relay (6ES7288-2DR32-0AA0)

Pin	X10	X11	X12	X13
1	L+ / 24 V DC	Functional Earth	1L	3L
2	M / 24 V DC	No connection	DQ a.0	DQ b.0
3	1M	2M	DQ a.1	DQ b.1
4	DI a.0	DI b.0	DQ a.2	DQ b.2
5	DI a.1	DI b.1	DQ a.3	DQ b.3
6	DI a.2	DI b.2	No connection	No connection
7	DI a.3	DI b.3	2L	4L
8	DI a.4	DI b.4	DQ a.4	DQ b.4
8	DI a.5	DI b.5	DQ a.5	DQ b.5
10	DI a.6	DI b.6	DQ a.6	DQ b.6
11	DI a.7	DI b.7	DQ a.7	DQ b.7

A.4 Analog inputs and outputs expansion modules (EMs)

A.4.1 EM AE04 and EM AE08 analog input specifications

Table A-59 General specifications

Model	EM Analog 4 x inputs (EM AE04)	EM Analog 8 x inputs (EM AE08)
Article number	6ES7288-3AE04-0AA0	6ES7288-3AE08-0AA0
Dimensions W x H x D (mm)	45 x 100 x 81	45 x 100 x 81
Weight	147 grams	186 grams
Power dissipation	1.5 W (no load)	2.0 W (no load)
Current consumption (SM Bus)	80 mA	80 mA
Current consumption (24 V DC)	40 mA (no load)	70 mA (no load)

Table A-60 Analog inputs

Model	EM Analog 4 x inputs (EM AE04)	EM Analog 8 x inputs (EM AE08)
Number of inputs	4	8
Type	Voltage or current (differential), selectable in groups of 2	Voltage or current (differential), selectable in groups of 2
Range	±10 V, ±5 V, ±2.5 V, or 0 to 20 mA	±10 V, ±5 V, ±2.5 V, or 0 to 20 mA
Full scale range (data word)	-27,648 to 27,648	-27,648 to 27,648
Overshoot/undershoot range (data word)	Voltage: 27,649 to 32,511 / -27,649 to -32,512 Current: 27,649 to 32,511 / -4864 to 0	Voltage: 27,649 to 32,511 / -27,649 to -32,512 Current: 27,649 to 32,511 / -4864 to 0 (Refer to Analog input representation for voltage and Analog input representation for current (Page 1026).)
Overflow/underflow (data word)	Voltage: 32,512 to 32,767 / -32,513 to -32,768 Current: 32,512 to 32,767 / -4,865 to -32,768	Voltage: 32,512 to 32,767 / -32,513 to -32,768 Current: 32,512 to 32,767 / -4,865 to -32,768 (Refer to Analog input representation for voltage and Analog input representation for current (Page 1026).)
Resolution	Voltage mode: 12 bits + sign bit Current mode: 12 bits	Voltage mode: 12 bits + sign bit Current mode: 12 bits
Maximum withstand voltage/ current	±35 V / ±40 mA	±35 V / ±40 mA
Smoothing	None, weak, medium or strong	None, weak, medium or strong (Refer to Analog input response times for step response time.) (Page 1025)
Noise rejection	400, 60, 50, or 10 Hz	400, 60, 50, or 10 Hz
Input impedance	≥1 M Ω (voltage) / 290 Ω (current)	≥9 M Ω (voltage) / 250 Ω (current)
Isolation (field side to logic)	None	None

Model	EM Analog 4 x inputs (EM AE04)	EM Analog 8 x inputs (EM AE08)
Accuracy (25 °C / -20 to 60 °C)	Voltage mode: ±0.1% / ±0.2% of full scale Current mode: ±0.2% / ±0.3% of full scale	Voltage mode: ±0.1% / ±0.2% of full scale Current mode: ±0.2% / ±0.3% of full scale
Measuring principle	Actual value conversion	Actual value conversion
Common mode rejection	40 dB, DC to 60 Hz	40 dB, DC to 60 Hz
Operational signal range	Signal plus common mode voltage must be less than +12 V and greater than -12 V	Signal plus common mode voltage must be less than +12 V and greater than -12 V
Cable length (max.), in meters	100 m twisted and shielded	100 m twisted and shielded

Table A-61 Diagnostics

Model	EM Analog 4 x inputs (EM AE04)	EM Analog 8 x inputs (EM AE08)
Overflow/underflow	Yes	Yes
24 V DC low voltage	Yes	Yes

EM AE04 and EM AE08 wiring current transducers

Wiring current transducers are available as 2-wire transducers and 4-wire transducers as shown below.

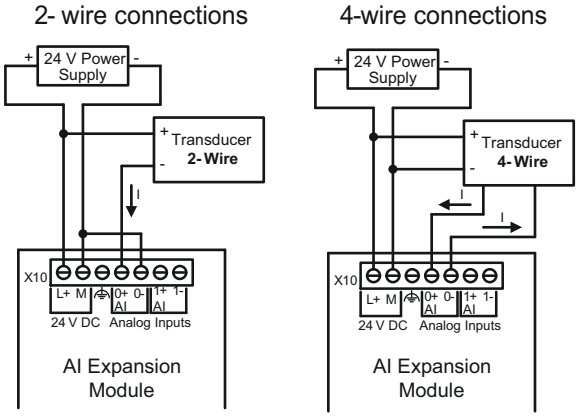


Table A-62 Wiring diagram EM AE04 Analog 4 x Inputs (6ES7288-3AE04-0AA0) and EM AE08 Analog 8 x Inputs (6ES7288-3AE08-0AA0

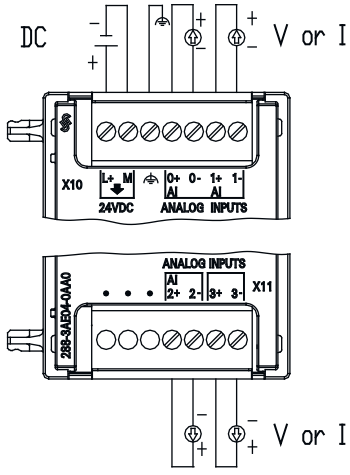
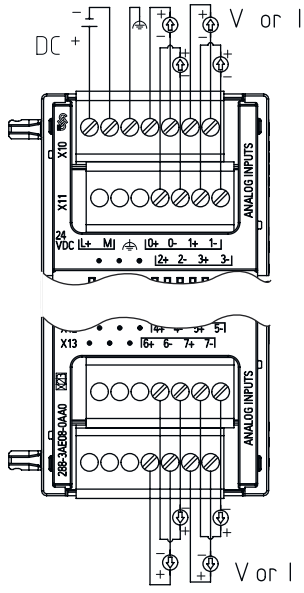
EM AE04 Analog 4 x Inputs (6ES7288-3AE04-0AA0)	**EM AE08 Analog 8 x Inputs (6ES7288-3AE08-0AA0)**
	
Note: Connectors must be gold. See Appendix F, Spare parts and other hardware, for article number.	

Table A-63 Connector pin locations for EM AE04 Analog 4 x Inputs (6ES7288-3AE04-0AA0)

Pin	X10 (gold)	X11 (gold)
1	L+ / 24 V DC	No connection
2	M / 24 V DC	No connection
3	Functional Earth	No connection
4	AI 0+	AI 2+
5	AI 0-	AI 2-
6	AI 1+	AI 3+
7	AI 1-	AI 3-

Table A-64 Connector pin locations for EM AE08 Analog 8 x Inputs (6ES7288-3AE08-0AA0)

Pin	X10 (gold)	X11 (gold)	X12 (gold)	X13 (gold)
1	L+ / 24 V DC	No connection	No connection	No connection
2	M / 24 V DC	No connection	No connection	No connection
3	Functional Earth	No connection	No connection	No connection
4	AI 0+	AI 2+	AI 4+	AI 6+
5	AI 0-	AI 2-	AI 4-	AI 6-

Pin	X10 (gold)	X11 (gold)	X12 (gold)	X13 (gold)
6	AI 1+	AI 3+	AI 5+	AI 7+
7	AI 1-	AI 3-	AI 5-	AI 7-

A.4.2 EM AQ02 and EM AQ04 analog output module specifications

Table A-65 General specifications

Technical data	EM Analog 2 x outputs (EM AQ02)	EM Analog 4 x outputs (EM AQ04)
Article number	6ES7288-3AQ02-0AA0	6ES7288-3AQ04-0AA0
Dimensions W x H x D (mm)	45 x 100 x 81	45 x 100 x 81
Weight	147.1 grams	170.5 grams
Power dissipation	1.5 W (no load)	2.1 W (no load)
Current consumption (SM Bus)	60 mA	60 mA
Current consumption (24 V DC)	50 mA (no load)	75 mA (no load)
	90 mA (20 mA load per channel)	155 mA (20 mA load per channel)

Table A-66 Analog outputs

Technical data	EM Analog 2 x outputs (EM AQ02)	EM Analog 4 x outputs (EM AQ04)
Number of outputs	2	4
Type	Voltage or current	Voltage or current
Range	±10 V or 0 to 20 mA	±10 V or 0 to 20 mA
Resolution	Voltage mode: 11 bits + sign bit Current mode: 11 bits	Voltage mode: 11 bits + sign bit Current mode: 11 bits
Full scale range (data word)	Voltage: -27,648 to 27,648 Current: 0 to 27,648	Voltage: -27,648 to 27,648 Current: 0 to 27,648 (Refer to the output ranges for voltage and current (Page 1027).)
Accuracy (25 °C / -20 to 60 °C)	±0.5% / ±1.0% of full scale	±0.5% / ±1.0% of full scale
Settling time (95% of new value)	Voltage: 300 µs (R), 750 µs (1 µF) Current: 600 µs (1 mH), 2 ms (10 mH)	Voltage: 300 µs (R), 750 µs (1 µF) Current: 600 µs (1 mH), 2 ms (10 mH)
Load impedance	Voltage: ≥ 1000 Ω Current: ≤ 500 Ω	Voltage: ≥ 1000 Ω Current: ≤ 600 Ω
Output behavior in STOP	Last value or substitute value (default value 0)	Last value or substitute value (default value 0)
Isolation (field side to logic)	None	None
Cable length (max.), in meters	100 m twisted and shielded	100 m twisted and shielded

Table A-67 Diagnostics

Technical data	EM Analog 2 x outputs (EM AQ02)	EM Analog 4 x outputs (EM AQ04)
Overflow/underflow	Yes	Yes
Short to ground (voltage mode only)	Yes	Yes
Wire break (current mode only)	Yes	Yes
24 V DC low voltage	Yes	Yes

Table A-68 Wiring diagram for the EM AQ02 Analog 2 x Outputs (6ES7288-3AQ02-0AA0) and EM AQ04 Analog 4 x Outputs (6ES7288-3AQ04-0AA0)

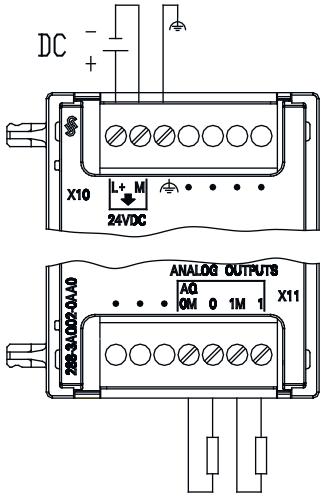
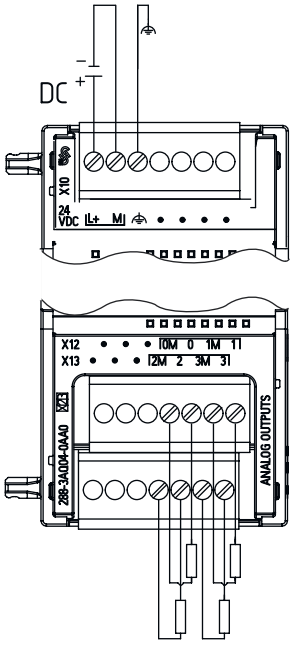
EM AQ02 Analog 2 x Outputs (6ES7288-3AQ02-0AA0)	EM AQ04 Analog 4 x Outputs (6ES7288-3AQ04-0AA0)
	
Note: Connectors must be gold. See Appendix F, Spare parts and other hardware, for article number.	

Table A-69 Connector pin locations for EM AQ02 Analog 2 x Outputs (6ES7288-3AQ02-0AA0)

Pin	X10 (gold)	X11 (gold)
1	L+ / 24 V DC	No connection
2	M / 24 V DC	No connection
3	Functional Earth	No connection
4	No connection	AQ 0M
5	No connection	AQ 0

Pin	X10 (gold)	X11 (gold)
6	No connection	AQ 1M
7	No connection	AQ 1

Table A-70 Connector pin locations for EM AQ04 Analog 4 x Outputs (6ES7288-3AQ04-0AA0)

Pin	X10 (gold)	X11 (gold)	X12 (gold)	X13 (gold)
1	L+ / 24 V DC	No connection	No connection	No connection
2	M / 24 V DC	No connection	No connection	No connection
3	Functional Earth	No connection	No connection	No connection
4	No connection	No connection	AQ 0M	AQ 2M
5	No connection	No connection	AQ 0	AQ 2
6	No connection	No connection	AQ 1M	AQ 3M
7	No connection	No connection	AQ 1	AQ 3

A.4.3 EM AM03 and EM AM06 analog input/output module specifications

Table A-71 General specifications

Technical data	EM 2 x Analog Inputs / 1 x Analog Outputs (AM03)	EM 4 x Analog Inputs / 2 x Analog Outputs (AM06)
Article number	6ES7288-3AM03-0AA0	6ES7288-3AM06-0AA0
Dimensions W x H x D (mm)	45 x 100 x 81	45 x 100 x 81
Weight	172 grams	173.4 grams
Power dissipation	1.1 W (no load)	2.0 W (no load)
Current consumption (SM Bus)	60 mA	80 mA
Current consumption (24 V DC)	30 mA (no load)	60 mA (no load)
	50 mA (20 mA load per channel)	100 mA (20 mA load per channel)

Table A-72 Analog inputs

Model	EM 2 x Analog Inputs / 1 x Analog Outputs (AM03)	EM 4 x Analog Inputs / 2 x Analog Outputs (AM06)
Number of inputs	2	4
Type	Voltage or current (differential): Selectable in groups of 2	Voltage or current (differential): Selectable in groups of 2
Range	±10 V, ±5 V, ±2.5 V, or 0 to 20 mA	±10 V, ±5 V, ±2.5 V, or 0 to 20 mA
Full scale range (data word)	-27,648 to 27,648	-27,648 to 27,648
Overshoot/undershoot range (data word)	Voltage: 27,649 to 32,511 / -27,649 to -32,512 Current: 27,649 to 32,511 / -4,864 to 0	Voltage: 27,649 to 32,511 / -27,649 to -32,512 Current: 27,649 to 32,511 / -4,864 to 0

Model	EM 2 x Analog Inputs / 1 x Analog Outputs (AM03)	EM 4 x Analog Inputs / 2 x Analog Outputs (AM06)
Overflow/underflow (data word)	Voltage: 32,512 to 32,767 / -32,513 to -32,768 Current: 32,512 to 32,767 / -4,865 to -32,768	Voltage: 32,512 to 32,767 / -32,513 to -32,768 Current: 32,512 to 32,767 / -4,865 to -32,768
Resolution	Voltage mode: 12 bits + sign Current mode: 12 bits	Voltage mode: 12 bits + sign Current mode: 12 bits
Maximum withstand voltage/ current	±35 V / ±40 mA	±35 V / ±40 mA
Smoothing	None, weak, medium, or strong	None, weak, medium, or strong
Noise rejection	400, 60, 50, or 10 Hz	400, 60, 50, or 10 Hz
Input impedance	≥1 M Ω (voltage) / 290 Ω (current)	≥1 M Ω (voltage) / 290 Ω (current)
Isolation (field side to logic)	None	None
Accuracy (25 °C / -20 to 60 °C)	Voltage mode: ±0.1% /±0.2% of full scale Current mode: ±0.2% /±0.3% of full scale	Voltage mode: ±0.1% /±0.2% of full scale Current mode: ±0.2% /±0.3% of full scale
Analog to digital conversion time	625 µs (400 Hz rejection)	625 µs (400 Hz rejection)
Common mode rejection	40 dB, DC to 60 Hz	40 dB, DC to 60 Hz
Operational signal range	Signal plus common mode voltage must be less than +12 V and greater than -12 V	Signal plus common mode voltage must be less than +12 V and greater than -12 V
Cable length (max.), in meters	100 m twisted and shielded	100 m twisted and shielded

Table A-73 Analog outputs

Technical data	EM 2 x Analog Inputs / 1 x Analog Outputs (AM03)	EM 4 x Analog Inputs / 2 x Analog Outputs (AM06)
Number of outputs	1	2
Type	Voltage or current	Voltage or current
Range	±10 V or 0 to 20 mA	±10 V or 0 to 20 mA
Resolution	Voltage mode: 11 bits + sign Current mode: 11 bits	Voltage mode: 11 bits + sign Current mode: 11 bits
Full scale range (data word)	Voltage: -27,648 to 27,648 Current: 0 to 27,648	Voltage: -27,648 to 27,648 Current: 0 to 27,648
Accuracy (25 °C / -20 to 60 °C)	±0.5% / ±1.0% of full scale	±0.5% / ±1.0% of full scale
Settling time (95% of new value)	Voltage: 300 µs (R), 750 µs (1 µF) Current: 600 µs (1 mH), 2 ms (10 mH)	Voltage: 300 µs (R), 750 µs (1 µF) Current: 600 µs (1 mH), 2 ms (10 mH)
Load impedance	Voltage: ≥ 1000 Ω Current: ≤ 500 Ω	Voltage: ≥ 1000 Ω Current: ≤ 500 Ω
Output behavior in STOP	Last value or substitute value (default value 0)	Last value or substitute value (default value 0)
Isolation (field side to logic)	None	None
Cable length (max.), in meters	100 m twisted and shielded	100 m twisted and shielded

Table A-74 Diagnostics

Model	EM 2 x Analog Inputs / 1 x Analog Outputs (AM03)	EM 4 x Analog Inputs / 2 x Analog Outputs (AM06)
Overflow/underflow	Yes	Yes
Short to ground (voltage mode only)	Yes	Yes
Wire break (current mode only)	Yes	Yes
24 V DC low voltage	Yes	Yes

EM AM03 wiring current transducers

Wiring current transducers are available as 2-wire transducers and 4-wire transducers as shown below.

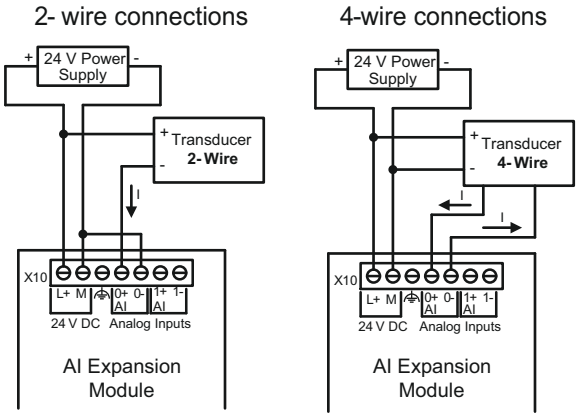


Table A-75 Wiring diagrams for the EM AM03 2 x Analog Inputs / 1 x Analog Outputs (6ES7288-3AM03-0AA and the EM AM06 4 x Analog Inputs / 2 x Analog Outputs (6ES7288-3AM06-0AA0)

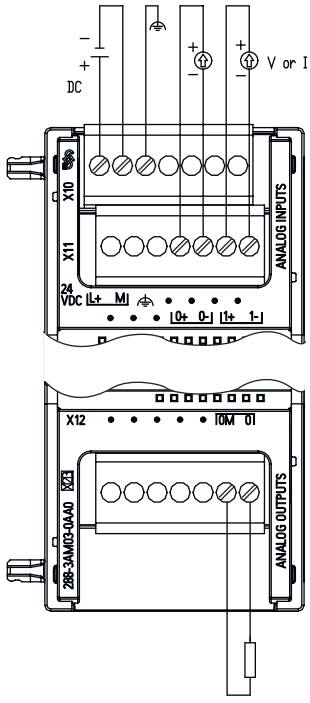
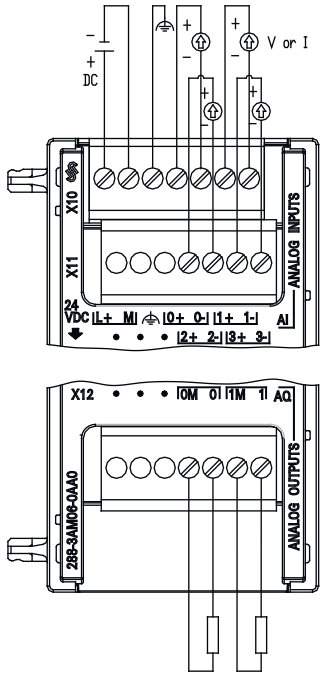
EM AM03 2 x Analog Inputs / 1 x Analog Outputs (6ES7288-3AM03-0AA0)	**EM AM06 4 x Analog Inputs / 2 x Analog Outputs (6ES7288-3AM06-0AA0)**
	
Note: Connectors must be gold. See Appendix F, Spare parts and other hardware, for article number.	

Table A-76 Connector pin locations for AM03 2 x Analog Inputs / 1 x Analog Outputs (6ES7288-3AM03-0AA0)

Pin	X10 (gold)	X11 (gold)	X12 (gold)
1	L+ / 24 V DC	No connection	No connection
2	M / 24 V DC	No connection	No connection
3	Functional Earth	No connection	No connection
4	No connection	AI 0+	No connection
5	No connection	AI 0-	No connection
6	No connection	AI 1+	AQ 0M
7	No connection	AI 1-	AQ 0

Table A-77 Connector pin locations for AM06 4 x Analog Inputs / 2 x Analog Outputs (6ES7288-3AM06-0AA0)

Pin	X10 (gold)	X11 (gold)	X12 (gold)
1	L+ / 24 V DC	No connection	No connection
2	M / 24 V DC	No connection	No connection
3	Functional Earth	No connection	No connection

Pin	X10 (gold)	X11 (gold)	X12 (gold)
4	AI 0+	AI 2+	AQ 0M
5	AI 0-	AI 2-	AQ 0
6	AI 1+	A1 3+	AQ 1M
7	AI 1-	A1 3-	AQ 1

A.4.4 Step response of the analog inputs

Table A-78 Step response (ms), 0 to full-scale measured at 95%

Smoothing selection (sample averaging)	Noise reduction/rejection frequency (Integration time selection)			
	400 Hz (2.5 ms)	60 Hz (16.6 ms)	50 Hz (20 ms)	10 Hz (100 ms)
None (1 cycle): No averaging	4 ms	18 ms	22 ms	100 ms
Weak (4 cycles): 4 samples	9 ms	52 ms	63 ms	320 ms
Medium (16 cycles): 16 samples	32 ms	203 ms	241 ms	1200 ms
Strong (32 cycles): 32 samples	61 ms	400 ms	483 ms	2410 ms
Sample time • 4 AI x 13 bits • 8 AI x 13 bits	• 0.625 ms • 1.25 ms	• 4.17 ms • 4.17 ms	• 5 ms • 5 ms	• 25 ms • 25 ms

A.4.5 Sample time and update times for the analog inputs

Table A-79 Sample time and update time

Rejection frequency (Integration time)	Sample time	Module update time for all channels	
		4-channel SM	8-channel SM
400 Hz (2.5 ms)	• 4-channel SM: 0.625 ms • 8-channel SM: 1.250 ms	0.625 ms	1.250 ms
60 Hz (16.6 ms)	4.170 ms	4.17 ms	4.17 ms
50 Hz (20 ms)	5.000 ms	5 ms	5 ms
10 Hz (100 ms)	25.000 ms	25 ms	25 ms

A.4.6 Measurement ranges of the analog inputs for voltage and current (SB and EM)

Table A-80 Analog input representation for voltage (SB and EM)

System		Voltage Measuring Range				
Decimal	**Hexadecimal**	**±10 V**	**±5 V**	**±2.5 V**	**±1.25 V**	
32767	7FFF[1]	11.851 V	5.926 V	2.963 V	1.481 V	Overflow
32512	7F00					
32511	7EFF	11.759 V	5.879 V	2.940 V	1.470 V	Overshoot range
27649	6C01					
27648	6C00	10 V	5 V	2.5 V	1.250 V	Rated range
20736	5100	7.5 V	3.75 V	1.875 V	0.938 V	
1	1	361.7 µV	180.8 µV	90.4 µV	45.2 µV	
0	0	0 V	0 V	0 V	0 V	
-1	FFFF					
-20736	AF00	-7.5 V	-3.75 V	-1.875 V	-0.938 V	
-27648	9400	-10 V	-5 V	-2.5 V	-1.250 V	
-27649	93FF					Undershoot range
-32512	8100	-11.759 V	-5.879 V	-2.940 V	-1.470 V	
-32513	80FF					Underflow
-32768	8000	-11.851 V	-5.926 V	-2.963 V	-1.481 V	

[1] 7FFF can be returned for one of the following reasons: overflow (as noted in this table), before valid values are available (for example immediately upon a power up), or if a wire break is detected.

Table A-81 Analog input representation for current (SB and EM)

System		Current measuring range	
Decimal	**Hexadecimal**	**0 mA to 20 mA**	
32767	7FFF	23.70 mA	Overflow
32512	7F00		
32511	7EFF	23.52 mA	Overshoot range
27649	6C01		
27648	6C00	20 mA	Nominal range
20736	5100	15 mA	
1	1	723.4 nA	
0	0	0 mA	
-1	FFFF		Undershoot range
-4864	ED00	-3.52 mA	
-4865	ECFF		Underflow
-32768	8000		

A.4.7 Measurement ranges of the analog outputs for voltage and current (SB and EM)

Table A-82 Analog output representation for voltage (SB and EM)

System		Voltage Output Range	
Decimal	**Hexadecimal**	**± 10 V**	
32767	7FFF	See note 1	Overflow
32512	7F00	See note 1	
32511	7EFF	11.76 V	Overshoot range
27649	6C01		
27648	6C00	10 V	Rated range
20736	5100	7.5 V	
1	1	361.7 µ V	
0	0	0 V	
-1	FFFF	-361.7 µ V	
-20736	AF00	-7.5 V	
-27648	9400	-10 V	
-27649	93FF		Undershoot range
-32512	8100	-11.76 V	
-32513	80FF	See note 1	Underflow
-32768	8000	See note 1	

[1] In an overflow or underflow condition, analog outputs will take on the substitute value of the STOP mode.

Table A-83 Analog output representation for current (SB and EM)

System		Current output range	
Decimal	**Hexadecimal**	**0 mA to 20 mA**	
32767	7FFF	See note 1	Overflow
32512	7F00	See note 1	
32511	7EFF	23.52 mA	Overshoot range
27649	6C01		
27648	6C00	20 mA	Rated range
20736	5100	15 mA	
1	1	723.4 nA	
0	0	0 mA	
-1	FFFF		Not possible. Output value limited to 0 mA.
-6912	E500		
-6913	E4FF		
-32512	8100		
-32513	80FF	See note 1	Underflow
-32768	8000	See note 1	

[1] In an overflow or underflow condition, analog outputs will take on the substitute value of the STOP mode.

A.5 Thermocouple and RTD expansion modules (EMs)

A.5.1 Thermocouple expansion modules (EMs)

A.5.1.1 EM AT04 thermocouple specifications

Table A-84 General specifications

Model	EM AT04 AI 4 x 16 bit TC
Article number	6ES7288-3AT04-0AA0
Dimensions W x H x D (mm)	45 x 100 x 81
Weight	125 grams
Power dissipation	1.5 W
Current consumption (SM Bus)	80 mA
Current consumption (24 V DC) [1]	40 mA

[1] 20.4 to 28.8 V DC (Class 2, Limited Power, or sensor power from PLC)

Table A-85 Analog inputs

Model		EM AT04 AI 4 x 16 bit TC
Number of inputs		4
Range Nominal range (data word) Overrange/underrange (data word) Overflow/underflow (data word)		See Thermocouple selection table.
Resolution	Temperature	0.1 °C / 0.1 °F
	Voltage	15 bits plus sign
Maximum withstand voltage		± 35
Noise rejection		85 dB for selected filter setting (10 Hz, 50 Hz, 60 Hz or 400 Hz)
Common mode rejection		> 120 dB at 120 V AC
Impedance		≥ 10 MΩ
Isolation	Field to logic	500 V AC
	Field to 24 V DC	500 V AC
	24 V DC to logic	500 V AC
Channel to channel isolation		--
Accuracy		See Thermocouple selection table.
Repeatability		±0.05% FS
Measuring principle		Integrating

Model	EM AT04 AI 4 x 16 bit TC
Module update time	See Filter selection table.
Cold junction error	±1.5 °C
Cable length (meters)	100 m to sensor max.
Wire resistance	100 Ω max.

Table A-86 Diagnostics

Model	EM AT04 AI 4 x 16 bit TC
Overflow/underflow [1]	Yes
Wire break [2]	Yes
24 V DC low voltage [1]	Yes

[1] The overflow, underflow and low voltage diagnostic alarm information will be reported in the analog data values even if the alarms are disabled in the module configuration.

[2] When wire break alarm is disabled and an open wire condition exists in the sensor wiring, the module may report random values.

The EM AT04 Thermocouple (TC) analog expansion module measures the value of voltage connected to the module inputs. The temperature measurement type can be either "Thermocouple" or "Voltage".

- "Thermocouple": The value will be reported in degrees multiplied by ten (for example, 25.3 degrees will be reported as decimal 253).
- "Voltage": The nominal range full scale value will be decimal 27648.

Table A-87 Wiring diagram for the EM AT04 Thermocouple 4 x 16 bit (6ES7288-3AT04-0AA0)

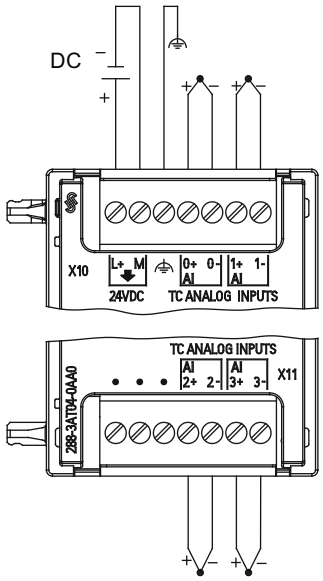
EM AT04 4 x 16 bit (6ES7288-3AT04-0AA0)	
	
Note: Connectors must be gold. See Appendix F, Spare Parts and other hardware for article number.	

Table A-88 Connector pin locations for EM AT04 4 x 16 bit (6ES7288-3AT04-0AA0)

Pin	X10 (gold)	X11 (gold)
1	L+ / 24 V DC	No connection
2	M / 24 V DC	No connection
3	Functional Earth	No connection
4	AI 0+ /TC	AI 2+ /TC
5	AI 0- /TC	AI 2- /TC
6	AI 1+ /TC	AI 3+ /TC
7	AI 1- /TC	AI 3- /TC

Note

Unused analog inputs should be shorted.

Thermocouples are formed whenever two dissimilar metals are electrically bonded to each other. A voltage is generated that is proportional to the junction temperature. This voltage is small; one microvolt could represent many degrees. Measuring the voltage from a thermocouple, compensating for extra junctions, and then linearizing the result forms the basis of temperature measurement using thermocouples.

When you connect a thermocouple to the EM AT04 Thermocouple module, the two dissimilar metal wires are attached to the module at the module signal connector. The place where the two dissimilar wires are attached to each other forms the sensor thermocouple.

Two more thermocouples are formed where the two dissimilar wires are attached to the signal connector. The connector temperature causes a voltage that adds to the voltage from the sensor thermocouple. If this voltage is not corrected, then the temperature reported will deviate from the sensor temperature.

Cold junction compensation is used to compensate for the connector thermocouple. Thermocouple tables are based on a reference junction temperature, usually zero degrees Celsius. The cold junction compensation compensates the connector to zero degrees Celsius. The cold junction compensation restores the voltage added by the connector thermocouples. The temperature of the module is measured internally, and then converted to a value to be added to the sensor conversion. The corrected sensor conversion is then linearized using the thermocouple tables.

For optimum operation of the cold junction compensation, the thermocouple module must be located in a thermally stable environment. Slow variation (less than 0.1 °C/minute) in ambient module temperature is correctly compensated within the module specifications. Air movement across the module will also cause cold junction compensation errors.

If better cold junction error compensation is needed, an external iso-thermal terminal block may be used. The thermocouple module provides for use of a 0 °C referenced or 50 °C referenced terminal block.

The ranges and accuracy for the different thermocouple types supported by the EM AT04 Thermocouple expansion module are shown in the table below.

Table A-89 EM AT04 Thermocouple selection table

Type	Under-range minimum[1]	Nominal range low limit	Nominal range high limit	Over-range maximum[2]	Normal range [3, 4] accuracy @ 25 °C	Normal range [1, 2] accuracy -20 °C to 60 °C
J	-210.0 °C	-150.0 °C	1200.0 °C	1450.0 °C	±0.3 °C	±0.6 °C
K	-270.0 °C	-200.0 °C	1372.0 °C	1622.0 °C	±0.4 °C	±1.0 °C
T	-270.0 °C	-200.0 °C	400.0 °C	540.0 °C	±0.5 °C	±1.0 °C
E	-270.0 °C	-200.0 °C	1000.0 °C	1200.0 °C	±0.3 °C	±0.6 °C
R & S	-50.0 °C	100.0 °C	1768.0 °C	2019.0 °C	±1.0 °C	±2.5 °C
B	0.0 °C	200.0 °C	800.0 °C	--	±2.0 °C	±2.5 °C
	--	800.0 °C	1820.0 °C	1820 °C	±1.0 °C	±2.3 °C
N	-270.0 °C	-200.0 °C	1300.0 °C	1550.0 °C	±1.0 °C	±1.6 °C
C	0.0 °C	100.0 °C	2315.0 °C	2500.0 °C	±0.7 °C	±2.7 °C
TXK/XK(L)	-200.0 °C	-150.0 °C	800.0 °C	1050 °C	±0.6 °C	±1.2 °C
Voltage	-32512	-27648 -80mV	27648 80mV	32511	±0.05%	±0.1%

1 Thermocouple values below the under-range minimum value are reported as -32768.

2 Thermocouple values above the over-range maximum value are reported as 32767.

3 Internal cold junction error is ±1.5 °C for all ranges. This adds to the error in this table. The module requires at least 30 minutes of warm-up time to meet this specification.

4 In the presence of radiated radio frequency of 970 MHz to 990 MHz, the accuracy of the EM AT04 AI 4 x 16 bit TC may be degraded.

Note

Thermocouple channel

Each channel on the Thermocouple expansion module can be configured with a different thermocouple type (selectable in the software during configuration of the module).

Table A-90 Noise reduction and update times for the EM AT04 Thermocouple

Rejection frequency selection	Integration time	4 Channel module update time (seconds)
400 Hz (2.5 ms)	10 ms [1]	0.143
60 Hz (16.6 ms)	16.67 ms	0.223
50 Hz (20 ms)	20 ms	0.263
10 Hz (100 ms)	100 ms	1.225

1 To maintain module resolution and accuracy when 400 Hz rejection is selected, the integration time is 10 ms. This selection also rejects 100 Hz and 200 Hz noise.

It is recommended for measuring thermocouples that a 100 ms integration time be used. The use of smaller integration times will increase the repeatability error of the temperature readings.

Note

After power is applied, the module performs internal calibration for the analog-to-digital converter. During this time the module reports a value of 32767 on each channel until valid data is available on that channel. Your user program may need to allow for this initialization time. Because the configuration of the module can vary the length of the initialization time, you should verify the behavior of the module in your configuration. If required, you can include logic in your user program to accommodate the initialization time of the module.

Representation of analog values for Thermocouple Type J

A representation of the analog values of thermocouples type J is shown in the table below.

Table A-91 Representation of analog values of thermocouples type J

Type J in °C	Units		Type J in °F	Units		Range
	Decimal	Hexadecimal		Decimal	Hexadecimal	
> 1450.0	32767	7FFF	> 2642.0	32767	7FFF	Overflow
1450.0 : 1200.1	14500 : 12001	38A4 : 2EE1	2642.0 : 2192.2	26420 : 21922	6734 : 55A2	Overrange
1200.0 : -150.0	12000 : -1500	2EE0 : FA24	2192.0 : -238.0	21920 : -2380	55A0 : F6B4	Rated range
-150.1 : -210.0	-1501 : -2100	FA23 : F7CC	-238.2 : -346.0	-2382 : -3460	F6B2 : F27C	Underrange
< -210.0	-32768	8000	< -346.0	-32768	8000	Underflow[1]

[1] Faulty wiring (for example, polarity reversal, or open inputs) or sensor error in the negative range (for example, wrong type of thermocouple) may cause the thermocouple module to signal underflow.

A.5.2 RTD expansion modules (EMs)

EM RTD specifications

Table A-92 General specifications

Technical data	EM RTD 2 x 16 bit (EM AR02)	EM RTD 4 x 16 bit (EM AR04)
Article number	6ES7288-3AR02-0AA0	6ES7288-3AR04-0AA0
Dimensions W x H x D (mm)	45 x 100 x 81	45 x 100 x 81
Weight	148.7 grams	150 grams
Power dissipation	1.5 W	1.5 W
Current consumption (SM Bus)	80 mA	80 mA
Current consumption (24 V DC) [1]	40 mA	40 mA

Table A-93 Analog inputs

Technical data		EM RTD 2 x 16 bit (EM AR02)	EM RTD 4 x 16 bit (EM AR04)
Number of inputs		2	4
Type		Module referenced RTD and Ω	Module referenced RTD and Ω
Range Nominal range (data word) Overshoot/undershoot range (data word) Overflow/underflow (data word)		See RTD Sensor selection table	See RTD Sensor selection table
Resolution	Temperature	0.1 °C / 0.1 °F	0.1 °C / 0.1 °F
	Resistance	15 bits + sign bit	15 bits + sign bit
Maximum withstand voltage		±35 V	±35 V
Noise rejection		85 dB at 10 Hz / 50 Hz /60 Hz / 400 Hz	85 dB at 10 Hz / 50 Hz /60 Hz / 400 Hz
Common mode rejection		>120 dB	>120 dB
Impedance		≥10 M Ω	≥10 M Ω
Isolation	Field side to logic	500 V AC	500 V AC
	Field to 24 V DC	500 V AC	500 V AC
	24 V DC to logic	500 V AC	500 V AC
Channel to channel isolation		0	0
Accuracy		See RTD Sensor selection table	See RTD Sensor selection table
Repeatability		±0.05% FS	±0.05% FS
Maximum sensor dissipation		0.5 m W	0.5 m W
Measuring principle		Sigma-delta	Sigma-delta
Module update time		See Noise reduction selection table	See Noise reduction selection table
Cable length (max.), in meters		100 m to sensor max.	100 m to sensor max.
Wire resistance (max.)	except 10 Ω RTD	20 Ω	20 Ω
	10 Ω RTD	2.7 Ω	2.7 Ω

Table A-94 Diagnostics

Technical data	EM RTD 2 x 16 bit (EM AR02)	EM RTD 4 x 16 bit (EM AR04)
Overflow/underflow [1,2]	Yes	Yes
Wire break [3]	Yes	Yes
24 V DC low voltage [1]	Yes	Yes

[1] The overflow, underflow and low voltage diagnostic alarm information will be reported in the analog data values even if the alarms are disabled in the module configuration.

[2] For resistance ranges underflow detection is never enabled.

[3] When wire break alarm is disabled and an open wire condition exists in the sensor wiring, the module may report random values.

The EM RTD analog expansion module measures the value of resistance connected to the module inputs. The measurement type can be selected as either "Resistor" or "Thermal resistor".

- "Resistor": The nominal range full scale value will be decimal 27648.
- "Thermal resistor": The value will be reported in degrees multiplied by ten (for example, 25.3 degrees will be reported as decimal 253).

The EM RTD module supports measurements with 2-wire, 3-wire and 4-wire connections to the sensor resistor.

Table A-95 Ranges and accuracy for the different sensors supported by the RTD expansion module

Temperature coefficient	RTD type	Under range minimum[1]	Nominal range low limit	Nominal range high limit	Over range maximum[2]	Normal range accuracy @ 25 °C	Normal range accuracy -20 ° C to 60 °C
Pt 0.003850	Pt 10	-243.0 °C	-200.0 °C	850.0 °C	1000.0 °C	±1.0 °C	±2.0 °C
ITS90 DIN EN 60751	Pt 50	-243.0 °C	-200.0 °C	850.0 °C	1000.0 °C	±0.5 °C	±1.0 °C
	Pt 100						
	Pt 200						
	Pt 500						
	Pt 1000						
Pt 0.003902 Pt 0.003916 Pt 0.003920	Pt 100	-243.0 °C	-200.0 °C	850.0 °C	1000.0 °C	± 0.5 °C	±1.0 °C
	Pt 200	-243.0 °C	-200.0 °C	850. 0°C	1000. 0°C	± 0.5 °C	±1.0 °C
	Pt 500						
	Pt 1000						
Pt 0.003910	Pt 10	-273.2 °C	-240.0 °C	1100. 0°C	1295 °C	±1.0 °C	±2.0 °C
	Pt 50	-273.2 °C	-240.0 °C	1100.0 °C	1295 °C	±0.8 °C	±1.6 °C
	Pt 100						
	Pt 500						
Ni 0.006720 Ni 0.006180	Ni 100	-105.0 °C	-60.0 °C	250.0 °C	295.0 °C	±0.5 °C	±1.0 °C
	Ni 120						
	Ni 200						
	Ni 500						
	Ni 1000						
LG-Ni 0.005000	LG-Ni 1000	-105.0 °C	-60.0 °C	250.0 °C	295.0 °C	±0.5 °C	±1.0 °C

Temperature coefficient	RTD type	Under range minimum[1]	Nominal range low limit	Nominal range high limit	Over range maximum[2]	Normal range accuracy @ 25 °C	Normal range accuracy -20 ° C to 60 °C
Ni 0.006170	Ni 100	-105.0 °C	-60.0 °C	180.0 °C	212.4 °C	±0.5 °C	±1.0 °C
Cu 0.004270	Cu 10	-240.0 °C	-200.0 °C	260.0 °C	312.0 °C	±1.0 °C	±2.0 °C
Cu 0.004260	Cu 10	-60.0 °C	-50.0 °C	200.0 °C	240.0 °C	±1.0 °C	±2.0 °C
	Cu 50	-60.0 °C	-50.0 °C	200.0 °C	240.0 °C	±0.6 °C	±1.2 °C
	Cu 100						
Cu 0.004280	Cu 10	-240.0 °C	-200.0 °C	200.0 °C	240.0 °C	±1.0 °C	±2.0 °C
	Cu 50	-240.0 °C	-200.0 °C	200.0 °C	240.0 °C	±0.7 °C	±1.4 °C
	Cu 100						

[1] RTD values below the under-range minimum value report -32768.

[2] RTD values above the over-range maximum value report +32767.

Table A-96 Resistance

Range	Under range minimum	Nominal range low limit	Nominal range high limit	Over range maximum[1]	Normal range accuracy @ 25 °C	Normal range accuracy -20 °C to 60°C
48 Ω	n/a	0 (0 Ω)	27648 (48 Ω)	56.4384 Ω	±0.05%	±0.1%
150 Ω	n/a	0 (0 Ω)	27648 (150 Ω)	176.383 Ω	±0.05%	±0.1%
300 Ω	n/a	0 (0 Ω)	27648 (300 Ω)	352.767 Ω	±0.05%	±0.1%
600 Ω	n/a	0 (0 Ω)	27648 (600 Ω)	705.534 Ω	±0.05%	±0.1%
3000 Ω	n/a	0 (0 Ω)	27648 (3000 Ω)	3527.4 Ω	±0.05%	±0.1%

[1] Resistance values above the over-range minimum value are reported as +32767.

Note

The module reports 32767 on any activated channel with no sensor connected. If open wire detection is also enabled, the module flashes the appropriate red LEDs.

When 600 Ω and 3000 Ω RTD ranges are used with other lower value resistors, the error may increase to two times the specified error.

Best accuracy will be achieved for the 48 Ω RTD ranges if 4 wire connections are used.

The resistance of the connection wires in 2 wire mode will cause an error in the sensor reading and therefore accuracy is not guaranteed.

Table A-97 Noise reduction and update times for the RTD module

Rejection frequency selection	Integration time	Update time (seconds)
400 Hz (2.5 ms)	10 ms [1]	4-/2-wire: 0.142 3-wire: 0.285
60 Hz (16.6 ms)	16.67 ms	4-/2-wire: 0.222 3-wire: 0.445

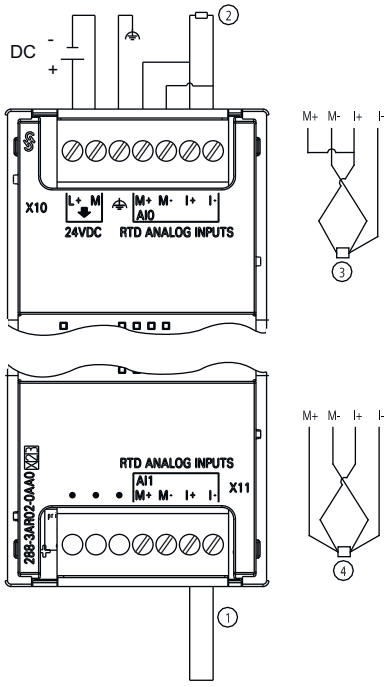
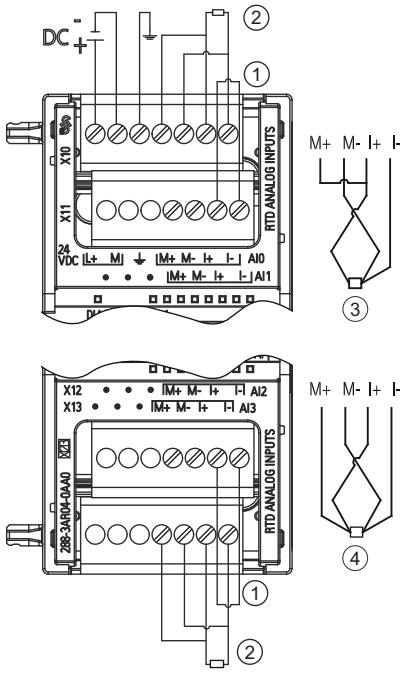
Rejection frequency selection	Integration time	Update time (seconds)
50 Hz (20 ms)	20 ms	4-/2-wire: 0.262 3-wire: .505
10 Hz (100 ms)	100 ms	4-/2-wire: 1.222 3-wire: 2.445

[1] To maintain module resolution and accuracy when the 400 Hz filter is selected, the integration time is 10 ms. This selection also rejects 100 Hz and 200 Hz noise.

Note

After power is applied, the module performs internal calibration for the analog-to-digital converter. During this time the module reports a value of 32767 on each channel until valid data is available on that channel. Your user program may need to allow for this initialization time. Because the configuration of the module can vary the length of the initialization time, you should verify the behavior or the module in your configuration. If required, you can include logic in your user program to accommodate the initialization time of the module.

Table A-98 Wiring diagrams for the EM AR02 RTD 2 x 16 bit (6ES7288-3AR02-0AA0) and EM AR04 RTD 4 x 16 bit (6ES7288-3AR04-0AA0)

EM AR02 RTD 2 x 16 bit (6ES7288-3AR02-0AA0)	EM AR04 RTD 4 x 16 bit (6ES7288-3AR04-0AA0)
	

Note: Connectors must be gold. See Appendix F, Spare parts and other hardware, for article number.

① Loop-back unused RTD inputs
② 2-wire RTD ③ 3-wire RTD ④ 4-wire RTD
Note: Connectors must be gold. See Appendix F, Spare parts and other hardware for article number.

Table A-99 Connector pin locations for EM AR02 RTD 2 x 16 bit (6ES7288-3AR02-0AA0)

Pin	X10 (gold)	X11 (gold)
1	L+ / 24 V DC	No connection
2	M / 24 V DC	No connection
3	Functional Earth	No connection
4	AI 0 M+/RTD	AI 1 M+/RTD
5	AI 0 M-/RTD	AI 1 M-/RTD
6	AI 0 I+/RTD	AI 1 I+/RTD
7	AI 0 I-/RTD	AI 1 I-/RTD

Table A-100 Connector pin locations for EM AR04 RTD 4 x 16 bit (6ES7288-3AR04-0AA0)

Pin	X10 (gold)	X11 (gold)	X12 (gold)	X13 (gold)
1	L+ / 24 V DC	No connection	No connection	No connection
2	M / 24 V DC	No connection	No connection	No connection
3	Functional Earth	No connection	No connection	No connection
4	AI 0 M+/RTD	AI 1 M+/RTD	AI 2 M+/RTD	AI 3 M+/RTD
5	AI 0 M-/RTD	AI 1 M-/RTD	AI 2 M-/RTD	AI 3 M-/RTD
6	AI 0 I+/RTD	AI 1 I+/RTD	AI 2 I+.RTD	AI 3 I+/RTD
7	AI 0 I-/RTD	AI 1 I-/RTD	AI 2 I-/RTD	AI 3 I/-/RTD

A.6 Digital signal boards

A.6.1 SB DT04 digital input/output specifications

Table A-101 General specifications

Technical data	SB Digital 2 x Inputs / 2 x Digital Outputs (DT04)
Article number	6ES7288-5DT04-0AA0
Dimensions W x H x D (mm)	35 x 52.2 x 16
Weight	18.1 grams
Power dissipation	1.0 W
Current consumption (5 V DC)	50 mA
Current consumption (24 V DC)	4 mA / Input used

Table A-102 Digital inputs

Technical data	SB Digital 2 x Inputs / 2 x Digital Outputs (DT04)
Number of inputs	2
Type	Sink (IEC Type 1 sink)
Rated voltage	24 V DC at 4 mA, nominal
Continuous permissible voltage	30 V DC max.
Surge voltage	35 V DC for 0.5 sec.
Logic 1 signal (min.)	15 V DC at 2.5 mA
Logic 0 signal (max.)	5 V DC at 1 mA
Isolation (field side to logic)	500 V AC for 1 minute
Isolation groups	1

Technical data	SB Digital 2 x Inputs / 2 x Digital Outputs (DT04)
Filter times	Individually selectable on each channel: µs: 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 12.8 ms: 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 12.8
Number of inputs on simultaneously	2
Cable length (max.), in meters	Shielded: 500 m normal inputs Unshielded: 300 m normal inputs

Table A-103 Digital outputs

Technical data	SB Digital 2 x Inputs / 2 x Digital Outputs (DT04)
Number of outputs	2
Output type	Solid state - MOSFET (sourcing)
Voltage range	20.4 to 28.8 V DC
Logic 1 signal at max. current	20 V DC min.
Logic 0 signal at max. current	0.1 V DC max.
Rated current per point (max.)	0.5 A
Rated current per common (max.)	1 A
Lamp load	5 W
On state contact resistance	0.6 Ω max.
Leakage current per point	10 µA max.
Surge current	5 A for 100 ms max.
Overload protection	No
Isolation (field side to logic)	500 V AC for 1 minute
Isolation groups	1
Inductive clamp voltage	L+ minus 48 V, 1 W dissipation
Switching delay	2 µs max. off to on 10 µs max. on to off
Output behavior in STOP	Last value or substitute value (default value 0)
Number of outputs on simultaneously	2
Cable length (max.), in meters	Shielded: 500 m normal inputs Unshielded: 150 m normal inputs

Table A-104 Wiring diagram for the SB DT04 2 Digital Input/2 Digital Output (6ES7288-5DT04-0AA0)

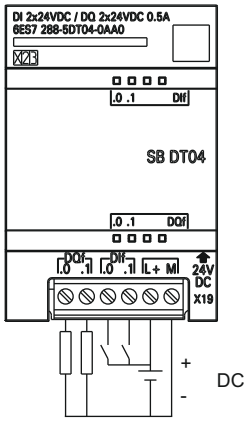
SB DT04 2 Digital Input/2 Digital Output (6ES7288-5DT04-0AA0)	
	

Table A-105 Connector pin locations for SB DT04 2 Digital Input/2 Digital Output (6ES7288-5DT04-0AA0)

Pin	X19
1	DQ f.0
2	DQ f.1
3	DI f.0
4	DI f.1
5	L+ / 24 V DC
6	M / 24 V DC

A.7 Analog signal boards

A.7.1 SB AE01 analog input specifications

Table A-106 General specifications

Technical data	SB Analog 1 x Input (SB AE01)
Article number	6ES7288-5AE01-0AA0
Dimensions W x H x D (mm)	35 x 52.2 x 16
Weight	20 grams
Power dissipation	0.4 W
Current consumption (5 V DC)	50 mA (5 V and 3.3 combined)
Current consumption (24 V DC)	None

Table A-107 Analog inputs

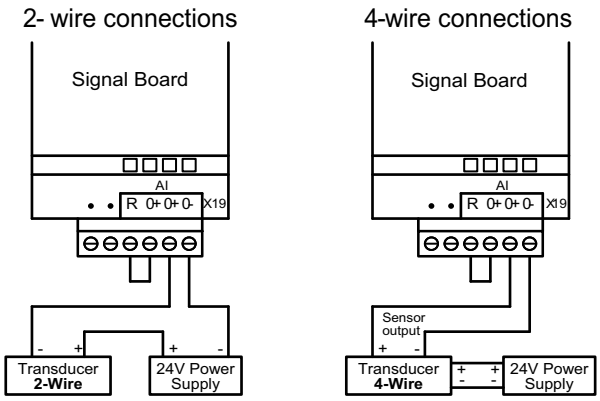
Technical data		SB Analog 1 x input (SB AE01)
Number of inputs		1
Type		Voltage or current (differential)
Range		±10 V, ±5 V, ±2.5 V, or 0 to 20 mA
Resolution		11 bits + sign bit (voltage mode) 11 bits (current mode)
Full scale range (data word)		-27,648 to 27,648
Accuracy (25 °C / -20 to 60 °C)		Voltage mode: ±0.3 % / ±0.6 % of full scale Current mode: ±0.3 % / ±0.6 % of full scale
Overshoot/undershoot range (data word)		Voltage: 27,649 to 32,511 / -27,649 to -32,512 Current: 27,649 to 32,511 / -4864 to 0 (Refer to Analog input representation for voltage and Analog input representation for current (Page 1026).)
Overflow/underflow (data word)		Voltage: 32,512 to 32,767 / -32,513 to -32,768 Current: 32,512 to 32,767 / -4,865 to -32,768 (Refer to Analog input representation for voltage and Analog input representation for current (Page 1026).)
Maximum withstand voltage / current		±35 V / ±40 mA
Smoothing		None, weak, medium, or strong (Refer to Analog input response times for step response time (Page 1025).)
Noise rejection		400, 60, 50, or 10 Hz
Measuring principle common mode rejection		40 dB, DC to 60 Hz rejection
Operational signal range (signal plus common mode voltage)		Signal plus common mode voltage must be less than +35 V and greater than -35 V
Input impedance		
	Differential mode	220 KΩ (voltage) / 250 Ω (current)
	Common mode	55 KΩ (voltage) / 55 Ω (current)
Isolation (field side to logic)		None
Cable length (meters)		100 m twisted and shielded

Table A-108 Diagnostics

Model	SB Analog 1 x input (SB AE01)
Overflow/underflow	Yes
24 V DC low voltage	None

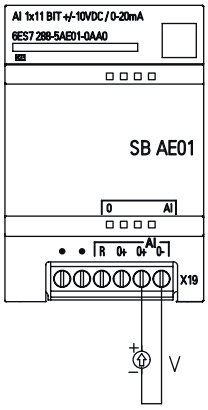
SB AE01 wiring current transducers

Wiring current transducers are available as 2-wire transducers and 4-wire transducers as shown below.



SB AE01 wiring voltage transducers

Wiring voltage transducers is shown below.



Note: Connectors must be gold. See Appendix F, Spare parts and other hardware for article number.

Table A-109 Connector pin locations for SB AE01 Analog Input 1 x Input (6ES7288-5AE01-0AA0)

Pin	X19
1	No connection
2	No connection
3	AI R
4	AI 0+
5	AI 0+
6	AI 0-

A.7.2 SB AQ01 analog output specifications

Table A-110 General specifications

Technical data	SB Analog 1 x Output (SB AQ01)
Article number	6ES7288-5AQ01-0AA0
Dimensions W x H x D (mm)	35 x 52.2 x 16
Weight	17.4 grams
Power dissipation	1.5 W
Current consumption (5 V DC)	15 mA
Current consumption (24 V DC)	40 mA (no load)

Table A-111 Analog outputs

Technical data	SB Analog 1 x Output (SB AQ01)
Number of outputs	1
Type	Voltage or current
Range	±10 V, 0 to 20 mA
Resolution	Voltage: 11 bits + sign Current: 11 bits
Full scale range (data word) Refer to the output ranges for voltage and current.	-27,648 to 27,648 (-10 V to 10 V) 0 to 27,648 (0 to 20 mA)
Accuracy (25 °C / -20 to 60 °C)	±0.5% / ±1%
Settling time (95% of new value)	Voltage: 300 µs (R), 750 µs (1 µF) Current: 600 µs (1mH), 2 ms (10 mH)
Load impedance	Voltage: ≥ 1000 ohms Current: ≤ 600 ohms
Output behavior in STOP	Last value, substitute value (default value 0)
Isolation (field side to logic)	None
Cable length (max.), in meters	10 m twisted and shielded

Table A-112 Diagnostics

Technical data	SB Analog 1 x Output (SB AQ01)
Overflow/underflow	Yes
Short to ground (voltage mode only)	Yes
Wire break (current mode only)	Yes

Table A-113 Wiring diagram for the SB AQ01 Analog 1 x Output (6ES7288-5AQ01-0AA0)

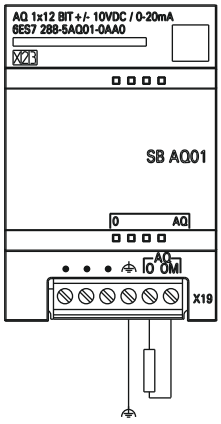
SB AQ01 Analog 1 x Output (6ES7288-5AQ01-0AA0)	
	

Table A-114 Connector pin locations for SB AQ01 Analog 1 x Output (6ES7288-5AQ01-0AA0)

Pin	X19
1	No connection
2	No connection
3	No connection
4	Functional Earth
5	AQ 0
6	AQ 0M

A.8 RS485/RS232 signal boards

A.8.1 SB RS485/RS232 specifications

Table A-115 General specifications

Technical data	SB RS485/RS232
Article number	6ES7288-5CM01-0AA0
Dimensions W x H x D (mm)	35 x 52.2 x 16
Weight	18.2 grams
Power dissipation	0.5 W
Current consumption (5 V DC)	50 mA
Current consumption (24 V DC)	N/A

Table A-116 RS485 Transmitter and receiver

Technical data	SB RS485/RS232
Common mode voltage range	-7 V to +12 V, 1 second, 3 VRMS continuous
Transmitter differential output voltage	2 V min. at RL = 100 Ω 1.5 V min. at RL = 54 Ω
Termination and bias	4.7 K Ω to +5 V on TXD 4.7 K Ω to GND on RXD
Receiver input impedance	12 K Ω min.
Receiver threshold/sensitivity	+/- 0.2 V min. 60 mV typical hysteresis
Isolation RS 485 signal to chassis ground RS485 signal to CPU logic common	None
Cable length, shielded	With isolated repeater: 1000 m up to 187.5 Kbps Without isolated repeater: 50 m

Table A-117 RS232 Transmitter and receiver

Technical data	SB RS485/RS232
Transmitter output voltage	+/-5 V min. at RL = 3K Ω
Transmit output voltage	+/- 15 V DC max.
Receiver input impedance	3 K Ω min.
Receiver threshold/sensitivity	0.8 V min. low, 2.4 max. high 0.5 V typical hysteresis
Receiver input voltage	+/- 30 V DC max.
Isolation RS232 signal to chassis ground RS232 signal to CPU logic common	None
Cable length, shielded	10 m max.

Table A-118 Wiring diagram for the SB CM01 RS485/RS232 (6ES7288-5CM01-0AA0)

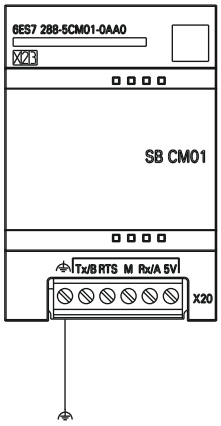
SB CM01 RS485/RS232 (6ES7288-5CM01-0AA0)	
	

Table A-119 Connector pin locations for SB CM01 RS485/RS232 (6ES7288-5CM01-0AA0)

Pin	X20
1	Functional Earth
2	Tx/B
3	RTS
4	M
5	Rx/A
6	5 V Out (Bias Voltage)

A.9 Battery board signal boards (SBs)

A.9.1 SB BA01 Battery board

SB BA01 Battery board

The S7-200 SMART SB BA01 battery board provides for long-term backup of the real-time clock. The battery board plugs into the signal board slot of the S7-200 SMART CPU (firmware V2.0 and later versions). You must add the SB BA01 to the device configuration and download the hardware configuration to the CPU for the SB BA01 to gain access to the additional battery health reporting options.

When you purchase the SB BA01 battery board, it does not include the battery (type CR1025). You must purchase the battery separately.

Note

The SB BA01 is mechanically designed to fit the CPUs with the firmware V2.0 and later versions.

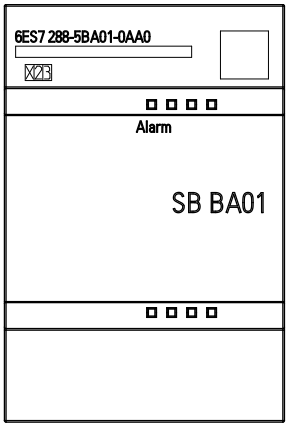
Table A-120 General specifications

Technical data	SB BA01 Battery Board
Article number	6ES7288-5BA01-0AA0
Dimensions W x H x D (mm)	35 x 52.2 x 16
Weight	20 grams
Power dissipation	0.6 W
Current consumption (5 V DC)	18 mA
Current consumption (24 V DC)	None

Battery (not included)	SB BA01 Battery Board
Hold up time	Approximately 1 year
Battery type	CR1025 Refer to Installing or replacing a battery in the SB BA01 battery board (Page 93) Note: Verify whether the battery operating temperature conforms to module operating temperature requirement.
Nominal voltage	3 V
Nominal capacity	30 mAH

Diagnostics	SB BA01 Battery Board
Critical battery level	< 2.5 V
Battery diagnostic	Low voltage indicator: • Low battery voltage causes the LED on the BA01 panel to illuminate with the red light continuously ON. • Diagnostic alarm and/or digital output status of battery low condition available
Battery status	Battery status bit provided 0 = Battery OK 1 = Battery low
Battery status update	Battery status is updated at power up and then once per day while CPU is in RUN mode.

Table A-121 Wiring diagram for the SB BA01 Battery board (6ES7288-5BA01-0AA0)

SB BA01 Battery board (6ES7288-5BA01-0AA0)	
	

A.10 EM DP01 PROFIBUS DP module

Table A-122 General specifications

Technical data		EM DP01 PROFIBUS DP
Article number		6ES7288-7DP01-0AA0
Dimensions W x H x D (mm)		70 x 100 x 81
Weight		176.2 g
Power dissipation		1.5 W (no load)
V DC requirements		
	+5 V DC (SM Bus)	150 mA (no load)
	+24 V DC	See below

Table A-123 EM features

Technical data	EM DP01 PROFIBUS DP module
Number of ports (limited power)	1
Electrical interface	RS485
PROFIBUS DP / MPI baud rates (set automatically)	9.6, 19.2, 45.45, 93.75, 187.5, and 500 Kbps; 1.5, 3, 6, and 12 Mbps
Protocols	PROFIBUS DP slave and MPI slave
Cable length	
Up to 93.7 Kbps	1200 m
187.5 Kbps	1000 m
500 Kbps	400 m
1 to 1.5 Mbps	200 m

Technical data	EM DP01 PROFIBUS DP module
3 to 12 Mbps	100 m
Network capabilities	
Station address settings	0 to 99 (set by rotary switches)
Maximum stations per segment	32
Maximum stations per network	126, up to 99 EM DP01 stations
MPI connections	6 total (1 reserved for an OP)

Table A-124 Power supply

Technical data	EM DP01 PROFIBUS DP
24 V DC input power requirements	
Voltage range	20.4 to 28.8 V DC (Class 2, Limited Power, or sensor power from PLC)
Maximum current: Module only with port active Add 90 mA of 5 V port load Add 120 mA of 24 V port load	 30 mA 60 mA 180 mA
Ripple noise (< 10 MHz)	< 1 V peak to peak (maximum)
Isolation (field to logic) [1]	500 V AC for 1 minute
5 V DC power on communications port	
Maximum current per port	900 mA @ nominal 5 V
Current limit	2.7 A @ 5 V
Isolation (5 V DC to logic)	500 V AC for 1 minute
24 V DC power on communications port	
Voltage range	20.4 to 28.8 V DC
Maximum current per port	120 mA @ nominal 24 V
Current limit	0.7 to 2.4 A
Isolation	Not isolated, same circuit as input 24 V DC

[1] No power is supplied to module logic by the 24 V DC supply. 24 V DC supplies power for the communications port.

A.10.1 S7-200 SMART CPUs that support the EM DP01 PROFIBUS DP module

The S7-200 SMART EM DP01 PROFIBUS DP module is an intelligent expansion module designed to work with the S7-200 SMART CPUs, firmware version 2.1 or later, shown in the following table.

Table A-125 EM DP01 PROFIBUS DP module compatibility with S7-200 SMART CPUs, firmware version 2.1 or later

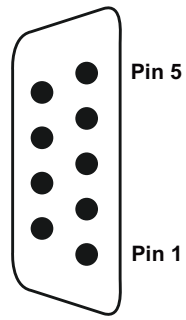
CPU	Description
ST20	CPU ST20 (DC/DC/DC)
SR20	CPU SR20 (AC/DC/Relay)

CPU	Description
ST30	CPU ST30 (DC/DC/DC)
SR30	CPU SR30 (AC/DC/Relay)
ST40	CPU ST40 (DC/DC/DC)
SR40	CPU SR40 (AC/DC/Relay)
ST60	CPU ST60 (DC/DC/DC)
SR60	CPU SR60 (AC/DC/Relay)

A.10.2 Connector pin assignments for EM DP01

The RS485 communication serial port on the EM DP01 is RS485-compatible on a nine-pin subminiature D female connector, in accordance with the PROFIBUS standard as defined in the European Standard EN 50170. The following table shows the connector that provides physical connection for the communication port and describes the communication port pin assignments.

Table A-126 Pin assignments for the S7-200 SMART EM DP01

Pin Number	Connector	PROFIBUS
1		Shield
2		24 V Return
3		RS485 Signal B
4		Request-to-Send
5		5 V Return
6		+5 V (isolated)
7		+24 V
8		RS485 Signal A
9		NC

A.10.3 EM DP01 PROFIBUS DP module wiring diagram

Table A-127 Wiring diagram for the EM DP01 PROFIBUS DP module (6ES7288-7DP01-0AA0

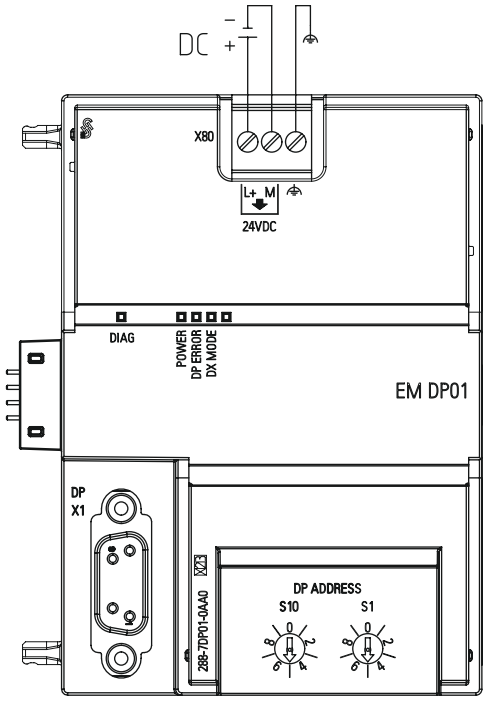
EM DP01 PROFIBUS DP module (6ES7288-7DP01-0AA0)	
	

Table A-128 Connector pin locations for EM DP01 PROFIBUS DP module (6ES7288-7DP01-0AA0)

Pin	X80
1	L+ / 24 V DC
2	M / 24 V DC
3	Functional Earth

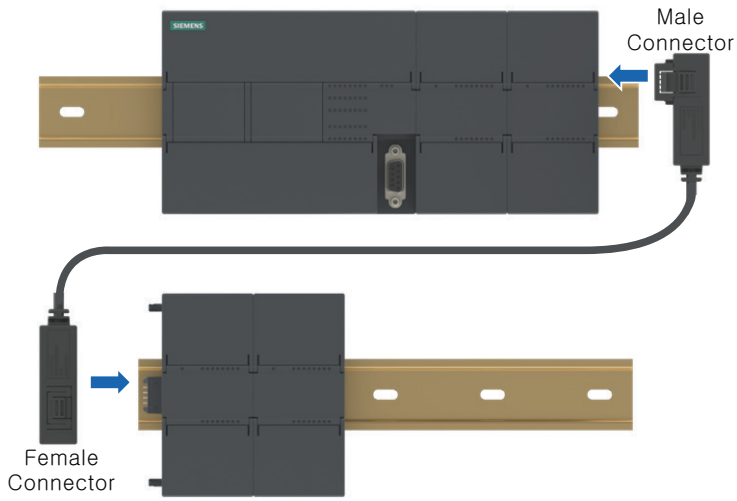
A.11 S7-200 SMART cables

A.11.1 S7-200 SMART I/O expansion cable

Table A-129 S7-200 SMART Expansion cable

Technical Data	
Article number	6ES7288-6EC01-0AA0
Cable length	1 m
Weight	80 g

Refer to the installation section (Page 99) for information about installing and removing the S7-200 SMART expansion cable.



A.11.2 RS-232/PPI Multi-Master Cable and USB/PPI Multi-Master Cable

A.11.2.1 Overview

RS-232/PPI Multi-Master Cable and USB/PPI Multi-Master Cable Specifications

Table A-130 General Specifications

Technical Data	RS-232/PPI Multi-Master cable	USB/PPI Multi-Master cable
Article number	6ES7901-3CB30-0XA0	6ES7901-3DB30-0XA0
Supply voltage	14.4 to 28.8 VDC	14.4 to 28.8 VDC
Supply current at 24 V nominal supply	60 mA RMS max.	50 mA RMS max.
Isolation	RS-485 to RS-232: 500 VDC	RS-485 to USB: 500 VDC

RS-485 Side Electrical Characteristics		
Common mode voltage range	-7 V to +12 V, 1 second, 3 V RMS continuous	-7 V to +12 V, 1 second, 3 V RMS continuous
Receiver input impedance	5.4 kΩ min. including termination	5.4 kΩ min. including termination
Termination/bias	10 kΩ to +5 V on B, PROFIBUS pin 3 10 kΩ to GND on A, PROFIBUS pin 8	10 kΩ to +5 V on B, PROFIBUS pin 3 10 kΩ to GND on A, PROFIBUS pin 8
Receiver threshold/sensitivity	±0.2 V, 60 mV typical hysteresis	±0.2 V, 60 mV typical hysteresis
Transmitter differential output voltage	2 V min. at R_L = 100 Ω, 1.5 V min. at R_L = 54 Ω	2 V min. at R_L = 100 Ω, 1.5 V min. at R_L =54 Ω

RS-232 Side Electrical Characteristics		
Receiver input impedance	3 kΩ min.	-
Receiver threshold/sensitivity	0.8 V min. low, 2.4 V max. high 0.5 V typical hysteresis	-
Transmitter output voltage	± 5 V min. at R_L = 3 kΩ	-

USB Side Electrical Characteristics		
Full speed (12 MB/s), Human Interface Device (HID)		
Supply current at 5 V	-	50 mA max.
Power down current	-	400 µA max.

Features

You can use the RS232/PPI Multi-Master cable with the S7-200 SMART CPUs in Freeport mode. See the Freeport mode for more information.

The USB cable requires the STEP 7-Micro/WIN SMART V2.3 (or later) programming software for operation.

A.11.2.2 RS-232/PPI Multi-Master Cable

Table A-131 RS-232/PPI Multi-Master Cable - Pin-outs for RS-485 to RS-232 Local Mode Connector

RS-485 Connector Pin-out		RS-232 Local Connector Pin-out	
Pin Number	**Signal description**	**Pin number**	**Signal Description**
1	No connect	1	Data Carrier Detect (DCD) (not used)
2	24 V Return (RS-485 logic ground)	2	Receive Data (RD) (output from PC/PPI cable)
3	Signal B (RxD/TxD+)	3	Transmit Data (TD) (input to PC/PPI cable)
4	RTS (TTL level)	4	Data Terminal Ready (DTR)[1]
5	No connect	5	Ground (RS-232 logic ground)
6	No connect	6	Data Set Ready (DSR) [1]
7	24 V Supply	7	Request To Send (RTS) (not used)
8	Signal A (RxD/TxD-)	8	Clear To Send (CTS) (not used)
9	Protocol select	9	Ring Indicator (RI) (not used)

[1] Pins 4 and 6 are connected internally.

Table A-132 RS-232/PPI Multi-Master Cable - Pin-outs for RS-485 to RS-232 Remote Mode Connector

RS-485 Connector Pin-out		RS-232 Remote Connector Pin-out [1]	
Pin Number	**Signal Description**	**Pin Number**	**Signal Description**
1	No connect	1	Data Carrier Detect (DCD) (not used)
2	24 V Return (RS-485 logic ground)	2	Receive Data (RD) (input to PC/ PPI cable)
3	Signal B (RxD/TxD+)	3	Transmit Data (TD) (output from PC/PPI cable)
4	RTS (TTL level)	4	Data Terminal Ready (DTR) [2]
5	No connect	5	Ground (RS-232 logic ground)
6	No connect	6	Data Set Ready (DSR) [2]
7	24 V Supply	7	Request To Send (RTS) (output from PC/PPI cable)
8	Signal A (RxD/TxD-)	8	Clear To Send (CTS) (not used)
9	Protocol select	9	Ring Indicator (RI) (not used)

[1] A conversion from female to male, and a conversion from 9-pin to 25-pin is required for modems.

[2] Pins 4 and 6 are connected internally.

Use the RS-232/PPI Multi-Master Cable for Freeport operation

For connection directly to your personal computer:

- Set the PPI/Freeport mode (Switch 5=0)
- Set the baud rate (Switches 1, 2, and 3)
- Set Local (Switch 6=0). The Local setting is the same as setting the PC/PPI cable to DCE.
- Set the Protocol select to 10 Bit (Switch 7=1). See Note below.

For connection to a modem:

- Set the PPI/Freeport mode (Switch 5=0)
- Set the baud rate (Switches 1, 2, and 3)
- Set Remote (Switch 6=1). The Remote setting is the same as setting the PC/PPI cable to DTE.
- Set the Protocol select to 10 Bit (Switch 7 = 1). See Note below.

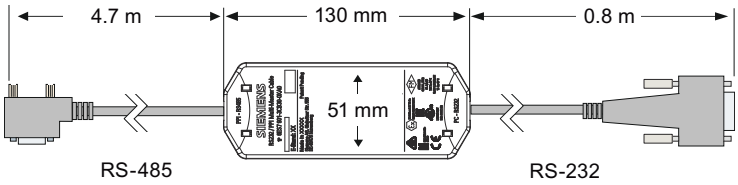
Note

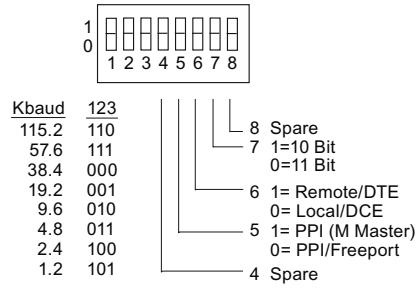
The CRxxs CPUs require the protocol select switch 7 to be set to 1 to allow Freeport mode. Setting switch 7 to 0 disables Freeport mode in the CRxxs CPUs.

The SR and ST CPUs ignore the select switch 7 setting.

RS-232/PPI Multi-Master cable dimensions, label and LEDs

The following figure shows the RS-232/PPI Multi-Master Cable dimensions, label and LEDs.





LED	Color	Description
Tx	Green	RS-232 transmit indicator
Rx	Green	RS-232 receive indicator
PPI	Green	RS-485 transmit indicator

A.11.2.3 USB/PPI Multi-Master Cable

To use the USB cable, you must have STEP 7-Micro/WIN SMART V2.3 (or later) installed. It is recommended that you use the USB cable only with S7-200 SMART CPUs firmware version V2.3 (or later).

The USB cable does not support Freeport communications.

Note

If you use this cable with S7-200 SMART CPU versions prior to V2.3, certain operations are not possible (for example, downloading the user program).

Note

Attaching a USB-PPI cable to the CPU's RS485 port forces the CPU to exit Freeport mode and enable PPI mode. This allows STEP 7-Micro/WIN SMART V2.3 to regain control of the the CPU.

Table A-133 USB/PPI Multi-Master cable - Pin-outs for the RS-485 to USB Series "A" Connector

RS-485 Connector Pin-out		USB Connector Pin-out	
Pin Number	**Signal Description**	**Pin Number**	**Signal Description**
1	No connect	1	USB -DataP
2	24 V Return (RS-485 logic ground)	2	USB -DataM
3	Signal B (RxD/TxD+)	3	USB 5 V
4	RTS (TTL level)	4	USB logic ground
5	No connect	-	-
6	No connect	-	-
7	24 V Supply	-	-
8	Signal A (RxD/TxD-)	-	-
9	Protocol select	-	-

The following figure shows theUSB/PPI Multi-Master Cable dimensions and LEDs.

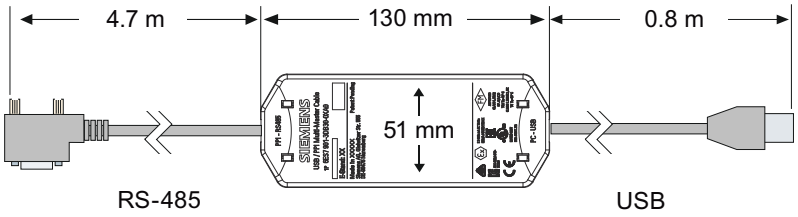


Figure A-1 USB/PPI Multi-Master Cable Dimensions and LEDs

LED	Color	Description
Tx	Green	USB transmit indicator
Rx	Green	USB receive indicator
PPI	Green	RS-485 transmit indicator

Calculating a power budget

B

B.1 Power budget

Your CPU has an internal power supply that provides power for the CPU, the expansion modules, signal boards, and other 24 V DC user power requirements. Use the following information as a guide for determining how much power (or current) the CPU can provide for your configuration.

Refer to the technical specifications for your particular CPU (Page 986) to determine the 24 V DC sensor supply power budget, the 5 V DC logic budget supplied by your CPU and the 5 V DC power requirements of the expansion modules and signal boards. Refer to the Calculating a power budget to determine how much power (or current) the CPU can provide for your configuration.

The standard CPU provides the 5 V DC logic power needed for any expansion in your system. Pay careful attention to your system configuration to ensure that the CPU can supply the 5 V DC power required by your selected expansion modules. If your configuration requires more power than the CPU can supply, you must remove a module.

Note

If the CPU power budget is exceeded, you may not be able to connect the maximum number of modules allowed for your CPU.

The standard CPU also provides a 24 V DC sensor supply that can supply 24 V DC for input points, for relay coil power on the expansion modules, or for other requirements. If your power requirements exceed the budget of the sensor supply, then you must add an external 24 V DC power supply to your system. You must manually connect the 24 V DC supply to the input points or relay coils.

If you require an external 24 V DC power supply, ensure that the power supply is not connected in parallel with the sensor supply of the CPU. For improved electrical noise protection, it is recommended that the commons (M) of the different power supplies be connected.

WARNING

Connecting power supplies safely

Connecting an external 24 V DC power supply in parallel with the 24 V DC sensor supply of the CPU can result in a conflict between the two supplies as each seeks to establish its own preferred output voltage level.

The result of this conflict can be shortened lifetime or immediate failure of one or both power supplies, with consequent unpredictable operation of the PLC system. Unpredictable operation could result in death or serious injury to personnel, and/or damage to equipment.

The DC sensor supply of the CPU and any external power supply should provide power to different points. A single connection of the commons is allowed.

Some of the 24 V DC power input ports in the S7-200 SMART system are interconnected, with a common logic circuit connecting multiple M terminals. For example, the following circuits are interconnected when designated as "not isolated" in the data sheets: the 24 V DC power supply of the CPU, the power input for the relay coil of an EM, or the power supply for a non-isolated analog input. All non-isolated M terminals must connect to the same external reference potential.

WARNING

Avoiding unwanted current flow

Connecting non-isolated M terminals to different reference potentials will cause unintended current flows that may cause damage or unpredictable operation in the PLC and any connected equipment.

Failure to comply with these guidelines could cause damage or unpredictable operation which could result in death or severe personal injury and/or property damage.

Always ensure that all non-isolated M terminals in an S7-200 SMART system are connected to the same reference potential.

Refer to the technical specifications for your particular CPU (Page 986) to determine the 24 V DC sensor supply power budget, the 5 V DC logic budget supplied by your CPU and the 5 V DC power requirements of the expansion modules and signal boards.

B.2 Calculating a sample power requirement

Calculating a sample power requirement

The following table shows a sample calculation of the power requirements for a CPU that includes the following:

- CPU SR40 AC/DC/Relay
- 3 each EM Digital Output 8 x Relay (EM DR08)
- 1 each EM Digital 8 x Inputs (EM DE08)

This installation has a total of 32 inputs and 40 outputs.

Note

The CPU has already allocated the power required to drive the CPUs internal relay coils. You do not need to include the internal relay coil power requirements in a power budget calculation.

The CPU in this example provides sufficient 5 V DC current, but does not provide enough 24 V DC current from the sensor supply for all of the inputs and expansion relay coils. The I/O

requires 392 mA and the CPU provides 300 mA. This installation requires an additional source of at least 92 mA of 24 V DC power to operate all the included 24 V DC inputs and outputs.

Table B-1 Calculation of the power budget for a sample configuration

CPU power budget	**5 V DC**	**24 V DC**
CPU SR40 AC/DC/Relay	1400 mA	300 mA
	minus	
System requirements	**5 V DC**	**24 V DC**
CPU SR40, 24 inputs		24 * 4 mA = 96 mA
Slot 0: EM DR08	120 mA	8 * 11 mA = 88 mA
Slot 1: EM DR08	120 mA	8 * 11 mA = 88 mA
Slot 2: EM DR08	120 mA	8 * 11 mA = 88 mA
Slot 3: EM DE08	105 mA	8 * 4 mA = 32 mA
Total requirements	465 mA	392 mA
	equals	
Current balance	**5 V DC**	**24 V DC**
Current balance total	935 mA	(92 mA)

B.3 Calculating your power requirement

Calculating your power requirement

Use the table below to determine how much power (or current) the CPU can provide for your configuration. Refer to the technical specifications (Page 981) for the power budgets of your CPU model and the power requirements of your digital modules, analog modules or signal boards.

Table B-2 Power budget

Power budget	**5 V DC**	**24 V DC**
	minus	
System requirements	**5 V DC**	**24 V DC**
Total requirements		

Power budget	**5 V DC**	**24 V DC**
	equals	
Current balance	**5 V DC**	**24 V DC**
Current balance total		

Error codes

C

C.1 Timestamp mismatch

This warning message indicates that the timestamps for the project do not match the timestamps for the program in the PLC. This may indicate that the programs are different, in which case it would be dangerous to continue the current operation. However, the programs might be functionally identical and still have different timestamps.

What actions modify the program timestamps?

Each program contains two distinct timestamps; the "Created" timestamp and the "Last Modified" timestamp. The created timestamp is set when the project is created by the New Project option. The Created timestamp is not affected by any user edits or program compilation.

The Last Modified timestamp is used to indicate when the user last modified the program. There are many conditions that cause the Last Modified timestamp to be set:

1. An edit of instructions or operands in the program block editor.
2. Adding, deleting, or modifying a global or local variable.
3. Adding or deleting a POU.
4. Compiling the program block.
5. Downloading the program block (this automatically compiles the program block and therefore sets the last modified timestamp).

Note that although all of these actions will cause the last modified timestamp to be set, this does not necessarily mean that the programs are different. For this reason, STEP 7-Micro/WIN SMART V3 provides the "Compare" option, to allow you to determine whether the programs are really different

How do I tell if the programs are really different?

You can choose to compare the program block in the PLC with the project's program block by clicking the "Compare" button. The results of this comparison allow you to determine whether to continue the status operation.

How do I synchronize timestamps?

Downloading a new project to the PLC synchronizes the timestamps, which allows you to run status.

C.2 PLC non-fatal error codes

PLC compiler and run-time errors are non-fatal errors. Non-fatal errors can degrade some aspect of the performance of your PLC, but do not render the PLC incapable of executing the user program or updating the I/O.

- **Run-time programming errors** are non-fatal error conditions created by your program while the program is being executed. An example of this is an indirect-address pointer, which was valid when the program compiled, and then modified by program execution to point to an out-of-range address. Access the PLC Information from the PLC menu ribbon strip to determine what type of error has occurred.
  You can correct run-time programming errors only by modifying the user program. The CPU clears the run-time programming errors at the next transition from STOP to RUN mode.
- **PLC compiler errors** (or program-compile errors) prevent you from downloading the program to the PLC. STEP 7-Micro/WIN SMART V3 detects compile errors when you compile or download (Page 82) the program, and shows errors in the output window. If there is a compile error, the PLC retains the current program that is resident in the PLC.

I/O errors are also non-fatal errors. When problems occur with the I/O of the CPU, signal board, and expansion modules, the PLC records the error information in special memory (SM) bits (Page 1067) that your program can monitor and evaluate.

Non-fatal error codes

Hexadecimal error code	Non-fatal PLC program compiler errors
0080	The program is too large for the CPU; please reduce the program size
0081	Logic stack underflow; split the network into multiple networks
0082	Illegal instruction; check instruction mnemonics
0083	Illegal instruction before end of main program; remove incorrect instruction
0085	Illegal combination of FOR/NEXT; add FOR instruction or delete NEXT instruction
0086	Illegal combination of FOR/NEXT; add NEXT instruction or delete FOR instruction
0087	Missing label or POU; add the appropriate label
0088	Illegal instruction before end of subroutine; add RET to the end of the subroutine or remove incorrect instruction
0089	Illegal instruction before end of interrupt routine; add RETI to the end of the interrupt routine or remove incorrect instruction
008B	Illegal jump in or out of a SCR segment
008C	Duplicate label or POU name
008D	Exceeded maximum label or POU number; ensure that the number of labels allowed has not been exceeded
0090	Illegal operand
0091	Memory range error; check the operand ranges
0092	Illegal count operand; verify the maximum count size
0093	FOR/NEXT nesting level exceeded
0095	Missing LSCR instruction
0096	Missing SCRE instruction or illegal instruction before SCRE
0099	Too many password-protected POUs

Hexadecimal error code	Non-fatal PLC program compiler errors
009B	Illegal index for a string operation
009D	Illegal parameter detected in system block
009F	Illegal program organization

Hexadecimal error code	Transition to RUN mode prevented (run inhibit conditions)
0070	Run inhibit due to memory card inserted
0071	Run inhibit due to missing configured device
0072	Run inhibit due to mismatched device configuration (note: this error also includes device parameterization errors)
0073	Run inhibit due to attempted firmware update
0074	Run inhibit due to serious HW error on expansion module or signal board

Hexadecimal error code	Non-fatal run-time programming problem
0000	No non-fatal errors present
0001	HSC instruction enabled before executing HDEF instruction
0002	Input interrupt point already assigned to an HSC
0003	HSC input point already assigned to an input interrupt or other HSC
0004	Instruction not allowed in an interrupt routine
0005	Simultaneous HSC/PLS/motion instructions
0006	Indirect addressing error
0007	Time of day instructions data error
0008	Maximum user subroutine nesting level exceeded
0009	Simultaneous XMT/RCV instructions on Port 0
000A	Execution of an HDEF instruction of a previously configured HSC
000B	Simultaneous execution of XMT/RCV instructions on Port 1
000D	Attempt to redefine pulse output while it is active
000E	Number of PTO profile segment was set to 0
000F	Illegal numeric value encountered in compare contact instruction
0010	Command is not allowed in current PTO mode of operation
0011	Illegal PTO command code
0012	Illegal PTO profile table
0013	Illegal PID loop table
0014	Data log error: • There are too many DATx_WRITE subroutine executions in one program scan. Only 10 to 15 data log writes per second can be sustained. When there are too many DATx-WRITE executions per second, then the allotted memory will be full and for a short period of time no new data log records are stored. • Executing a data log write subroutine without first configuring a data log with the data log wizard

Hexadecimal error code	Non-fatal run-time programming problem
0016	HSC or interrupt input point already assigned to motion
0017	PTO/PWM output point already assigned to motion
0019	Signal Board not present or not configured
001A	Scan watchdog timeout.
001B	Attempt to change time base on enabled PWM
001C	Serious hardware error on expansion module or signal board
001D	HSC pin map invalid
001E	Clock A and Clock B are not in the same pin group
001F	Simultaneous execution of XMT/RCV on port 2
0090	Illegal operand
0091	Operand range error; check the operand ranges
0092	Illegal count operand; verify the maximum count size
0098	Illegal program edit in RUN mode
009A	Attempt to switch into freeport mode inside a user interrupt routine
009B	Illegal index for a string operation (user requested index=0)
00A0	Web application static resource error
00A1	Web OSS static resource error
00A2	Calling an axes group without SDB0
00B0	Security UMLOG file error
00B1	Security User manager error

C.3 PLC non-fatal error SM flags

Overview

Non-fatal errors are those that may degrade some aspect of PLC performance, but do not render the PLC incapable of executing the user program and updating I/O. To help you in debugging your program, information associated with error conditions is stored in Special Memory (SM) locations, which can be accessed by the user program. For example, if you do not want to continue in RUN mode with certain non-fatal error conditions, you can have the user program force a transition to STOP mode when the undesirable condition occurs.

The following table lists and describes the Special Memory non-fatal error information.

SM bit	Non-fatal error description	SM byte	Non-fatal error description
SM0.2	Retentive Data Lost	SMB6	CPU's I/O Error Byte
SM0.7	RTC_Lost	SMB9	Module 0 I/O Error Byte
SM1.3	Divide by Zero Error	SMB11	Module 1 I/O Error Byte
SM3.0	Parity Error	SMB13	Module 2 I/O Error Byte
SM4.0	Comm. Interrupt Queue Overflow	SMB15	Module 3 I/O Error Byte
SM4.1	Input Interrupt Queue Overflow	SMB17	Module 4 I/O Error Byte
SM4.2	Timed Interrupt Queue Overflow	SMB19	Module 5 I/O Error Byte

SM bit	Non-fatal error description	SM byte	Non-fatal error description
SM4.3	Run-Time Programming Problem	SMB132	Module 6 I/O Error Byte
SM5.0	I/O Error (any I/O error bit set)	SMB134	Module 7 I/O Error Byte
		SMB29	Signal Board 0 I/O Error Byte
		SMB33	Signal Board 1 I/O Error Byte

C.4 PLC fatal error codes

Overview

Fatal errors cause the PLC to stop the execution of your program. Depending on the severity of the error, a fatal error can render the PLC incapable of performing any or all functions. The objective for handling fatal errors is to bring the PLC to a safe state from which the PLC can respond to interrogations about the existing error conditions.

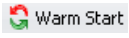
The PLC performs the following tasks when a fatal error is detected.

- Changes to STOP mode
- Turns on both the System Fault LED and the STOP LED
- Turns off the outputs

The PLC remains in this condition until the fatal error is corrected. The table shown below provides a list with descriptions for the fatal error codes that can be read from the PLC.

STEP 7-Micro/WIN SMART displays the error codes generated by the PLC, along with a brief description, in the PLC Information dialog. To access PLC information, click the PLC button

from the Information area of the PLC menu ribbon strip.

Once you have corrected the conditions that caused the fatal error, power-cycle the PLC or perform a warm restart from STEP 7-Micro/WIN SMART. To perform a warm start, click the Warm Start button 

in the Modify area of the PLC menu ribbon strip.

Restarting the PLC clears the fatal error condition and causes power-up diagnostic testing. If another fatal error condition occurs, the PLC sets the System Fault LED again; otherwise, the PLC begins normal operation.

There are several possible error conditions that can render the PLC incapable of communication, in which case you cannot view the PLC error code. This type of error indicates a hardware failure requiring the PLC module to be repaired; it cannot be fixed by changes to the program or by clearing the PLC memory.

Fatal error codes

Hexadecimal error code	Description
0000	No fatal errors present
0001	System firmware checksum error
0002	Compiled user program checksum error
0004	Permanent memory failed

Hexadecimal error code	Description
0005	Permanent memory error on user program
0006	Permanent memory error on system block
0007	Permanent memory error on force data
0009	Permanent memory error on user data, DB1
000A	Memory card failed
000B	Memory card error on user program
000C	Memory card error on system block
000D	Memory card error on force data
000F	Memory card error on user data, DB1
0010	Internal firmware error
0015	User program has compile error on power-up
0016	User data has compile error on power-up
0017	System block has compile error on power-up
0018	CPU HW identification data not available or corrupted
0019	HW watchdog timeout error
001A	Scan watchdog timeout
0020	PN SDB checksum error in internal flash

C.5 PROFINET alarm

Overview

PROFINET alarms indicates the errors during PROFINET communication. To help you in debugging your program, information associated with error conditions is stored in Special Memory (SM) locations (Page 1067), which can be accessed by the user program.

The following table lists and describes the PROFINET alarm.

USI	Alarm data	Description
0x1	0x1	Device is running with error.

Special memory (SM) and system symbol names

D

D.1 SM (Special Memory) overview

The S7-200 SMART CPU provides special memory that contains system data. SMW is the prefix that indicates a special memory word. SMB is the prefix that indicates a special memory byte. You address individual bits as SM<byte number>.<bit number>. The System defined variable table in STEP 7-Micro/WIN SMART V3 displays the special memory.

SMB0 to SMB29, SMB32 to SMB33, SMB250 to SMB349, SMB480 to SMB515, and SMB950 to SMB1999 (S7-200 SMART read-only special memory)

The S7-200 SMART CPU writes new changes to the system data stored in special memory.

Read system status from the CPU

SMB0 to SMB29, SMB32 to SMB33, SMB250 to SMB349, SMB480 to SMB515, and SMB950 to SMB1999 are read-only from your program. If a program includes logic to write to a read-only SM address, STEP 7-Micro/WIN SMART V3 compiles the program without error. The CPU program compiler, however, will reject the program and display "Operand range error, Download failed".

Your program can read data stored in special memory addresses, evaluate the current system status, and use conditional logic to decide how to respond. In run mode, the continuous scanning of your program logic provides continuous monitoring of system data.

- SMB0 (Page 1069) System status bits
- SMB1 (Page 1070) Instruction execution status bits
- SMB2 (Page 1070) Freeport receive character
- SMB3 (Page 1071) Freeport parity error
- SMB4 (Page 1071) Interrupt queue overflow, run-time program error, interrupts enabled, freeport transmitter idle, and forced value
- SMB5 (Page 1072) I/O error status bits
- SMB6-SMB7 (Page 1073) CPU ID, error status, and digital I/O points
- SMB8-SMB19 (Page 1073) I/O module ID and errors
- SMW22-SMW26 (Page 1075) Scan times
- SMB28-SMB29 (Page 1075) Signal board 0 ID and errors
- SMB32-SMB33 (Page 1075) Signal board 1 ID and errors
- SMB220-SMB235 (Page 1093) PID control error code
- SMB250-SMB349 (Page 1100) EM (expansion module) hardware/firmware ID
- SMB480-SMB515 (Page 1094) Data log status (read only)

- SMB950-SMB999 (Page 1099) SB (signal board 1) hardware/firmware ID
- SMB1000-SMB1049 (Page 1099) CPU hardware/firmware ID
- SMB1050-SMB1099 (Page 1099) SB (signal board 0) hardware/firmware ID
- SMB1100-SMB1399 (Page 1100) EM (expansion module) hardware/firmware ID
- SMB1400-SMB1799 (Page 1103) EM (expansion module) module-specific data
- SMB1800-SMB1939 (Page 1104) PROFINET device status
- SMB1940-SMB1946 (Page 1104) Web server status

SMB30 to SMB194 and SMB566 to SMB749 (S7-200 SMART read/write special memory)

The S7-200 SMART CPU performs these actions:

- Reads configuration/control data from special memory
- Writes new changes to the system data in special memory.

Read system status from CPU

Write (send) control commands to the CPU

Your program can read and write all SM addresses in this range. The normal usage of SM data varies according to the function of each address.

SM addresses provide a means to access system status data, configure system options, and control system functions. In run mode, continuous scanning of your program provides for continuous access to special system features.

- SMB30 (Port 0), SMB130 (Port 1) and SMB31 (port 2) (Page 1076) Port configuration for the integrated RS485 port (Port 0), the CM01 Signal Board (SB) RS232/RS485 port (Port 1) and port (port 2)
- SMB34-SMB35 (Page 1078) Time intervals for timed interrupts
- SMB195-SMB206 (Page 1078) HSC type and frequency measurement sampling period
- SMB36-SMB45 (HSC0), SMB46-SMB55 (HSC1), SMB56-SM65 (HSC2), SMB136-SMB145 (HSC3), SMB146-SMB155 (HSC4), SMB156-SMB165 (HSC5), SMB360-SMB369 (HSC6) and SMB370-SMB379 (HSC7) (Page 1078):High-speed counter configuration and operation
- SMB66-SMB85 (Page 1080) PLS0 and PLS1 high-speed outputs
- SMB86-SMB94 and SMB186-SMB194 (Page 1087) Receive message control
- SMW98 (Page 1091) I/O expansion bus communication errors
- SMW100-SMW120 (Page 1091) System alarms
- SMB130 (Page 1076) Port configuration for the CM01 Signal Board (SB) RS232/RS485 port (Port 1) (See SMB30)
- SMB166-SMB169 (Page 1080) PTO0 profile definition table
- SMB176-SMB179 (Page 1080) PTO1 profile definition table

- SMB186-SMB194 (Page 1087) Receive message control (See SMB86-SMB94)
- SMB195-SMB206 (Page 1092) HSC Type and Frequency measurement sampling period
- SMB380-SMB404 (Page 1093) HSC User-defined pin mapping enable
- SMB566-SMB575 (Page 1080) PLS2 high-speed output
- SMB576-SMB579 (Page 1080) PTO2 profile definition table
- SMB580-SMB589 (Page 1080) PLS3 high-speed output
- SMB590-SMB593 (Page 1080) PTO3 profile definition table
- SMB600-SMB649 (Page 1094) Axis 0 open loop motion control
- SMB650-SMB699 (Page 1096) Axis 1 open loop motion control
- SMB700-SMB749 (Page 1097) Axis 2 open loop motion control
- SMB750-SMB799 (Page 1097) Axis 3 open loop motion control
- SMB800-SMB858 (Page 1097) Axes group 0 open loop motion control
- SMB870-SMB928 (Page 1098) Axes group 1 open loop motion control

D.2 SMB0: System status

Special Memory Byte 0 (SM0.0 - SM0.7) provides eight bits the S7-200 SMART CPU updates at the end of each scan cycle.

Table D-1 SMB0 system status bits

S7-200 SMART variable name	SM address	Description
Always_On	SM0.0	This bit is always TRUE.
First_Scan_On	SM0.1	The CPU sets this bit to TRUE, for the first scan cycle, and sets it to FALSE thereafter. One use for this bit is to call an initialization subroutine.
Retentive_Lost	SM0.2	The CPU sets this bit TRUE for one scan cycle after: • Reset to factory communication command • Reset to factory memory card evaluation • Evaluation of program transfer card in which a new system block was loaded from the card. • Problem with retentive record that the CPU stored on the last power down. This bit can be used as either an error memory bit or as a mechanism to invoke a special start-up sequence.
RUN_Power_Up	SM0.3	The CPU sets this bit to TRUE for one scan cycle when RUN mode is entered from a power-up or warm restart condition. This bit can be used to provide machine warm-up time before starting an operation.
Clock_60s	SM0.4	This bit provides a clock pulse. The bit is FALSE for 30 seconds and TRUE for 30 seconds, for a cycle time of one minute. This bit provides an easy-to-use delay or a one-minute clock pulse.
Clock_1s	SM0.5	This bit provides a clock pulse. The bit is FALSE for 0.5 seconds and then TRUE for 0.5 seconds for a cycle time of one second. This bit provides an easy-to-use delay or a one-second clock pulse.

S7-200 SMART variable name	SM address	Description
Clock_Scan	SM0.6	This bit is a scan cycle clock that is TRUE for one scan and then FALSE for the next scan. On subsequent scans the bit alternates between TRUE and FALSE. You can use this bit as a scan counter input.
RTC_Lost	SM0.7	This bit applies to CPU models that have a real-time clock. The CPU sets this bit to TRUE for one scan cycle if the time on the real time clock device was reset or lost at power-up. The program can use this bit as either an error memory bit or to invoke a special start-up sequence.

D.3 SMB1: Instruction execution status

Special memory byte 1 (SM1.0 - SM1.7) provides execution status for various instructions, such as table and math operations. These bits are set and reset by instructions at execution time.

Table D-2 SMB1 instruction execution status bits

S7-200 SMART variable name	SM address	Description
Result_0	SM1.0	Certain instructions set this bit to TRUE when the result of the operation is zero.
Overflow_Illegal	SM1.1	Certain instructions set this bit to TRUE when either an overflow results or when the instruction detects an illegal number value.
Neg_Result	SM1.2	Math operations set this bit TRUE when the operation produces a negative result.
Divide_By_0	SM1.3	The CPU sets this bit TRUE when the program attempts a division by zero.
Table_Overflow	SM1.4	The Add to Table (ATT) instruction sets this bit TRUE when the referenced data table is full.
Table_Empty	SM1.5	The CPU sets this bit TRUE when either LIFO or FIFO instructions attempt to read from an empty table.
Not_BCD	SM1.6	The CPU sets this bit TRUE for an illegal value (non-BCD) in a BCD to binary conversion.
Not_Hex	SM1.7	The CPU sets this bit TRUE for an illegal value (non-hex ASCII digit) during ASCII to Hex (ATH) conversion.

D.4 SMB2: Freeport receive character

Special memory byte 2 is the Freeport receive character buffer. While in Freeport mode, the CPU stores each character that the CPU receives in this byte for easy access by your program.

Table D-3 SMB2 Freeport received character

S7-200 SMART variable name	SM address	Description
Receive_Char	SMB2	This byte contains each character that the CPU receives from Port 0, Port 1 or Port 2 during Freeport communication.

Note

Port 0, Port 1 and Port 2 share SMB2 and SMB3

When the CPU receives a character on Port 0, note the following:

- The CPU executes the interrupt routine attached to that event (interrupt event 8).
- SMB2 contains the character received on Port 0.
- SMB3 contains the parity status of the received character.

Likewise, when the CPU receives a character on Port 1, note the following:

- The CPU executes the interrupt routine attached to that event (interrupt event 25).
- SMB2 contains the character received on Port 1.
- SMB3 contains the parity status of the received character.

Likewise, when the CPU receives a character on Port 2, note the following:

- The CPU executes the interrupt routine attached to that event (interrupt event 47).
- SMB2 contains the character received on Port 2.
- SMB3 contains the parity status of the received character.

D.5 SMB3: Freeport character error

The CPU sets SM3.0 TRUE when Freeport communication detects a parity, framing, break, or overrun error on a received character. Use this bit to discard the message.

Table D-4 SMB3 Freeport character error

S7-200 SMART variable name	SM address	Description
Parity_Err	SM3.0	This bit indicates that the CPU received a parity, framing, break, or overrun error from Port 0 or Port 1: • FALSE: no error • TRUE: error

D.6 SMB4: Interrupt queue overflow, run-time program error, interrupts enabled, freeport transmitter idle, and value forced

Special memory byte 4 (SM4.0 - SM4.7) contains the interrupt queue overflow bits. These bits indicate either that interrupts are occurring at a rate greater than the CPU can process or that the global interrupt disable (DISI) instruction (Page 393) has disabled interrupts.

Other bits indicate:

- Enabled or disabled status of interrupts
- A run-time program error

- Freeport transmitter status
- One or more forced PLC memory values

Table D-5 SMB4 system status

S7-200 SMART variable name	SM address	Description
Comm_Int_Ovr	** SM4.0	TRUE: Communication interrupt queue has overflowed.
Input_Int_Ovr	** SM4.1	TRUE: Input interrupt queue has overflowed.
Timed_Int_Ovr	** SM4.2	TRUE: Timed interrupt queue has overflowed.
RUN_Err	SM4.3	TRUE: CPU has detected a run-time programming non-fatal error.
Int_Enable	SM4.4	TRUE: Enabled interrupts exists
Xmit0_Idle	SM4.5	TRUE: Port 0 transmitter is idle (FALSE: Transmission in progress).
Xmit1_Idle	SM4.6	TRUE: Port 1 transmitter is idle (FALSE: Transmission in progress).
Force_On	SM4.7	TRUE: Existence of forced PLC memory

** Use status bits SM4.0, SM4.1, and SM4.2 only inside an interrupt routine. The CPU resets these status bits when the CPU empties the interrupt queue and returns control to the main program.

D.7 SMB5: I/O error status

Special Memory Byte 5 (SM5.0 - SM5.7) contains a status bit that indicates error conditions in the I/O system.

Table D-6 SMB5 I/O error status

S7-200 SMART variable name	SM address	Description
IO_Err	SM5.0	This bit is set ON if any I/O errors are present.

D.8 SMB6-SMB7: CPU ID, error status, and digital I/O points

Special memory bytes 6 and 7 provide CPU information.

S7-200 SMART variable name	SM address	Read-only SMB6 and SMB7 (CPU ID, error status, and digital I/O point)								
CPU_ID	SMB6	MSB							LSB	
		7							0	
		1	x	x	x	c	d	0	0	
	SM6.4 to SM6.6		0	0	0	= CPU CR20s				
			0	0	1	= CPU CR40s				
			0	1	0	= CPU CR60s				
			0	1	1	= CPU SR20 / ST20				
			1	0	0	= CPU SR40 / ST40				
			1	0	1	= CPU SR60 / ST60				
			1	1	0	= CPU CR30s				
			1	1	1	= CPU SR30 / ST30				
	SM6.2 to SM6.3					c				Configuration / parameterization error 0 = no error, 1 = error
							d			Diagnostic alarm (See SMW100 for alarm codes) 0 = no error, 1 = error
CPU_IO	SMB7	MSB							LSB	
		7							0	
		i	i	i	i	q	q	q	q	
	SM7.0 to SM7.7	i	i	i	i					0 to 15 bytes of digital input points
						q	q	q	q	0 to 15 bytes of digital output points

See also SMW100-SMW114 System alarm codes (Page 1091)

D.9 SMB8-SMB19, SMB131-SMB134: I/O module ID and errors

SMB8 through SMB19 and SMB131 to SMB134 are organized in byte pairs for expansion modules 0 to 7.

The even-numbered byte of each pair is the module-identification register. These bytes identify the module type, the I/O type, and the number of inputs and outputs.

The odd-numbered byte of each pair is the module error register. These bytes provide an indication of any errors detected in the I/O for that module.

S7-200 SMART variable name	SM address	Read-only SMB8-SMB21 module ID and error data
EM0_ID	SMB8	Expansion module 0 ID register
EM0_Err	SMB9	Expansion module 0 error register (See SMW104 for diagnostic alarm code)

EM1_ID	SMB10	Expansion module 1 ID register
EM1_Err	SMB11	Expansion module 1 error register (See SMW106 for diagnostic alarm code)
EM2_ID	SMB12	Expansion module 2 ID register
EM2_Err	SMB13	Expansion module 2 error register (See SMW108 for diagnostic alarm code)
EM3_ID	SMB14	Expansion module 3 ID register
EM3_Err	SMB15	Expansion module 3 error register (See SMW110 for diagnostic alarm code)
EM4_ID	SMB16	Expansion module 4 ID register
EM4_Err	SMB17	Expansion module 4 error register (See SMW112 for diagnostic alarm code)
EM5_ID	SMB18	Expansion module 5 ID register
EM5_Err	SMB19	Expansion module 5 error register (See SMW114 for diagnostic alarm code)
EM6_ID	SMB131	Expansion module 6 ID register
EM6_Err	SMB132	Expansion module 6 error register (See SMW116 for diagnostic alarm code)
EM7_ID	SMB133	Expansion module 7 ID register
EM7_Err	SMB134	Expansion module 7 error register (See SMW118 for diagnostic alarm code)

	Even numbered byte-I/O module ID register									Odd numbered byte-I/O module error register									
	MSB								LSB		MSB								LSB
	7							0			7							0	
	m	0	0	a	i	i	q	q			c	d	0	b	0	0	0	m	
m: Module present	0	= Present								c:	0	No error							
	1	= Not present									1	Configuration / parameterization error							
										d:		0	No error						
												1	Diagnostic alarm						
a: I/O type				0	= Digital														
				1	= Analog					b:				0	No error				
ii: inputs					0	0	= No inputs							1	Bus access error				
					0	1	= 2 AI or 8 DI												
					1	0	= 4 AI or 16 DI												
					1	1	= 8 AI or 32 DI												
qq: Outputs							0	0	= No outputs	m:								0	OK
							0	1	= 2 AQ or 8 DQ									1	Configured module missing
							1	0	= 4 AQ or 16 DQ										
							1	1	= 8 AQ or 32 DQ										

See also SMW100-SMW114 System alarm codes (Page 1091)

D.10 SMW22-SMW26: Scan times

SMW22, SMW24, and SMW26 contain information on the scan time. You can read the last scan time, minimum scan time, and maximum scan time (millisecond values).

Table D-7 SMW22-SMW26 PLC scan times

S7-200 SMART variable name	SM address	Description
Last_Scan	SMW22	Scan time of the last scan.
Minimum_Scan	SMW24	Minimum scan time recorded since entering the RUN mode or since resetting these values from the PLC Information dialog. After resetting, this value will change to 65535.
Maximum_Scan	SMW26	Maximum scan time recorded since entering the RUN mode or since resetting these values from the PLC Information dialog. After resetting, this value will change to 0.

D.11 SMB28-SMB29, SMB32-SMB33: Signal board ID and errors

SMB28-SMB29 byte addresses store the signal board 0 type and error status.

S7-200 SMART variable name	SM address	Read-only SMB28-SMB29 signal board ID and error data
SB_ID	SMB28	Signal board ID register
SB_Err	SMB29	Signal board error register (See SMW102 for diagnostic alarm code)

	SMB28 signal board ID register									SMB29 signal board error register									
	MSB								LSB		MSB								LSB
	7							0			7							0	
	m	0	0	a	i	i	q	q			c	d	0	b	0	0	0	m	
m: Module present	0	= Present								c:	0	No error							
	1	= Not present									1	Configuration / parameterization error							
										d:		0	No error						
												1	Diagnostic alarm						
a: I/O type				0	= Digital														
				1	= Analog					b:				0	No error				
ii: inputs					0	0	= No inputs							1	Bus access error				
					0	1	= 2 AI or 8 DI												
					1	0	= 4 AI or 16 DI												
					1	1	= 8 AI or 32 DI												

qq: Outputs	0	0	= No outputs	m:	0	OK
	0	1	= 2 AQ or 8 DQ		1	Configured module missing
	1	0	= 4 AQ or 16 DQ			
	1	1	= 8 AQ or 32 DQ			

SMB32-SMB33 byte addresses store the signal board 1 type and error status.

S7-200 SMART variable name	SM address	Read-only SMB32-SMB33 signal board ID and error data
SB_ID	SMB32	Signal board ID register
SB_Err	SMB33	Signal board error register (See SMW102 for diagnostic alarm code)

	SMB32 signal board ID register									SMB33 signal board error register									
	MSB								LSB	MSB									LSB
	7							0			7							0	
	m	0	0	a	i	i	q	q			c	d	0	b	0	0	0	m	
m: Module present	0	= Present								c:	0	No error							
	1	= Not present									1	Configuration / parameterization error							
										d:		0	No error						
												1	Diagnostic alarm						
a: I/O type				0	= Digital														
				1	= Analog					b:				0	No error				
ii: inputs					0	0	= No inputs							1	Bus access error				
					0	1	= 2 AI or 8 DI												
					1	0	= 4 AI or 16 DI												
					1	1	= 8 AI or 32 DI												
qq: Outputs							0	0	= No outputs	m:								0	OK
							0	1	= 2 AQ or 8 DQ									1	Configured module missing
							1	0	= 4 AQ or 16 DQ										
							1	1	= 8 AQ or 32 DQ										

See also SMW100-SMW110 System alarm codes (Page 1091)

D.12 SMB30: (Port 0), SMB130: (Port1) and SMB31: (Port 2)

SMB30 configures Port 0 (onboard RS485 port).

SMB130 configures Port 1 (optional CM01 signal board).

SMB31 configures Port 2 (optional CM01 signal board).

You can read and write to SMB30, SMB31 and SMB130. These bytes configure the respective communication port for Freeport operation and provide selection of either Freeport or system protocol support.

S7-200 SMART variable name	**SM address**			Bit format	**Bit format**								
					MSB								LSB
	Port 0	**Port 1**	**Port 2**		7							0	
P0_Config	**SMB30**				p	p	d	b	b	b	m	m	
P1_Config		**SMB130**											
P2_Config			**SMB31**										
	SM30.6 - SM30.7	SM130.6 - SM130.7	SM31.6 - SM31.7	pp:	0	0	= No parity						
					0	1	= Even parity						
					1	0	= No parity						
					1	1	= Odd parity						
	SM30.5	SM130.5	SM31.5	d:			0	= 8 data bits per character					
							1	= 7 data bits per character					
	SM30.2 - SM30.4	SM130.2 - SM130.4	SM31.2 - SM31.4	bbb:				0	0	0	= 38,400 bps		
								0	0	1	= 19,200 bps		
								0	1	0	= 9,600 bps		
								0	1	1	= 4,800 bps		
								1	0	0	= 2,400 bps		
								1	0	1	= 1,200 bps		
								1	1	0	= 115,200 bps		
								1	1	1	= 57,600 bps		
P0_Config_0	SM30.0			mm:							0	0	= PPI slave mode
P1_Config_1		SM130.0											
P1_Config_2			SM31.0								0	1	= Freeport protocol
	SM30.1	SM130.1	SM31.1								1	0	= Reserved (defaults to PPI slave mode)
											1	1	= Reserved (defaults to PPI slave mode)
											Note: Bits 2 through 7 are ignored in PPI modes		

D.13 SMB34-SMB35: Time intervals for timed interrupts

Special memory bytes 34 and 35 control the time interval of timed interrupts 0 and 1 (Page 395). You can assign time intervals in 1-ms increments from 1 ms to 255 ms. The CPU captures the time-interval value at the time that the CPU attaches the interrupt routine to corresponding timed interrupt event. To change the time interval, you must reattach the timed interrupt event to the same interrupt routine or to a different interrupt routine. You can terminate a timed interrupt event by detaching the event.

Table D-8 SMB34-SMB35 timed interrupt intervals

S7-200 SMART variable name	SM address	Description
Time_0_Intrvl	SMB34	Timed interrupt 0: Time interval value (in 1 ms increments from 1 ms to 255 ms).
Time_1_Intrvl	SMB35	Timed interrupt 1: Time interval value (in 1 ms increments from 1 ms to 255 ms).

D.14 SMB36-SMB45 (HSC0), SMB46-SMB55 (HSC1), SMB56-SMB65 (HSC2), SMB136-SMB145 (HSC3), SMB146-SMB155 (HSC4), SMB156-SMB165 (HSC5), SMB360-SMB369 (HSC6), SMB370-SMB379 (HSC7): high-speed counters

These bytes provide configuration and operation information for the high-speed counters:

- HSC0
- HSC1
- HSC2
- HSC3
- HSC4
- HSC5
- HSC6
- HSC7

SM definition	S7-200 SMART variable name	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7
HSC* counter status **Note**: Counter status bits are valid only while the CPU is executing an interrupt routine that a high-speed counter event triggered.	**HSC*_Status**								
Current counting direction status bit: • 0 = Counting down • 1 = Counting up	HSC*_Status_5	SM36.5	SM46.5	SM56.5	SM136.5	SM146.5	SM156.5	SM360.5	SM370.5

D.14 SMB36-SMB45 (HSC0), SMB46-SMB55 (HSC1), SMB56-SMB65 (HSC2), SMB136-SMB145 (HSC3), SMB146-SMB155 (HSC4), SMB156-SMB165 (HSC5), SMB360-SMB369 (HSC6), SMB370-SMB379 (HSC7): high-speed counters

SM definition	S7-200 SMART variable name	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7
Current value equals preset value status bit: • 0 = Unequal • 1 = Equal	HSC*_Status_6	SM36.6	SM46.6	SM56.6	SM136.6	SM146.6	SM156.6	SM360.6	SM370.6
Current value is greater than preset value status bit: • 0 = Less than or equal to • 1 = Greater than	HSC*_Status_7	SM36.7	SM46.7	SM56.7	SM136.7	SM146.7	SM156.7	SM360.7	SM370.7
HSC0 counter control	**HSC*_Ctrl**								
Active level control bit for Reset: • 0 = Reset is active high • 1 = Reset is active low	HSC*_Reset_Level	SM37.0	SM47.0	SM57.0	SM137.0	SM147.0	SM157.0	SM361.0	SM371.0
Counting rate selection for Quadrature counters: • 0 = 4x counting rate • 1 = 1x counting rate	HSC*_Rate	SM37.2	SM47.2	SM57.2	SM137.2	SM147.2	SM157.2	SM361.2	SM371.2
Direction control bit: • 0 = Counting down • 1 = Counting up	HSC*_Dir	SM37.3	SM47.3	SM57.3	SM137.3	SM147.3	SM157.3	SM361.3	SM371.3
Update direction: • 0 = No update direction • 1 = Update direction	HSC*_Dir_Update	SM37.4	SM47.4	SM57.4	SM137.4	SM147.4	SM157.4	SM361.4	SM371.4
Update preset value: • 0 = Not Write new preset value to HSC* preset • 1: Write new preset value to HSC* preset	HSC*_PV_Update	SM37.5	SM47.5	SM57.5	SM137.5	SM147.5	SM157.5	SM361.5	SM371.5
Update current value: • 0 = Not Write new preset value to HSC* current • 1: Write new preset value to HSC* current	HSC*_CV_Update	SM37.6	SM47.6	SM57.6	SM137.6	SM147.6	SM157.6	SM361.6	SM371.6
Enable bit: • 0 = disable • 1 = enable	HSC*_Enable	SM37.7	SM47.7	SM57.7	SM137.7	SM147.7	SM157.7	SM361.7	SM371.7

SM definition	S7-200 SMART variable name	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7
HSC* new current value For example, you use SMD38 to set HSC0 current value to any value you choose. To update the current value, write the new current value to SMD38; write 1 to SM37.6; and execute the HSC instruction. The instruction then writes the new current value to HSC0's current count register.	HSC*_CV	SMD38	SMD48	SMD58	SMD138	SMD148	SMD158	SMD362	SMD372
HSC* new preset value For example, you use SMD42 to set HSC0 preset value to any value you choose. To update the preset value, write the new preset value to SMD42; write 1 to SM37.5; and execute the HSC instruction. The instruction then writes the new preset value to HSC0's preset register.	HSC*_PV	SMD42	SMD52	SMD62	SMD142	SMD152	SMD162	SMD366	SMD376

See also

High-speed counter programming (Page 330)

D.15 SMB66-SMB85 (PTO0/PWM0, PTO1/PWM1), SMB166-SMB169 (PTO0), SMB176-SMB179 (PTO1), SMB566-SMB579 (PTO2/PWM2), SMB580-SMB593 (PTO3/PWM3), SMB2000-SMB2013 (PTO4/PWM4): high-speed outputs

The S7-200 SMART CPU uses the following special memory to monitor and control the pulse train outputs (PTO0 and PTO1) and pulse width modulation outputs (PWM0 and PWM1) for the PLS (Pulse) instruction:

The CPU uses SMB566-SMB579 to monitor and control pulse train output PTO2 and pulse width modulation output PWM2.

The CPU uses SMB580-SMB593 to monitor and control pulse train output PTO3 and pulse width modulation output PWM3.

D.15 SMB66-SMB85 (PTO0/PWM0, PTO1/PWM1), SMB166-SMB169 (PTO0), SMB176-SMB179 (PTO1), SMB566-SMB579 (PTO2/PWM2), SMB580-SMB593 (PTO3/PWM3), SMB2000-SMB2013 (PTO4/PWM4): high-speed outputs

The CPU uses SMB2000-SMB2013 to monitor and control pulse train output PTO4 and pulse width modulation output PWM4.

Table D-9 High-speed output 0 configuration and control

S7-200 SMART variable name	SM address	Function
PTO0_Status	**SMB66**	**PTO0 Status**
PLS0_Abort_AE	SM66.4	PTO0 profile was aborted due to an add error: FALSE: No abort; TRUE: aborted
PLS0_Disable_UC	SM66.5	PTO0 user manually disabled a PTO profile while it was running: • FALSE: Not disabled • TRUE: Manually disabled
PLS0_Ovr	SM66.6	PTO0 pipeline overflow/underflow, loading pipeline while full or transferring an empty pipeline: • FALSE: No overflow • TRUE: Pipeline overflow/underflow
PLS0_Idle	SM66.7	PTO0 idle: • FALSE: PTO in progress • TRUE: PTO is idle
PLS0_Ctrl	**SMB67**	**Monitor and control PTO0 (Pulse Train Output) and PWM0 (Pulse Width Modulation) for Q0.0**
PLS0_Cycle_Update	SM67.0	PTO0/PWM0 update the cycle time or frequency value: • FALSE: No update • TRUE: Write new cycle time/frequency
PWM0_PW_Update	SM67.1	PWM0 update the pulse width value: • FALSE: No update • TRUE: Write new pulse width
PTO0_PC_Update	SM67.2	PTO0 update the pulse count value: • FALSE: No update • TRUE: Write new pulse count
PWM0_TimeBase	SM67.3	PWM0 time base: • FALSE: 1 µs/tick • TRUE: 1 ms/tick
PTO0_Override	SM67.4	PTO0 override: • FALSE:Disable • TRUE: Enable
PTO0_Operation	SM67.5	PTO0 select single/multiple segment operation: • FALSE: Single • TRUE:Multiple
PLS0_Select	SM67.6	PTO0/PWM0 mode select: • FALSE: PWM • TRUE: PTO

S7-200 SMART variable name	SM address	Function
PLS0_Enable	SM67.7	PTO0/PWM0 enable: • FALSE: Disable • TRUE: Enable
PLS0_Cycle	**SMW68**	**Word data type: PWM0 cycle time value (2 to 65,535 units of time base); PTO0 frequency value (1 to 65,535 Hz)**
PWM0_PW	**SMW70**	**Word data type: PWM0 pulse width value (0 to 65,535 units of time base)**
PTO0_PC	**SMD72**	**Double Word data type: PTO0 pulse count value (1 to 2,147,483,647)**
PTO0_Seg_Num	**SMB166**	**Byte data type: The currently executing segment number for PTO0's profile**
PTO0_Profile_Offset	**SMW168**	**Word data type: Starting location of PTO0's profile table (byte offset from V0)**

Table D-10 High-speed output 1 configuration and control

S7-200 SMART variable name	SM address	Function
PTO1_Status	**SMB76**	**PTO1 Status**
PLS1_Abort_AE	SM76.4	PTO1 profile was aborted due to an add error: • FALSE: No abort • TRUE: Aborted
PLS1_Disable_UC	SM76.5	PTO1 user manually disabled a PTO profile while it was running: • FALSE: Not disabled • TRUE: Manually disabled
PLS1_Ovr	SM76.6	PTO1 pipeline overflow/underflow, loading pipeline while full or transferring an empty pipeline: • FALSE: No overflow • TRUE: Pipeline overflow/underflow
PLS1_Idle	SM76.7	PTO1 idle: • FALSE: PTO in progress • TRUE: PTO is idle
PLS1_Ctrl	**SMB77**	**Monitor and control PTO1 (Pulse Train Output) and PWM1 (Pulse Width Modulation) for Q0.1**
PLS1_Cycle_Update	SM77.0	PTO1/PWM1 update the cycle time or frequency value: • FALSE: No update • TRUE: Write new cycle time/frequency
PWM1_PW_Update	SM77.1	PWM1 update the pulse width value: • FALSE: No update • TRUE: Write new pulse width
PTO1_PC_Update	SM77.2	PTO1 update the pulse count value: • FALSE: No update • TRUE: Write new pulse count

S7-200 SMART variable name	SM address	Function
PWM1_TimeBase	SM77.3	PWM1 time base: • FALSE: 1 µs/tick • TRUE: 1 ms/tick
PTO1_Override	SM77.4	PTO1 override: • FALSE:Disable • TRUE: Enable
PTO1_Operation	SM77.5	PTO1 select single/multiple segment operation: • FALSE: Single • TRUE: Multiple
PLS1_Select	SM77.6	PTO1/PWM1 mode select: • FALSE: PWM • TRUE: PTO
PLS1_Enable	SM77.7	PTO1/PWM1 enable: • FALSE: Disable • TRUE: Enable
PLS1_Cycle	**SMW78**	**Word data type: PWM1 cycle time value (2 to 65,535 units of time base); PTO1 frequency value (1 to 65,535 Hz)**
PWM1_PW	**SMW80**	**Word data type: PWM1 pulse width value (0 to 65,535 units of time base)**
PTO1_PC	**SMD82**	**Double Word data type: PTO1 pulse count value (1 to 2,147,483,647)**
PTO1_Seg_Num	**SMB176**	**Byte data type: The currently executing segment number for PTO1's profile**
PTO1_Profile_Offset	**SMW178**	**Word data type: Starting location of PTO1's profile table (byte offset from V0)**

Table D-11 High-speed output 2 configuration and control

S7-200 SMART variable name	SM address	Description
PTO2_Status	**SMB566**	**PTO2 Status**
PLS2_Abort_AE	SM566.4	PTO2 profile was aborted due to an add error: • FALSE: No abort • TRUE:Aborted
PLS2_Disable_UC	SM566.5	PTO2 user manually disabled a PTO profile while it was running: • FALSE: Not disabled • TRUE: Manually disabled
PLS2_Ovr	SM566.6	PTO2 pipeline overflow/underflow, loading pipeline while full or transferring an empty pipeline: • FALSE: No overflow • TRUE: Pipeline overflow/underflow
PLS2_Idle	SM566.7	PTO2 idle: • FALSE: PTO in progress • TRUE: PTO is idle

S7-200 SMART variable name	SM address	Description
PLS2_Ctrl	**SMB567**	**Monitor and control PTO2 (Pulse Train Output) and PWM2 (Pulse Width Modulation) for Q0.3**
PLS2_Cycle_Update	SM567.0	PTO2/PWM2 update the cycle time or frequency value: • FALSE: No update • TRUE: Write new cycle time/frequency
PWM2_PW_Update	SM567.1	PWM2 update the pulse width value: • FALSE: No update • TRUE: Write new pulse width
PTO2_PC_Update	SM567.2	PTO2 update the pulse count value: • FALSE: No update • TRUE: Write new pulse count
PLS2_TimeBase	SM567.3	PWM2 time base: • FALSE: 1 µs/tick • TRUE: 1 ms/tick
PTO2_Override	SM567.4	PTO2 override: • FALSE:Disable • TRUE: Enable
PTO2_Operation	SM567.5	PTO2 select single/multiple segment operation: • FALSE: Single • TRUE: Multiple
PLS2_Select	SM567.6	PTO2/PWM2 mode select: • FALSE: PWM • TRUE: PTO
PLS2_Enable	SM567.7	PTO2/PWM2 enable: • FALSE: disable • TRUE: enable
PLS2_Cycle	**SMW568**	**Word data type: PWM2 cycle time value (2 to 65,535 units of time base); PTO2 frequency value (1 to 65,535 Hz)**
PWM2_PW	**SMW570**	**Word data type: PWM2 pulse width value (0 to 65,535 units of time base)**
PTO2_PC	**SMD572**	**Double Word data type: PTO2 pulse count value (1 to 2,147,483,647)**
PTO2_Seg_Num	**SMB576**	**Byte data type: The currently executing segment number for PTO2's profile**
PTO2_Profile_Offset	**SMW578**	**Word data type: Starting location of PTO2's profile table (byte offset from V0)**

Table D-12 High-speed output 3 configuration and control

S7-200 SMART variable name	SM address	Description
PTO3_Status	**SMB580**	**PTO3 Status**
PLS3_Abort_AE	SM580.4	PTO3 profile was aborted due to an add error: • FALSE: No abort • TRUE:Aborted
PLS3_Disable_UC	SM580.5	PTO3 user manually disabled a PTO profile while it was running: • FALSE: Not disabled • TRUE: Manually disabled
PLS3_Ovr	SM580.6	PTO3 pipeline overflow/underflow, loading pipeline while full or transferring an empty pipeline: • FALSE: No overflow • TRUE: Pipeline overflow/underflow
PLS3_Idle	SM580.7	PTO3 idle: • FALSE: PTO in progress • TRUE: PTO is idle
PLS3_Ctrl	**SMB581**	**Monitor and control PTO3 (Pulse Train Output) and PWM3 (Pulse Width Modulation) for Q0.2**
PLS3_Cycle_Update	SM581.0	PTO3/PWM3 update the cycle time or frequency value: • FALSE: No update • TRUE: Write new cycle time/frequency
PWM3_PW_Update	SM581.1	PWM3 update the pulse width value: • FALSE: No update • TRUE: Write new pulse width
PTO3_PC_Update	SM581.2	PTO3 update the pulse count value: • FALSE: No update • TRUE: Write new pulse count
PLS3_TimeBase	SM581.3	PWM3 time base: • FALSE: 1 µs/tick • TRUE: 1 ms/tick
PTO3_Override	SM581.4	PTO3 override: • FALSE:Disable • TRUE: Enable
PTO3_Operation	SM581.5	PTO3 select single/multiple segment operation: • FALSE: Single • TRUE: Multiple
PLS3_Select	SM581.6	PTO3/PWM3 mode select: • FALSE: PWM • TRUE: PTO
PLS3_Enable	SM581.7	PTO3/PWM3 enable: • FALSE: disable • TRUE: enable

S7-200 SMART variable name	SM address	Description
PLS3_Cycle	**SMW582**	**Word data type: PWM3 cycle time value (2 to 65,535 units of time base); PTO3 frequency value (1 to 65,535 Hz)**
PWM3_PW	**SMW584**	**Word data type: PWM3 pulse width value (0 to 65,535 units of time base)**
PTO3_PC	**SMD586**	**Double Word data type: PTO3 pulse count value (1 to 2,147,483,647)**
PTO3_Seg_Num	**SMB590**	**Byte data type: The currently executing segment number for PTO3's profile**
PTO3_Profile_Offset	**SMW592**	**Word data type: Starting location of PTO3's profile table (byte offset from V0)**

Table D-13 High-speed output 4 configuration and control

S7-200 SMART variable name	SM address	Description
PTO4_Status	**SMB2000**	**PTO4 Status**
PLS4_Abort_AE	SM2000.4	PTO4 profile was aborted due to an add error: • FALSE: No abort • TRUE:Aborted
PLS4_Disable_UC	SM2000.5	PTO4 user manually disabled a PTO profile while it was running: • FALSE: Not disabled • TRUE: Manually disabled
PLS4_Ovr	SM2000.6	PTO4 pipeline overflow/underflow, loading pipeline while full or transferring an empty pipeline: • FALSE: No overflow • TRUE: Pipeline overflow/underflow
PLS4_Idle	SM2000.7	PTO4 idle: • FALSE: PTO in progress • TRUE: PTO is idle
PLS4_Ctrl	**SMB2001**	**Monitor and control PTO4 (Pulse Train Output) and PWM4 (Pulse Width Modulation) for Q0.4**
PLS4_Cycle_Update	SM2001.0	PTO4/PWM4 update the cycle time or frequency value: • FALSE: No update • TRUE: Write new cycle time/frequency
PWM4_PW_Update	SM2001.1	PWM4 update the pulse width value: • FALSE: No update • TRUE: Write new pulse width
PTO4_PC_Update	SM2001.2	PTO4 update the pulse count value: • FALSE: No update • TRUE: Write new pulse count
PLS4_TimeBase	SM2001.3	PWM4 time base: • FALSE: 1 µs/tick • TRUE: 1 ms/tick

S7-200 SMART variable name	SM address	Description
PTO4_Override	SM2001.4	PTO4 override: • FALSE:Disable • TRUE: Enable
PTO4_Operation	SM2001.5	PTO4 select single/multiple segment operation: • FALSE: Single • TRUE: Multiple
PLS4_Select	SM2001.6	PTO4/PWM4 mode select: • FALSE: PWM • TRUE: PTO
PLS4_Enable	SM2001.7	PTO4/PWM4 enable: • FALSE: disable • TRUE: enable
PLS4_Cycle	**SMW2002**	**Word data type: PWM4 cycle time value (2 to 65,535 units of time base); PTO4 frequency value (1 to 65,535 Hz)**
PWM4_PW	**SMW2004**	**Word data type: PWM4 pulse width value (0 to 65,535 units of time base)**
PTO4_PC	**SMD2006**	**Double Word data type: PTO4 pulse count value (1 to 2,147,483,647)**
PTO4_Seg_Num	**SMB2010**	**Byte data type: The currently executing segment number for PTO4's profile**
PTO4_Profile_Offset	**SMW2012**	**Word data type: Starting location of PTO4's profile table (byte offset from V0)**

D.16 SMB86-SMB94, SMB186-SMB194 and SMB405-SMB413: Receive message control

You use SMB86-SMB94, SMB405-SMB413 and SMB186-SMB194 to control and read status of the RCV (Receive message) instruction (Page 270).

S7-200 SMART variable name	SM address			Bit format	Receive message status byte								
					MSB								LSB
	Port 0	**Port 1**	**Port 2**		7							0	
P0_Stat_Rcv	**SMB86**				n	r	e	0	0	t	c	p	
P1_Stat_Rcv		**SMB186**											
P2_Stat_Rcv			**SMB405**										
P0_Stat_Rcv_7	SM86.7			n:	1	= Receive message was terminated by user disable command else n = 0							
P1_Stat_Rcv_7		SM186.7											
P2_Stat_Rcv_7			SM405.7										

P0_Stat_Rcv_6	SM86.6			r:	1	= Receive message terminated: (No start condition defined, character count of 0, or execution of receive message with transmit active) else r = 0
P1_Stat_Rcv_6		SM186.6				
P2_Stat_Rcv_6			SM405.6			
P0_Stat_Rcv_5	SM86.5			e:	1	= End character received else e = 0
P1_Stat_Rcv_5		SM186.5				
P2_Stat_Rcv_5			SM405.5			
P0_Stat_Rcv_2	SM86.2			t:	1	= Receive message terminated: timer expired else n = 0
P1_Stat_Rcv_2		SM186.2				
P2_Stat_Rcv_2			SM405.2			
P0_Stat_Rcv_1	SM86.1			c:	1	= Receive message terminated: maximum character count reached else c = 0
P1_Stat_Rcv_1		SM186.1				
P2_Stat_Rcv_1			SM405.1			
P0_Stat_Rcv_0	SM86.0			p:	1	= Receive message is terminated due to a parity, framing, break, or overrun error else p = 0
P1_Stat_Rcv_0		SM186.0				
P2_Stat_Rcv_0			SM405.0			

S7-200 SMART variable name	**SM address**			Bit format	**Receive message control byte**								
					MSB							LSB	
	Port 0	**Port 1**	**Port 2**		7							0	
P0_Ctrl_Rcv	**SMB87**				en	sc	ec	il	c/m	Tmr	bk	0	
P1_Ctrl_Rcv		**SMB187**											
P2_Ctrl_Rcv			**SMB406**										
P0_Ctrl_Rcv_7	SM87.7			en:	0	= Receive message function disabled							
P1_Ctrl_Rcv_7		SM187.7			1	= Receive message function enabled							
P2_Ctrl_Rcv_7			SM406.7										
P0_Ctrl_Rcv_6	SM86.6			sc:	0	= Ignore SMB88 or SMB188							
P1_Ctrl_Rcv_6		SM186.6			1	= Use the value of SMB88 or SMB188 to detect start of message							
P2_Ctrl_Rcv_6			SM405.6										

P0_Ctrl_Rcv_5	SM86.5			ec:	0	= Ignore SMB89 or SMB189
P1_Ctrl_Rcv_5		SM186.5			1	= Use the value of SMB89 or SMB189 to detect end of message
P2_Ctrl_Rcv_5			SM405.5			
P0_Ctrl_Rcv_4	SM86.4			il:	0	= Ignore SMB90 or SMB190
P1_Ctrl_Rcv_4		SM186.4			1	= Use the value of SMB89 or SMB189 to detect an idle line condition
P2_Ctrl_Rcv_4			SM405.4			
P0_Ctrl_Rcv_3	SM86.3			c/m:	0	= Timer is an inter-character timer
P1_Ctrl_Rcv_3		SM186.3			1	= Timer is a message timer
P2_Ctrl_Rcv_3			SM405.3			
P0_Ctrl_Rcv_2	SM86.2			Tmr:	0	= Ignore SMW92 or SMW192
P1_Ctrl_Rcv_2		SM186.2			1	= Terminate receiving if the time period in SMW92 or SMW192 is exceeded
P2_Ctrl_Rcv_2			SM405.2			
P0_Ctrl_Rcv_1	SM86.1				0	= Ignore break conditions
P1_Ctrl_Rcv_1		SM186.1			1	= Use break condition as start of message detection
P2_Ctrl_Rcv_1			SM405.1			

You use the bits of the message control byte define the criteria for identifying the message. The message control byte also includes the start of message and end of message criteria. To determine the start of a message, either of two sets of logically ANDed start-of-message criteria must be true and must occur in sequence. "In sequence" means one of the following:

- Start character follows idle line
- Start character follows break

A logical OR of the end-of-message criteria determines the end of a message. The equations for start and end criteria are as follows:

Start of message = (il AND sc) OR (bk AND sc)

End of message = ec OR tmr OR maximum character count reached

S7-200 SMART variable name	**SM address**			Bit format	**Receive message control byte**								
					MSB							LSB	
	Port 0	**Port 1**	**Port 2**		7							0	
P0_Ctrl_Rcv	**SMB87**				en	sc	ec	il	c/m	Tmr	bk	0	
P1_Ctrl_Rcv		**SMB187**											
P2_Ctrl_Rcv			**SMB406**										
Idle line detection						0		1			0		SMW90 > 0
Start character detection						1		0			0		SMW90 = don't care
Break detection						0		0			1		SMW90 = don't care
Any response to a request						0		1			0		SMW90 = 0 (Message timer can be used to terminate receiving, if there is no response)
Break and a start character						1		0			1		SMW90 = don't care
Idle line and start character						1		1			0		SMW90 > 0
Idle line and start character (illegal)						1		1			0		SMW90 = 0
Note: Receive is automatically terminated by a parity, framing, overrun or break error.													

S7-200 SMART variable name	**SM address**			
	Port 0	**Port 1**	**Port 2**	
P0_Stat_Char	**SMB88**			Start of message character
P1_Stat_Char		**SMB188**		
P2_Stat_Char			**SMB407**	
P0_End_Char	SMB89			End of message character
P1_End_Char		SMB189		
P2_End_Char			SMB408	
P0_Idle_Time	SMW90			Word data: Idle line time period given in milliseconds. The first character received after the idle line time has expired is the start of a new message.
P1_Idle_Time		SMW190		
P2_Idle_Time			SMW409	
P0_Timeout	SMW92			Word data: Inter-character/message timer timeout value given in milliseconds. If the time period is exceeded, the receive message is terminated.
P1_Timeout		SMW192		
P2_Timeout			SMW411	
P0_Max_Char	SMB94			Maximum number of characters to be received (1 to 255 bytes). **Note**: This range must be set to the expected maximum buffer size, even if the character count message termination is not used.
P1_Max_Char		SMB194		
P2_Max_Char			SMB413	

D.17 SMW98: Expansion I/O bus communication errors

SMW98 gives you information about errors on the expansion I/O bus.

Table D-14 SMW98 Expansion I/O bus communication error counter

S7-200 SMART variable name	SM address (Read/Write)	Description
EM_Parity_Err	SMW98	The CPU increments this word each time the CPU detects a parity error on the expansion I/O bus. Power cycling the CPU clears the word or writing a zero to the word.

D.18 SMW100-SMW120 System alarms

Special memory words SMW100-SMW120 provide alarm and diagnostic error codes for CPU, SB (signal board), and EM (expansion modules).

S7-200 SMART variable name	SM address	Current diagnostic alarm source
CPU_Alarm	SMW100	CPU diagnostic alarm code
SB_Slot0_Alarm	SMW102	Signal board 0 diagnostic alarm code
EM0_Alarm	SMW104	Expansion module bus slot 0 diagnostic alarm code
EM1_Alarm	SMW106	Expansion module bus slot 1 diagnostic alarm code
EM2_Alarm	SMW108	Expansion module bus slot 2 diagnostic alarm code
EM3_Alarm	SMW110	Expansion module bus slot 3 diagnostic alarm code
EM4_Alarm	SMW112	Expansion module bus slot 4 diagnostic alarm code
EM5_Alarm	SMW114	Expansion module bus slot 5 diagnostic alarm code
EM6_Alarm	SMW116	Expansion module bus slot 6 diagnostic alarm code
EM7_Alarm	SMW118	Expansion module bus slot 7 diagnostic alarm code
SB_Slot1_Alarm	SMW120	Signal board 1 diagnostic alarm code

Alarm code format	MSB															LSB
	15	14	13					8	7							0
	d	s	c	c	c	c	c	c	c	a	a	a	a	a	a	a
d: Alarm direction	0	Input channel or not applicable														
	1	Output channel														
s: Alarm scope		0	On individual channel													
		1	On individual channel													
c: Channel number			c	c	c	c	c	c	Number of the affected channel, if alarm scope = “On individual channel” or 0, if alarm scope = “On entire module”							
a: Alarm type									0	0	0	0	0	0	0	0	00H: No alarm
									0	0	0	0	0	0	0	1	01H: Short circuit
									0	0	0	0	0	x	x	x	02H to 05H: Reserved
									0	0	0	0	0	1	1	0	06H: Wire break

	0	0	0	0	0	1	1	1	07H: Upper limit exceeded
	0	0	0	0	1	0	0	0	08H: Lower limit exceeded
	0	0	0	0	x	x	x	x	09H to 0FH: Reserved
	0	0	0	1	0	0	0	0	10H: Parameterization error
	0	0	0	1	0	0	0	1	11H: Sensor or load voltage missing
	0	0	0	x	x	x	x	x	12H to 1FH: Reserved
	0	0	1	0	0	0	0	0	20H: Internal error (MID problem)
	0	0	1	0	0	0	0	1	21H: Internal error (IID problem)
	0	0	1	0	0	0	1	0	22H: Reserved
	0	0	1	0	0	0	1	1	23H: Configuration error
	0	0	1	0	0	1	0	0	24H: Reserved
	0	0	1	0	0	1	0	1	25H: Bad or missing firmware
	0	0	1	0	x	x	x	x	26H to 2AH: Reserved
	0	0	1	0	1	0	1	1	2BH: Battery voltage low
	x	x	x	x	x	x	x	x	2CH to FFH: Reserved

D.19 SMB195-SMB210: HSC Type and Frequency measurement sampling period

SMB195 through SMB210 are reserved for setting HSC type and frequency measurement sampling period. For information about how your module uses SMB195 through SMB210, refer to High-speed counter programming (Page 330)

Table D-15 Special Memory Bytes SMB195 to SMB210

SM definition	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7
HSC Type: • 0 = Counting mode • 1 = Frequency measurement mode	SMB195	SMB197	SMB199	SMB201	SMB203	SMB205	SMB207	SMB209
HSC frequency measurement sampling period, working in frequency measurement mode: • 0 = Sampling period 1 s • 1 = Sampling period 100 ms • 2 = Sampling period 10 ms	SMB196	SMB198	SMB200	SMB202	SMB204	SMB206	SMB208	SMB210

D.20 SMB220-SMB235: PID control Error code

Special Memory Byte SMB220 - SMB235 contains status bit that indicates PID control error.

Table D-16 SMB220 to SMB 235

SM address	Description
SMB220 - SMB235 contains status bit that indicates PID control error from loop 0 to loop 15, respectively. For example, SMB220 contains the error message for loop 0, SMB221 contains the error message for loop 1, and so on.	• 0- No error • 1 - PID loop control has error

D.21 SMB380-SMB404: HSC User-defined pin mapping enable

SMB380 is for setting whether you want the legacy or new HSC pin mapping. If you enabled the user-defined pin mapping, you must enter the input points for the HSC in the appropriate SMB. For the configuration of SMB381-SMB404 with input points: I0.0-I0.7 corresponds to 16#00 - 16#07, and I1.0-I1.7 corresponds to 16#08 - 16#0F.

SM definition	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7
HSC User-defined pin mapping enable [1] • 0 = user-define pin mapping disabled, use the legacy pin mapping • 1 = user-define pin mappling enabled	SM380.0	SM380.1	SM380.2	SM380.3	SM380.4	SM380.5		
User-defined input points for Clock A [2]	SMB381	SMB384	SMB387	SMB390	SMB393	SMB396	SMB399	SMB402
User-defined input points for Clock Dir/ Clock B [2]	SMB382	SMB385	SMB388	SMB391	SMB394	SMB397	SMB400	SMB403
User-defined input points for Reset [2]	SMB383	SMB386	SMB389	SMB392	SMB395	SMB398	SMB401	SMB404

[1] HSC6 and HSC7 don't have default pin mapping. When you use HSC6 and HSC7, you must set this bit to 1 and you must assign input points for HSC6 and HSC7. Otherwise you get an `NF_HSC_PIN_MAP_INVALID` in PLC information.

[2] For the same channel HSC, Clock A and Direction / Clock B can only be selected simultaneously in IB0 or simultaneously in IB1.

D.22 SMB480-SMB515: Data log status

SMB480 through SMB515 are read-only special memory addresses that indicate the status of Data log operations.

S7-200 SMART variable name	SM address	Function
DL0_InitResult	SMB480	Initialization result code for Data Log 0: After power-up and after a download of the System block, the CPU performs the data log analysis. • 00H: data log OK • 01H: initialization in progress • 02H: data log file not found • 03H: data log initialization error • 04H to FEH: reserved • FFH: data log not configured
DL1_InitResult	SMB481	Initialization result code for Data Log 1 (See SMB 480 for result codes)
DL2_InitResult	SMB482	Initialization result code for Data Log 2 (See SMB 480 for result codes)
DL3_InitResult	SMB483	Initialization result code for Data Log 3 (See SMB 480 for result codes)
DL0_Maximum	SMW500	Data log 0: Configured maximum allowed number of records
DL0_Current	SMW502	Data log 0: Actual maximum allowed number of records
DL1_Maximum	SMW504	Data log 1: Configured maximum allowed number of records
DL1_Current	SMW506	Data log 1: Actual maximum allowed number of records
DL2_Maximum	SMW508	Data log 2: Configured maximum allowed number of records
DL2_Current	SMW510	Data log 2: Actual maximum allowed number of records
DL3_Maximum	SMW512	Data log 3: Configured maximum allowed number of records
DL3_Current	SMW514	Data log 3: Actual maximum allowed number of records

D.23 SMB600-SMB799: Axis (0, 1, 2 and 3) open loop motion control (for Motion(V2.x))

Axis configuration and control SM addresses

Wizard-generated program code reads and writes the axis special memory data.

Axis data SM address				Axis function
Axis 0	Axis 1	Axis 2	Axis 3	
SMB600 - SMB615	SMB650 - SMB665	SMB700 - SMB715	SMB750 - SMB765	Axis name (16 ASCII characters). The first character is the lowest numbered byte in the sequence.
SMB616 - SMB619	SMB666 - SMB669	SMB716 - SMB719	SMB766 - SMB769	Reserved
SMW620	SMW670	SMW720	SMW770	Error code - See Axis of Motion error codes (Page 938)

Axis 0 SMB622	Axis 1 SMB672	Axis 2 SMB722	Axis 3 SMB772		MSB							LSB	
					7	6	5	4	3	2	1	0	
					DIS	0	TRIG	STP	LMT-	LMT+	RPS	ZP	
SM622.7	SM672.7	SM722.7	SM772.7	**DIS**: Disable outputs	0	Not active							
					1	Active							
SM622.5	SM672.5	SM722.5	SM772.5	**TRIG**: Trigger input			0	Not active					
							1	Active					
SM622.4	SM672.4	SM722.4	SM772.4	**STP**: Stop input				0	Not active				
								1	Active				
SM622.3	SM672.3	SM722.3	SM772.3	**LMT-**: Negative travel limit input					0	Not active			
									1	Active			
SM622.2	SM672.2	SM722.2	SM772.2	**LMT+**: Positive travel limit input						0	Not active		
										1	Active		
SM622.1	SM672.1	SM722.1	SM772.1	**RPS**: Reference point switch input							0	Not active	
											1	Active	
SM622.0	SM672.0	SM722.0	SM772.0	**ZP**: Zero pulse input								0	Not active
												1	Active

Axis 0 SMB623	Axis 1 SMB673	Axis 2 SMB723	Axis 3 SMB773		MSB							LSB	
					7	6	5	4	3	2	1	0	
					0	0	0	0	0	OR	R	CFG	
SM623.2	SM673.2	SM723.2	SM773.2	**OR**: Target speed out of range						0	In range		
										1	Out of range		
SM623.1	SM673.1	SM723.1	SM773.1	**R**: Direction of rotation							0	Positive rotation	
											1	Negative rotation	
SM623.0	SM673.0	SM723.0	SM773.0	**CFG**: Axis configured								0	Not configured
												1	Configured

Axis data SM address				Axis function
Axis 0	**Axis 1**	**Axis 2**	**Axis 3**	
SMB624	SMB674	SMB724	SMB774	**CUR_PF** is a byte that indicates which profile the Axis of Motion is currently executing.

Axis data SM address				Axis function
SMB625	SMB675	SMB725	SMB775	**CUR_STP** is a byte that indicates which step in the profile the Axis of Motion is currently executing.
SMD626	SMD676	SMD726	SMD776	**CUR_POS** is a double-word value that indicates the current position of the axis.
SMD630	SMD680	SMD730	SMD780	**CUR_SPD** is a double-word value that indicates the current speed of the axis.

Axis 0 SMB634	**Axis 1 SMB684**	**Axis 2 SMB734**	**Axis 3 SMB784**		**MSB**							**LSB**	
					7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
					D	**Error**							
SM634.7	SM684.7	SM734.7	SM784.7	**D**: Done bit	0	Operation is in progress							
					1	Operation complete (set by the axis during initialization)							
SM634.0-SM634.6	SM684.0-SM684.6	SM734.0-SM734.6	SM784.0-SM784.6	**Error**:		See motion instruction error codes							

Axis data SM address				Axis function
Axis 0	**Axis 1**	**Axis 2**	**Axis 3**	
SMB635-SMB645	SMB685-SMB695	SMB735-SMB745	SMB785-SMB795	Reserved
SMD646	SMD696	SMD746	SMD796	V memory pointer to the configuration/profile table for the axis. A pointer value to an area other than V memory is invalid.

See also

Error codes for the Axis of Motion instruction (seven LS bits of SMB634, SMB684, SMB734 or SMB784)) (Page 939)

D.24 SMB600-SMB649: Axis 0 open loop motion control

Refer to "SMB600-SMB799: Axis (0, 1, 2 and 3) open loop motion control (Page 1094)" for details.

D.25 SMB650-SMB699: Axis 1 open loop motion control

Refer to "SMB600-SMB799: Axis (0, 1, 2 and 3) open loop motion control (Page 1094)" for details.

D.26 SMB700-SMB749: Axis 2 open loop motion control

Refer to "SMB600-SMB799: Axis (0, 1, 2 and 3) open loop motion control (Page 1094)" for details.

D.27 SMB750-SMB799: Axis 3 open loop motion control

Refer to "SMB600-SMB799: Axis (0, 1, 2 and 3) open loop motion control (Page 1094)" for details.

D.28 SMB800-SMB858: Axes group 0 open loop motion control

Wizard-generated program code reads and writes the axes group 0 special memory data.

Table D-17 Axes group 0 SM addresses

SM Bits	Description							
SMB800-SMB815	Axes group 0's name (16 ASCII characters). SMB800 is the first character.							
SMB816-SMB819	Reserved							
SMW820	Error code - See Axes group of Motion error codes (Page 978)							
SMB822	Reserved for instantaneous axes group status.							
SMB823		MSB						LSB
		7	6	5	4	3	2	1	0
		Done	Error code of the axes group of motion instruction. See Axes group of Motion instruction error codes (Page 978).						
	Done	0	Motion on the axes group is in progress.						
		1	Operation completes (set by the axes group during initialization).						
SMB824	Current path							
SMB825	Reserved							
SMB826	Current segment							
SMB827	Reserved							
SMD828	Current composite speed of the axes group							
SMD832	Current position of axis X							
SMD836	Current position of axis Y							
SMD840	Current position of axis Z							
SMD844	Reserved							
SMD848 V memory pointer	V memory pointer to the configuration and path table for the axes group.							
SMB852	Buffer busy flag for axes group							
SMB853	ID of the path under execution or pending for execution							
SMB854	The number of current buffered segment							
SMB855	State of axes group							
SMB856	Axes enabled state							

SM Bits	Description
SMB857	Last completed path ID
SMB858	Last completed segment ID

D.29 SMB870-SMB928: Axes group 1 open loop motion control

Table D-18 Axes group 1 SM addresses

SM Bits	Description								
SMB870-SMB885	Axes group 1's name (16 ASCII characters). SMB870 is the first character.								
SMB886-SMB889	Reserved								
SMW890	Error code - See Axes group of Motion error codes (Page 978)								
SMB892	Reserved for instantaneous axes group status.								
SMB893		MSB							LSB
		7	6	5	4	3	2	1	0
		Done	Error code of the axes group of motion instruction. See Axes group of Motion instruction error codes (Page 978).						
	Done	0	Motion on the axes group is in progress.						
		1	Operation completes (set by the axes group during initialization).						
SMB894	Current path								
SMB895	Reserved								
SMB896	Current segment								
SMB897	Reserved								
SMD898	Current composite speed of the axes group 1								
SMD902	Current position of axis X								
SMD906	Current position of axis Y								
SMD910	Current position of axis Z								
SMD914	Reserved								
SMD918 V memory pointer	V memory pointer to the configuration and path table for the axes group.								
SMB922	Buffer busy flag for axes group								
SMB923	ID of the path under execution or pending for execution								
SMB924	The number of current buffered segment								
SMB925	State of axes group								
SMB926	Axes enabled state								
SMB927	Last completed path ID								
SMB928	Last completed segment ID								

D.30 SMB1000-SMB1049: CPU hardware/firmware ID

This CPU writes the information to special memory after a power-up or warm restart transition. The SMB1000-SMB1049 section of special memory is read-only.

SM address	Description
SMW1000	CPU vendor ID: (always 0x002A)
SMB1002 to SMB1021	CPU order ID (MLFB): ASCII characters, left-justified in field, padded with spaces
SMB1022 to SMB1037	CPU serial number: ASCII characters, left-justified in field, padded with spaces
SMW1038	CPU hardware version: Represents the hardware E-stand; range = 0x0001 to 0xFFFD 0x0000, 0xFFFE, and 0xFFFF are reserved values.
SMD1040	CPU firmware version: • Byte 0: ASCII 'V' • Byte 1: Functional version • Byte 2: Minor change version • Byte 3: Bug fix version
SMW1044	CPU firmware version counter (range 0x0000 to 0x00FF)
SMW1046	Reserved: Always 0x0000
SMW1048	CPU device type: Always 0x0001

D.31 SMB950-SMB998, SMB1050-SMB1099: SB (signal board) hardware/firmware ID

The CPU writes the signal board information to special memory after a power-up or warm restart transition. The SMB950-SMB999 and SMB1050-SMB1099 section of special memory is read-only.

SM address		Description
Signal board 0	Signal board 1	
SMW1050	SMW950	Signal board vendor ID: 0x002A if a Siemens SB is present; 0x0000 if no SB is present
SMB1052 to SMB1071	SMB952 to SMB971	Signal board order ID (MLFB): ASCII characters, left-justified in field, padded with spaces
SMB1072 to SMB1087	SMB972 to SMB987	Signal board serial number: ASCII characters, left-justified in field, padded with spaces
SMW1088	SMW988	Signal board hardware version: Represents the hardware E-stand; range=0x0001 to 0xFFFD 0x0000, 0xFFFE, and 0xFFFF are reserved values.
SMD1090	SMD990	Signal board firmware version: • Byte 0: ASCII 'V' • Byte 1: Functional version • Byte 2: Minor change version • Byte 3: Bug fix version
SMW1094	SMW994	Signal board firmware version counter (range 0x0000 to 0x00FF)

SM address		Description
Signal board 0	Signal board 1	
SMW1096	SMW996	Reserved, always 0x0000
SMW1098	SMW998	Signal board device type: I/O=0x0003, communications=0x0004, all other values reserved

D.32 SMB1100-SMB1399, SMB250-SMB349: EM (expansion module) hardware/firmware ID

The CPU writes expansion module information to special memory after a power-up or warm restart transition. The SMB1100-SMB1399 and SMB250 to SMB349 section of special memory is read-only.

SM addresses for slot 0	Description
SMW1100	EM bus Slot 0 vendor ID: 0x002A if a Siemens EM is present; 0x0000 if no EM is present
SMB1102 to SMB1121	EM bus Slot 0 order ID (MLFB): ASCII characters, left-justified in field, padded with spaces
SMB1122 to SMB1137	EM bus slot 0 serial number: ASCII characters, left-justified in field, padded with spaces
SMW1138	EM bus slot 0 hardware version: represents the hardware E-stand; range = 0x0001 to 0xFFFD 0x0000, 0xFFFE, and 0xFFFF are reserved values.
SMD1140	EM bus slot 0 firmware version: • Byte 0: ASCII 'V'; • Byte 1: Functional version, • Byte 2: Minor change version, • Byte 3: Bug fix version
SMW1144	EM bus slot 0 firmware version counter (range 0x0000 to 0x00FF)
SMW1146	Reserved, always 0x0000
SMW1148	EM bus slot 0 device type: I/O=0x0003, communications=0x0004, all other values reserved

SM addresses for slot 1	Description
SMW1150	EM bus slot 1 vendor ID: 0x002A if a Siemens EM is present; 0x0000 if no EM is present
SMB1152 to SMB1171	EM bus slot 1 order ID (MLFB): ASCII characters, left-justified in field, padded with spaces
SMB1172 to SMB1187	EM bus slot 1: serial number: ASCII characters, left-justified in field, padded with spaces
SMW1188	EM bus slot 1 hardware version: Represents the hardware E-stand; range=0x0001 to 0xFFFD 0x0000, 0xFFFE, and 0xFFFF are reserved values.
SMD1190	EM bus slot 1 firmware version: • Byte 0: ASCII 'V' • Byte 1: Functional version • Byte 2: Minor change version • Byte 3: Bug fix version
SMW1194	EM bus slot 1 firmware version counter (range 0x0000 to 0x00FF)
SMW1196	Reserved, always 0x0000
SMW1198	EM bus slot 1 device type: I/O=0x0003, communications=0x0004, all other values reserved

SM addresses for slot 2	Description
SMW1200	EM bus slot 2 vendor ID: 0x002A if a Siemens EM is present; 0x0000 if no EM is present
SMB1202 to SMB1221	EM bus slot 2 order ID (MLFB): ASCII characters, left-justified in field, padded with spaces
SMB1222 to SMB1237	EM bus slot 2 serial number: ASCII characters, left-justified in field, padded with spaces
SMW1238	EM bus slot 2 hardware version: Represents the hardware E-stand; range = 0x0001 to 0xFFFD 0x0000, 0xFFFE, and 0xFFFF are reserved values.
SMD1240	EM bus slot 2 firmware version: • Byte 0: ASCII 'V' • Byte 1: Functional version • Byte 2: Minor change version • Byte 3: Bug fix version
SMW1244	EM bus slot 2 firmware version counter (range 0x0000 to 0x00FF)
SMW1246	Reserved, always 0x0000
SMW1248	EM bus slot 2 device type: I/O=0x0003, communications=0x0004, all other values reserved

SM addresses for slot 3	Description
SMW1250	EM bus slot 3 vendor ID: 0x002A if a Siemens EM is present; 0x0000 if no EM is present
SMB1252 to SMB1271	EM bus slot 3 order ID (MLFB): ASCII characters, left-justified in field, padded with spaces
SMB1272 to SMB1287	EM bus slot 3 serial number: ASCII characters, left-justified in field, padded with spaces
SMW1288	EM bus slot 3 hardware version: Represents the hardware E-stand; range = 0x0001 to 0xFFFD 0x0000, 0xFFFE, and 0xFFFF are reserved values.
SMD1290	EM bus slot 3 firmware version: • Byte 0: ASCII 'V' • Byte 1: Functional version • Byte 2: Minor change version • Byte 3: Bug fix version
SMW1294	EM bus slot 3 firmware version counter (range 0x0000 to 0x00FF)
SMW1296	Reserved, always 0x0000
SMW1298	EM bus slot 3 device type: I/O=0x0003, communications=0x0004, all other values reserved

SM addresses for slot 4	Description
SMW1300	EM bus slot 4 vendor ID: 0x002A if a Siemens EM is present; 0x0000 if no EM is present
SMB1302 to SMB1321	EM bus slot 4 order ID (MLFB): ASCII characters, left-justified in field, padded with spaces
SMB1322 to SMB1327	EM bus slot 4 serial number: ASCII characters, left-justified in field, padded with spaces
SMW1338	EM bus slot 4 hardware version: Represents the hardware E-stand; range = 0x0001 to 0xFFFD 0x0000, 0xFFFE, and 0xFFFF are reserved values.
SMD1340	EM bus slot 4 firmware version: • Byte 0: ASCII 'V' • Byte 1: Functional version • Byte 2: Minor change version • Byte 3: Bug fix version

SM addresses for slot 4	Description
SMW1344	EM bus slot 4 firmware version counter (range 0x0000 to 0x00FF)
SMW1346	Reserved, always 0x0000
SMW1348	EM bus slot 4 device type: I/O=0x0003, communications=0x0004, all other values reserved

SM addresses for slot 5	Description
SMW1350	EM bus slot 5 vendor ID: 0x002A if a Siemens EM is present; 0x0000 if no EM is present
SMB1352 to SMB1371	EM bus slot 5 order ID (MLFB): ASCII characters, left-justified in field, padded with spaces
SMB1372 to SMB1387	EM bus slot 5 serial number: ASCII characters, left-justified in field, padded with spaces
SMW1388	EM bus slot 5 hardware version: Represents the hardware E-stand; range = 0x0001 to 0xFFFD 0x0000, 0xFFFE, and 0xFFFF are reserved values.
SMD1390	EM bus slot 5 firmware version: • Byte 0: ASCII 'V' • Byte 1: Functional version • Byte 2: Minor change version • Byte 3: Bug fix version
SMW1394	EM bus slot 5 firmware version counter (range 0x0000 to 0x00FF)
SMW1396	Reserved, always 0x0000
SMW1398	EM bus slot 5 device type: I/O=0x0003, communications=0x0004, all other values reserved

SM addresses for slot 6	Description
SMW250	EM bus slot 6 vendor ID: 0x002A if a Siemens EM is present; 0x0000 if no EM is present
SMB252 to SMB271	EM bus slot 6 order ID (MLFB): ASCII characters, left-justified in field, padded with spaces
SMB272 to SMB287	EM bus slot 6 serial number: ASCII characters, left-justified in field, padded with spaces
SMW288	EM bus slot 6 hardware version: Represents the hardware E-stand; range = 0x0001 to 0xFFFD 0x0000, 0xFFFE, and 0xFFFF are reserved values.
SMD290	EM bus slot 6 firmware version: • Byte 0: ASCII 'V' • Byte 1: Functional version • Byte 2: Minor change version • Byte 3: Bug fix version
SMW294	EM bus slot 6 firmware version counter (range 0x0000 to 0x00FF)
SMW296	Reserved, always 0x0000
SMW298	EM bus slot 6 device type: I/O=0x0003, communications=0x0004, all other values reserved

SM addresses for slot 7	Description
SMW300	EM bus slot 7 vendor ID: 0x002A if a Siemens EM is present; 0x0000 if no EM is present
SMB302 to SMB321	EM bus slot 7 order ID (MLFB): ASCII characters, left-justified in field, padded with spaces
SMB322 to SMB337	EM bus slot 7 serial number: ASCII characters, left-justified in field, padded with spaces
SMW338	EM bus slot 7 hardware version: Represents the hardware E-stand; range = 0x0001 to 0xFFFD 0x0000, 0xFFFE, and 0xFFFF are reserved values.

SM addresses for slot 7	Description
SMD340	EM bus slot 7 firmware version: • Byte 0: ASCII 'V' • Byte 1: Functional version • Byte 2: Minor change version • Byte 3: Bug fix version
SMW344	EM bus slot 7 firmware version counter (range 0x0000 to 0x00FF)
SMW346	Reserved, always 0x0000
SMW348	EM bus slot 7 device type: I/O=0x0003, communications=0x0004, all other values reserved

D.33 SMB1400-SMB1799: EM (expansion module) module-specific data

The CPU reserves an additional 50 bytes for each expansion module for read-only module-specific data:

SM addresses	Description
SMB1400 to SMB1449	EM bus Slot 0: Module specific information
SMB1450 to SMB1499	EM bus Slot 1: Module specific information
SMB1500 to SMB1549	EM bus Slot 2: Module specific information
SMB1550 to SMB1599	EM bus Slot 3: Module specific information
SMB1600 to SMB1649	EM bus Slot 4: Module specific information
SMB1650 to SMB1699	EM bus Slot 5: Module specific information
SMB1700 to SMB1749	EM bus Slot 6: Module specific information
SMB1750 to SMB1799	EM bus Slot 7: Module specific information

D.34 SMB1800-SMB1939: PROFINET device status

The SMB1800-SMB1935 section indicates the status of IO device or cyclic data if a S7-200 SMART CPU works as a controller.

SM address	Description
SMB1800 to SMB1807	Every byte indicates the status of each PROFINET device. • 0x00: Not configured. • 0x80: OK. • 0×81: Not connected. (The controller cannot connect with the device.) • 0×82: Diagnosis. (An alarm is reported.)
SMB1808 to SMB1871	Alarm status of each module. SMB1808~SMB1815 has 64 bits, which shows the alarm status of the device #1 (Device Number =1). 0 is "OK", 1 is "Error". For example, SM1808.0 indicates the first module status of the first device. SM1816.0 indicates the first module status of the second device.
SMB1872 to SMB1935	IO data status of IO device module. SMB1872~SMB1879 has 64 bits, which shows the alarm status of IO data in the device #1 (Device Number =1). 0 is "OK", 1 is "IO data error". For example, SM1872.0 indicates the first module IO data status of the first device. SM1880.2 indicates the third module IO data status of the second device.

The SM1936.0- SM1937.0 section shows the I-Device's connection status with its higher-level controller and the I-Device IO data status if a S7-200 SMART CPU works as an I-Device.

SM address	Description
SMB1936	This byte indicates the PROFINET I-Device's connection status with its higher-level controller. • 0x00: Not configured. • 0x80: OK. • 0×81: Not connected. (The I-Device is not connected with the higher-level controller.) • 0×82: Diagnosis. (The I-Device is connected with the higher-level controller, but the I-Device configuration doesn't match with the higher-level controller configuration.)
SM1937.0	I-Device IO data status. 0 is "OK", 1 is "IO data error".

D.35 SMB1940-SMB1946: Web server status

SMD1940 indicates the certificate validation status for Web server.

Value	Description
0	certificate OK
0x1	certificate expired
0x0200	The certificate validity starts in the future
0x10000000	Missing certificate
0x20000000	Certificate is broken
Others	Reserved certificate error

The SMB1944- SMB1946 section shows Web server status.

SM address	Description
SM1944.0	• 0: Web server is inactivated. • 1: Web server is activated.
SM1944.1	• 0: Web OSS is off. • 1: Web OSS is on.
SM1944.2~7	Reserved
SMB1945	Current activated web TCP connection
SMB1946	Maximum activated web TCP connection

References

E

E.1 Often-used special memory bits

The complete list of pre-defined special memory program variables is available to your project, in the system defined variable table.

Table E-1 Often-used special memory bits

SM address	Variable name	Description
SM0.0	Always_On	This bit is always TRUE.
SM0.1	First_Scan_On	The CPU sets this bit to TRUE, for the first scan cycle, and sets it to FALSE thereafter. One use for this bit is to call an initialization subroutine.
SM0.2	Retentive_Lost	TRUE for one scan cycle if retentive data is lost
SM0.3	RUN_Power_Up	TRUE for one scan cycle when RUN mode is entered from a power-up condition
SM0.4	Clock_60s	Clock pulse that is TRUE for 30 s, OFF for 30 s, for a cycle time of 1 min.
SM0.5	Clock_1s	Clock pulse that is TRUE for 0.5 s, OFF for 0.5 s, for a cycle time of 1 s.
SM0.6	Clock_Scan	Scan cycle clock that is TRUE for one scan cycle and OFF for the next scan cycle
SM0.7	RTC_Lost	For CPU models that have a real-time clock, the CPU sets this bit TRUE for one scan cycle if the time on the real-time clock device was reset or lost at power up. The program can use this bit as either an error memory bit or to invoke a special start-up sequence.
SM1.0	Result_0	Set to TRUE by the execution of certain instructions when the result of the operation = 0
SM1.1	Overflow_Illegal	Set to TRUE by the execution of certain instructions on overflow or illegal numeric value
SM1.2	Neg_Result	Set to TRUE when a math operation produces a negative result
SM1.3	Divide_By_0	Set to TRUE when an attempt is made to divide by zero
SM1.4	Table_Overflow	Set to TRUE when the Add to Table instruction attempts to overfill the table
SM1.5	Table_Empty	Set to TRUE when a LIFO or FIFO instruction attempts to read from an empty table
SM1.6	Not_BCD	Set to TRUE when an attempt is made to convert a non-BCD value to a binary value
SM1.7	Not_Hex	Set to TRUE when an ASCII value cannot be converted to a valid hexadecimal value

E.2 Interrupt events in priority order

Table E-2 Priority order for interrupt events

Priority group	Event	Description
Communications Highest Priority	8	Port 0 Receive character
	9	Port 0 Transmit complete
	23	Port 0 Receive message complete
	24	Port 1 Receive message complete
	25	Port 1 Receive character
	26	Port 1 Transmit complete
	46	Port 2 Receive message complete
	47	Port 2 Receive character
	48	Port 2 Transmit complete

Priority group	Event	Description
Discrete Medium Priority	19	PTO0 Pulse count complete
	20	PTO1 Pulse count complete
	34	PTO2 Pulse count complete
	45	PTO3 Pulse count complete
	67	PTO4 Pulse count complete
	0	I0.0 Rising edge
	2	I0.1 Rising edge
	4	I0.2 Rising edge
	6	I0.3 Rising edge
	1	I0.0 Falling edge
	3	I0.1 Falling edge
	5	I0.2 Falling edge
	7	I0.3 Falling edge
	35	I7.0 Rising edge (signal board 0, input 0)
	36	I7.0 Falling edge (signal board 0, input 0)
	37	I7.1 Rising edge (signal board 0, input 1)
	38	I7.1 Falling edge (signal board 0, input 1)
	49	I72.0 Rising edge (signal board 1, input 0)
	50	I72.0 Falling edge (signal board 1, input 0)
	51	I72.1 Rising edge (signal board 1, input 1)
	52	I72.1 Falling edge (signal board 1, input 1)
	12	HSC0 CV=PV (current value = preset value)
	27	HSC0 Direction changed
	28	HSC0 External reset
	13	HSC1 CV=PV (current value = preset value)
	14	HSC1 direction changed
	15	HSC1 external reset/Z phase
	16	HSC2 CV=PV (current value = preset value)
	17	HSC2 Direction changed
	18	HSC2 External reset
	32	HSC3 CV=PV (current value = preset value)
	53	HSC3 direction changed
	54	HSC3 external reset
	29	HSC4 CV=PV
	30	HSC4 direction changed
	31	HSC4 external reset
	33	HSC5 CV=PV
	43	HSC5 direction changed
	44	HSC5 external reset
	55	HSC6 CV=PV
	56	HSC6 direction changed
	57	HSC6 external reset
	58	HSC7 CV=PV

Priority group	Event	Description
	59	HSC7 direction changed
	60	HSC7 external reset
Timed Lowest Priority	10	Timed interrupt 0 SMB34
	11	Timed interrupt 1 SMB35
	21	Timer T32 CT=PT interrupt
	22	Timer T96 CT=PT interrupt
	61	Timer T262 CT=PT interrupt
	62	Timer T263 CT=PT interrupt
	63	Timer T264 CT=PT interrupt
	64	Timer T265 CT=PT interrupt
	65	Timer T266 CT=PT interrupt
	66	Timer T267 CT=PT interrupt

E.3 High-speed counter summary

	Clock A	Direction / Clock B	Reset	Single phase / Dual phase maximum clock / input rate	AB quadrature phase maximum clock / input rate
HSC0, HSC1, HSC2, HSC3, HSC4, HSC5 , HSC6, HSC7	I0.0-I1.7 [1]			• 200 kHz	• 200 kHz • A/B phase (4X) frequency support up to 800 kHZ

[1] For the same channel HSC, Clock A and Direction / Clock B can only be selected simultaneously in IB0（I0.0 ~I0.7） or simultaneously in IB1（I1.0 ~I1.7）.

E.4 STL instructions

Instructions

The STL instruction names and descriptions are shown in the tables below. See the chapter on program instructions for the LAD and FBD instructions.

Boolean instructions	
STL	**Description**
LD bit	Load
LDI bit	Load Immediate
LDN bit	Load Not
LDNI bit	Load Not Immediate
A bit	AND
AI bit	AND Immediate
AN bit	AND Not
ANI bit	AND Not Immediate

Boolean instructions	
STL	**Description**
O bit OI bit ON bit ONI bit	OR OR Immediate OR Not OR Not Immediate
LDBx IN1, IN2	Load result of Byte Compare IN1 (x:<, <=,=, >=, >, <>I) IN2
ABx IN1, IN2	AND result of Byte Compare IN1 (x:<, <=,=, >=, >, <>) IN2
OBx IN1, IN2	OR result of Byte Compare IN1 (x:<, <=,=, >=, >, <>) IN2
LDWx IN1, IN2	Load result of Word Compare IN1 (x:<, <=,=, >=, >, <>) IN2
AWx IN1, IN2	AND result of Word Compare IN1 (x:<, <=,=, >=, >, <>)I N2
OWxIN1, IN2	OR result of Word Compare IN1 (x:<, <=,=, >=, >, <>) IN2
LDDx IN1, IN2	Load result of DWord Compare IN1 (x:<, <=,=, >=, >, <>) IN2
ADx IN1, IN2	AND result of DWord Compare IN1 (x:<, <=,=, >=, >, <>)IN2
ODx IN1, IN2	OR result of DWord Compare IN1 (x:<, <=,=, >=, >, <>) IN2
LDRx IN1, IN2	Load result of Real Compare IN1 (x:<, <=,=, >=, >, <>) IN2
ARx IN1, IN2	AND result of Real Compare IN1 (x:<, <=,=, >=, >, <>) IN2
ORx IN1, IN2	OR result of Real Compare IN1 (x:<, <=,=, >=, >, <>) IN2
NOT	Stack Negation
EU ED	Detection of Rising Edge Detection of Falling Edge
= bit =I bit	Assign Value Assign Value Immediate
S bit, N R bit, N SI bit, N RI bit, N	Set bit Range Reset bit Range Set bit Range Immediate Reset bit Range Immediate
Not available in STL	SR (Set dominate bistable) RS (Reset dominate bistable)

Boolean instructions	
STL	**Description**
LDSx IN1, IN2 ASx IN1, IN2 OSx IN1, IN2	Load result of String Compare IN1 (x: =, <>) IN2 AND result of String Compare IN1 (x: =, <>) IN2 OR result of String Compare IN1 (x: =, <>) IN2
ALD OLD	AND Load OR Load
LPS LRD LPP LDS n	Logic Push (stack control) Logic Read (stack control) Logic Pop (stack control) Load Stack (stack control), n= 0 to 8
AENO	AND ENO

Math, increment, and decrement instructions	
STL	**Description**
+I IN1, IN2, OUT +D IN1, IN2, OUT +R IN1, IN2, OUT	Add Integer, Double Integer or Real IN1 + IN2 = OUT
-I IN1, IN2, OUT -D IN1, IN2, OUT -R IN1, IN2, OUT	Subtract Integer, Double Integer, or Real IN2 - IN1=OUT
MUL IN1, IN2, OUT	Multiply Integer (16*16->32)
*I IN1, IN2, OUT *D IN1, IN2, OUT *R IN1, IN2, IN2	Multiply Integer, Double Integer, or Real IN1 * IN2 = OUT
DIV IN1, IN2, OUT	Divide Integer (16/16->32)
/I IN1, IN2, OUT /D IN1, IN2, OUT /R IN1, IN2, OUT	Divide Integer, Double Integer, or Real IN2 / IN1 = OUT
SQRT IN, OUT	Square Root
LN IN, OUT	Natural Logarithm
EXP IN, OUT	Natural Exponential
SIN IN, OUT	Sine
COS IN, OUT	Cosine
TAN IN, OUT	Tangent
INCB OUT INCW OUT INCD OUT	Increment Byte, Word or DWord

Math, increment, and decrement instructions	
STL	**Description**
DECB OUT DECW OUT DECD OUT	Decrement Byte, Word, or DWord
PID TBL, LOOP	PID Loop

Timer and counter instructions	
STL	**Description**
TON Txxx, PT TOF Txxx, PT TONR Txxx, PT BITIM OUT CITIM IN, OUT	On-Delay Timer Off-Delay Timer Retentive On-Delay Timer Beginning Interval Timer Calculate Interval Timer
CTU Cxxx, PV CTD Cxxx, PV CTUD Cxxx, PV	Count Up Count Down Count Up/Down

High-speed instructions	
STL	**Description**
HDEF HSC, MODE	Define High-Speed Counter mode
HSC n	Activate High-Speed Counter
PLS n	Pulse Output:

Real time clock instructions	
STL	**Description**
TODR T TODW T TODRX T TODWX T	Read Time of Day clock Write Time of Day clock Read Real Time Clock Extended Set Real Time Clock Extended

Program control instructions	
STL	**Description**
END	Conditional End of Program
STOP	Transition to STOP Mode
WDR	WatchDog Reset (500 ms)
JMP N LBL N	Jump to defined Label Define a Label
CALL N [N1,...] CRET	Call a Subroutine [N1, ... up to 16 optional parameters] Conditional Return from SBR

Program control instructions	
STL	**Description**
FOR INDX,INIT,FINAL NEXT	For/Next Loop
LSCR N SCRT N CSCRE SCRE	Load, Transition, Conditional End, and End Sequence Control Relay segment
GERR ECODE	Get Error code

Move, Shift, and Rotate instructions	
STL	**Description**
MOVE IN, OUT	Move Byte, Word, DWord, Real, UDT, Array, string, FB instance

MOVB IN, OUT
MOVW IN, OUT
MOVD IN, OUT
MOVR IN, OUT
	Move Byte, Word, DWord, Real

BIR IN, OUT BIW IN, OUT	Move Byte Immediate Read Move Byte Immediate Write
BMB IN, OUT, N BMW IN, OUT, N BMD IN, OUT, N	Block Move Byte, Word, DWord
SWAP IN	Swap Bytes
SHRB DATA, S_BIT, N	Shift Register Bit
SRB IN, N, OUT SRW IN, N, OUT SRD IN, N, OUT	Shift Right Byte, Word, DWord
SLB IN, N, OUT SLW IN, N, OUT SLD IN, N, OUT	Shift Left Byte, Word, DWord
RRB IN, N, OUT RRW IN, N, OUT RRD IN, N, OUT	Rotate Right Byte, Word, DWord
RLB IN, N, OUT RLW IN, N, OUT RLD IN, N, OUT	Rotate Left Byte, Word, DWord

Logical instructions	
STL	**Description**
ANDB IN1, IN2, OUT ANDW IN1, IN2, OUT ANDD IN1, IN2, OUT	Logical AND of Byte, Word, and DWord
ORB IN1, IN2, OUT ORW IN1, IN2, OUT ORD IN1, IN2, OUT	Logical OR of Byte, Word, and DWord
XORB IN1, IN2, OUT XORW IN1, IN2, OUT XORD IN1, IN2, OUT	Logical XOR of Byte, Word, and DWord
NANDB IN1, IN2, OUT NANDW IN1, IN2, OUT NANDD IN1, IN2, OUT	Logical NAND of Byte, Word, and DWord
NORB IN1, IN2, OUT NORW IN1, IN2, OUT NORD IN1, IN2, OUT	Logical NOR of Byte, Word, and DWord
NXORB IN1, IN2, OUT NXORW IN1, IN2, OUT NXORD IN1, IN2, OUT	Logical NXOR of Byte, Word, and DWord
INVB OUT INVW OUT INVD OUT	Invert Byte, Word and DWord (1's complement)

String instructions	
STL	**Description**
SLEN IN, OUT	String Length
SCAT N, OUT	Concatenate String
SCPY IN, OUT	Copy String
SSCPY IN, INDX, N, OUT	Copy Substring from String
CFND IN1, IN2, OUT	Find First Character within String
SFND IN1, IN2, OUT	Find String within String

Table, Find, and Conversion instructions	
STL	**Description**
ATT DATA, TBL	Add data to table
LIFO TBL, DATA FIFO TBL, DATA	Get data from table
FND= TBL, PTN, INDX FND<> TBL, PTN, INDX FND< TBL, PTN, INDX FND> TBL, PTN, INDX	Find data value in table that matches comparison

Table, Find, and Conversion instructions	
STL	**Description**
FILL IN, OUT, N	Fill memory space with pattern
BCDI OUT IBCD OUT	Convert BCD to Integer Convert Integer to BCD
BTI IN, OUT ITB IN, OUT ITD IN, OUT DTI IN, OUT	Convert Byte to Integer Convert Integer to Byte Convert Integer to Double Integer Convert Double Integer to Integer
DTR IN, OUT TRUNC IN, OUT ROUND IN, OUT	Convert DWord to Real Convert Real to Double Integer Convert Real to Double Integer
ATH IN, OUT, LEN HTA IN, OUT, LEN ITA IN, OUT, FMT DTA IN, OUT, FM RTA IN, OUT, FM	Convert ASCII to Hex Convert Hex to ASCII Convert Integer to ASCII Convert Double Integer to ASCII Convert Real to ASCII
DECO IN, OUT ENCO IN, OUT	Decode Encode
SEG IN, OUT	Generate 7-segment pattern
ITS IN, FMT, OUT DTS IN, FMT, OUT RTS IN, FMT, OUT	Convert Integer to String Convert Double Integer to String Convert Real to String
STI STR, INDX, OUT STD STR, INDX, OUT STR STR, INDX, OUT	Convert Substring to Integer Convert Substring to Double Integer Convert Substring to Real

Interrupt instructions	
STL	**Description**
CRETI	Conditional Return from Interrupt
ENI DISI	Enable Interrupts Disable Interrupts
ATCH INT, EVNT DTCH EVNT	Attach Interrupt routine to event Detach event
CEVENT EVNT	Clear all interrupt events of type EVNT

Communications instructions	
STL	**LAD/FBD**
GET PUT	Reads remote station data Writes data to a remote station
XMT TBL, PORT RCV TBL, PORT	Freeport transmission Freeport receive message

Communications instructions	
STL	**LAD/FBD**
GIP ADDR, MASK, GATE SIP ADDR, MASK, GATE	Get CPU address, subnet mask, and gateway Set CPU address, subnet mask, and gateway
GPA ADDR, PORT SPA ADDR, PORT	Get Port Address Set Port Address

E.5 Memory ranges and features

The following table provides the memory ranges and features for CPUs of firmware V3.0.

Description		CPU SR40, CPU ST40	CPU SR60, CPU ST60
User program size		80 KB	100 KB
User data size		80 KB	100 KB
Variable memory (V)		VB0 to VB81919	VB0 to VB102399
Memory for unbinded global variables and FB instances (DB2)		1MB[1]	
Process image input register		I0.0 to I72.7	
Process image output register		Q0.0 to Q72.7	
Analog inputs (read only)		AIW0 to AIW530	
Analog outputs (write only)		AQW0 to AQW530	
Local memory (L)[2]		LB0 to LB1023	
Bit memory (M)		M0.0 to M63.7	
Special Memory (SM)	Total	SM0.0 to SM2047.7	
	Read only	SM0.0 to SM29.7 SMB480.0 to SM515.7 SM1000.0 to SM1699.7	
Timers		512(T0 to T511)	
Time accumulator	1 ms	T0, T64, T256 to T261	
	10 ms	T1 to T4, T65 to T68, T268 to T291	
	100 ms	T5 to T31, T69 to T95, T316 to T413	
On/Off delay	1 ms	T32, T96, T262 to T267	
	10 ms	T33 to T36, T97 to T100, T292 to T315	
	100 ms	T37 to T63, T101 to T255, T414 - T511	
Counters		512 (C0 to C511)	
High-speed counters		HC0 to HC7	
Sequential control relays (S)		S0.0 to S31.7	
Accumulator registers		AC0 to AC3	
Jumps/Labels		0 to 255	
Call/Subroutine		0 to 127	
Interrupt routines		0 to 127	
Positive/negative transitions		1024	

Description	CPU SR40, CPU ST40	CPU SR60, CPU ST60
PID control loops	0 to 15	
Ports	2 x Ethernet programming port Integrated RS485 port (port 0) CM01 Signal Board (SB) RS232/RS485 port (port 1, port 2)	

[1] The first 896KB is read/write, the last 128KB read only.

[2] LB1008 to LB1024 are reserved by STEP 7-Micro/WIN SMART V3 when programming in LAD or FBD.

The S7-200 SMART CPU firmware release V3.0 supports the PROFINET communication on the following four CPU models. For the detailed parameters and PROFINET process image information, refer to the following table.

Description	CPU SR40, CPU ST40	CPU SR60, CPU ST60
Maximum number of PROFINET device	8	
Device Number of PROFINET device	1 to 8	
Maximum input size of each PROFINET device	128 Bytes	
Maximum output size of each PROFINET device	128 Bytes	
Maximum number of modules	64	
Minimum cyclic update time of PROFINET device	The minimum value of the update time also depends on the communication component set for PROFINET, on the number of PROFINET devices, and the quantity of configured user data.	
CPU address range of PROFINET process image input register	I128.0 to I1151.7	
CPU address range of PROFINET process image output register	Q128.0 to Q1151.7	
CPU address of PROFINET process image input register for device #1	I128.0 to I255.7	
CPU address of PROFINET process image input register for device #2	I256.0 to I383.7	
CPU address of PROFINET process image input register for device #3	I384.0 to I511.7	
CPU address of PROFINET process image input register for device #4	I512.0 to I639.7	
CPU address of PROFINET process image input register for device #5	I640.0 to I767.7	
CPU address of PROFINET process image input register for device #6	I768.0 to I895.7	
CPU address of PROFINET process image input register for device #7	I896.0 to I1023.7	
CPU address of PROFINET process image input register for device #8	I1024.0 to I1151.7	
CPU address of PROFINET process image input register for I-Device	I1152.0 to I1279.7	
CPU address of PROFINET process image output register for device #1	Q128.0 to Q255.7	
CPU address of PROFINET process image output register for device #2	Q256.0 to Q383.7	
CPU address of PROFINET process image output register for device #3	Q384.0 to Q511.7	

Description	CPU SR40, CPU ST40	CPU SR60, CPU ST60
CPU address of PROFINET process image output register for device #4	Q512.0 to Q639.7	
CPU address of PROFINET process image output register for device #5	Q640.0 to Q767.7	
CPU address of PROFINET process image output register for device #6	Q768.0 to Q895.7	
CPU address of PROFINET process image output register for device #7	Q896.0 to Q1023.7	
CPU address of PROFINET process image output register for device #8	Q1024.0 to Q1151.7	
CPU address of PROFINET process image output register for I-Device	Q1152.0 to Q1279.7	

Ordering information

F

F.1 CPU modules

CPU models	Article number
CPU ST40, DC/DC/DC	6ES7288-1ST40-0AA2
CPU SR40, AC/DC/Relay	6ES7288-1SR40-0AA2
CPU SR60, AC/DC/Relay	6ES7288-1SR60-0AA2
CPU ST60, DC/DC/DC	6ES7288-1ST60-0AA2

F.2 Expansion modules (EMs) and signal boards (SBs)

Expansion modules and signal boards	Article number
EM Digital 8 x Inputs (EM DE08)	6ES7288-2DE08-0AA0
EM Digital 16 x Inputs (EM DE16)	6ES7288-2DE16-0AA0
EM Digital 8 x Outputs Transistor (EM DT08)	6ES7288-2DT08-0AA0
EM Digital 8 x Outputs Relay (EM DR08)	6ES7288-2DR08-0AA0
EM Digital 16 x Outputs Relay (EM QR16)	6ES7288-2QR16-0AA0
EM Digital 16 x Outputs Transistor (EM QT16)	6ES7288-2QT16-0AA0
EM Digital 8 x Inputs / Digital 8 x Outputs Transistor (EM DT16)	6ES7288-2DT16-0AA0
EM Digital 8 x Inputs/ Relay 8 x Outputs (EM DR16)	6ES7288-2DR16-0AA0
EM Digital 16 x Inputs / Digital 16 x Outputs Transistor (EM DT32)	6ES7288-2DT32-0AA0
EM Digital 16 x Inputs / Relay 16 x Outputs (EM DR32)	6ES7288-2DR32-0AA0
EM Analog 4 x Inputs (EM AE04)	6ES7288-3AE04-0AA0
EM Analog 2 x Outputs (EM AQ02)	6ES7288-3AQ02-0AA0
EM Analog 4 x Outputs (EM AQ04)	6ES7288-3AQ04-0AA0
EM Analog 8 x Inputs (EM AE08)	6ES7288-3AE08-0AA0
EM Analog 2 x Inputs / Analog 1 x Outputs (EM AM03)	6ES7288-3AM03-0AA0
EM Analog 4 x Inputs / Analog 2 x Outputs (EM AM06)	6ES7288-3AM06-0AA0
EM RTD 2 x 16 bit (EM AR02)	6ES 288-3AR02-0AA0
EM RTD 4 x 16 bit (EM AR04)	6ES7288-3AR04-0AA0
EM TC 4 x 16 bit (EM AT04)	6ES7288-3AT04-0AA0
EM DP01 PROFIBUS DP SMART (EM DP01)	6ES7288-7DP01-0AA0
SB Digital 2 x Inputs / Digital 2 x Outputs (SB DT04)	6ES7288-5DT04-0AA0
SB Analog 1 x Output (SB AQ01)	6ES7288-5AQ01-0AA0
SB Analog 1 x Input (SB AE01)	6ES7288-5AE01-0AA0

Expansion modules and signal boards	Article number
SB RS485/RS232 (SB CM01)	6ES7288-5CM01-0AA0
SB Battery (SB BA01)	6ES7288-5BA01-0AA0

F.3 Programming software

Programming software	Article number
STEP 7-Micro/WIN SMART V3 Individual License (CD-ROM)	6ES7288-8SW01-0AA1

F.4 Spare parts and other hardware

Cables, network connectors, repeaters, and end retainers	Article number
I/O Expansion cable, 1 m	6ES7288-6EC01-0AA0
MPI Cable	6ES7901-0BF00-0AA0
RS-232/PPI Multi-Master cable	6ES7901-3CB30-0XA0
USB/PPI Multi-Master cable	6ES7901-3DB30-0XA0
PROFIBUS Network cable	6XV1830-0EH30
Network Bus Connector with Programming Port Connector, Vertical Cable Outlet	6ES7972-0BB12-0XA0
Network Bus Connector (no programming port connector), Vertical Cable Outlet	6ES7972-0BA12-0XA0
RS485 Bus Connector with 35° Cable Outlet (no programming port connector)	6ES7972-0BA42-0XA0
RS485 Bus Connector with 35° Cable Outlet (with programming port connector)	6ES7972-0BB42-0XA0
RS485 IP 20 Repeater, Isolated	6ES7972-0AA02-0XA0
TD/CPU Connecting cable	6ES7901-3EB10-0XA0
End Retainer Thermoplast, 10 MM	8WA1808
End Retainer, Steel	8WA1805

Table F-1 Terminal block spare kits

If you have	Use this terminal block spare kit (4/pk)	
S7-200 SMART module (article number)	Terminal block article number	Terminal block description
CPU ST40, DC/DC/DC (6ES7288-1ST40-0AA1)	6ES7292-1AH30-0XA0	8 pin, tin-plated
	6ES7292-1AV30-0XA0	20 pin, tin-plated
	6ES7 292-1AL30-0XA0	11 pin, tin-plated
CPU SR40, AC/DC/Relay (6ES7288-1SR40-0AA1)	6ES7292-1AV30-0XA0	20 pin, tin-plated
	6ES7292-1AV40-0XA0	20 pin, tin-plated
	6ES7292-1AV30-0XA0	12 pin, tin-plated
CPU ST60, DC/DC/DC (6ES7288-1ST60-0AA1)	6ES7292-1AV30-0XA0	20 pin, tin-plated
	6ES7292-1AM30-0XA0	12 pin, tin-plated

If you have S7-200 SMART module (article number)	Use this terminal block spare kit (4/pk)	
	Terminal block article number	**Terminal block description**
CPU SR60, AC/DC/Relay (6ES7288-1SR60-0AA1)	6ES7292-1AV30-0XA0	20 pin, tin-plated
	6ES7292-1AV40-0XA0	20 pin, tin-plated
	6ES7292-1AV30-0XA0	12 pin, tin-plated
EM Digital 8 x Inputs (EM DE08) (6ES7288-2DE08-0AA0)	6ES7292-1AG30-0XA0	7 pin, tin-plated
EM Digital 8 x Outputs (EM DT08) (6ES7288-2DT08-0AA0)	6ES7292-1AG30-0XA0	7 pin, tin-plated
EM Digital 8 x Outputs Relay (EM DR08) (6ES7288-2DR08-0AA0)	6ES7292-1AG30-0XA0	7 pin, tin-plated
	6ES7292-1AG40-0XA0	7 pin, tin-plated, keyed-right
EM Digital 8 x Inputs / Digital 8 x Outputs (EM DT16) (6ES7288-2DT16-0AA0)	6ES7292-1AG30-0XA0	7 pin, tin-plated
EM Digital 8 x Inputs/ Relay 8 x Outputs (EM DR16) (6ES7288-2DR16-0AA0)	6ES7292-1AG30-0XA0	7 pin, tin-plated
	6ES7292-1AG40-0XA0	7 pin, tin-plated, keyed-right
EM Digital 16 x Inputs / Digital 16 x Outputs (EM DT32) (6ES7288-2DT32-0AA0)	6ES7292-1AL30-0XA0	11 pin, tin-plated
EM Digital 16 x Inputs / Relay 16 x Outputs (EM DR32) (6ES7288-2DR32-0AA0)	6ES7292-1AL30-0XA0	11 pin, tin-plated
	6ES7292-1AL40-0XA0	11 pin, tin-plated, keyed
EM Analog 4 x Inputs (EM AE04) (6ES7288-3AE04-0AA0)	6ES7292-1BG30-0XA0	7 pin, gold-plated
EM Analog 8 x Inputs (EM AE08) (6ES7288-3AE08-0AA0)	6ES7292-1BG30-0XA0	7 pin, gold-plated
EM Analog 2 x Outputs (EM AQ02) (6ES7288-3AQ02-0AA0)	6ES7292-1BG30-0XA0	7 pin, gold-plated
EM Analog 4 x Outputs (EM AQ04) (6ES7288-3AQ04-0AA0)	6ES7292-1BG30-0XA0	7 pin, gold-plated
EM Analog 2 x Inputs / Analog 1 x Outputs (EM AM03) (6ES7288-3AM03-0AA0)	6ES7292-1BG30-0XA0	7 pin, gold-plated
EM Analog 4 x Inputs / Analog 2 x Outputs (EM AM06) (6ES7288-3AM06-0AA0)	6ES7292-1BG30-0XA0	7 pin, gold-plated
EM RTD 2 x 16 bit (EM AR02) (6ES7288-3AR02-0AA0)	6ES7292-1BG30-0XA0	7 pin, gold-plated
EM RTD 4 x 16 bit (EM AR04) (6ES7288-3AR04-0AA0)	6ES7292-1BG30-0XA0	7 pin, gold-plated
EM TC 4 x 16 bit (EM AT04) (6ES7288-3AT04-0AA0)	6ES7292-1BG30-0XA0	7 pin, gold-plated
EM Profibus DP SMART (EM DP01) (6ES7288-7DP01-0AA0)	6ES7292-1BG30-0XA0	7 pin, gold-plated

Index

-

-D (STL-Subtract double integer), 362
-I (STL-Subtract integer), 362
-R (STL-Subtract real), 362

*****

*D (STL-Multiply double integer), 362
*I (STL-Multiply integer), 362
*R (STL-Multiply real), 362

.

.smart files, 143

/

/D (STL-Divide double integer), 362
/I (STL-Divide integer), 362
/R (STL-Divide real), 362

+

+D (STL-Add double integer), 362
+I (STL-Add integer), 362
+R (STL-Add real), 362

<

<=B, 297
<=R, 297
<=W, 297
<>B, 297
<>R, 297
<>S, 301
<>W, 297
<B, 297
<R, 297
<W, 297

=

= (STL-output), 249
==B, 297
==R, 297
==S, 301
==W, 297
=I (STL-output immediate), 249

>

>=B, 297
>=R, 297
>=W, 297
>B, 297
>R, 297
>W, 297

A

A (STL-AND), 241
AB< (STL-AND compare byte less than), 297
AB<= (STL-AND compare byte less than or equal), 297
AB<> (STL-AND compare byte not equal), 297
AB= (STL-AND compare byte equal), 297
AB> (STL-AND compare byte greater than), 297
AB>= (STL-AND compare byte greater than or equal), 297
ABS_DI, 376
ABS_I, 376
ABS_R, 376
ABSDI (STL-absolute double integer), 319, 376
ABSI (STL-absolute integer), 319, 376
Absolute positioning mode, 694
ABSR (STL-absolute real), 319, 376
AC
    isolation guidelines, 101
    wiring guidelines, 102
AC inductive loads, 104
Access rights
    CPU security, 49
    password privilege levels, 49
Active/Passive communication partners, 479
AD< (STL-AND compare double word less than), 297
AD<= (STL-AND compare double word less than or equal), 297
AD<> (STL-AND compare double word not equal), 297
AD= (STL-AND compare double word equal), 297
AD> (STL-AND compare double word greater than), 297

AD>= (STL-AND compare double word greater than or equal), 297
ADD_DI, 362
ADD_I, 362
ADD_R, 362
Addressing
accumulators, 115
analog inputs, 118
analog outputs, 118
counter memory, 114
creating pointers and using indirect address, 121
DB2 memory, 113
example of pointer offset to access data, 124
example of pointer to access data in a table, 123
flag memory, 113
high-speed counters, 115
local and expansion I/O, 120
local memory, 117
memory areas, 112
process image output register, 112
sequence control relay (SCR) memory, 119
special memory (SM) bits, 116
timer memory, 113
V memory, 113
variable memory, 112
AENO (STL stack-AND ENO bit)
instruction, 245
logic stack overview, 244
AI (STL-AND immediate), 242
Air flow, 87
Alarms
analog input configuration, 195
from system (SMW100-SMW120), 1091
ALD (STL stack-AND Load), 245
ALT (STL), 253
ALTP
STL, 253
AN (STL-AND NOT), 241
Analog I/O
input representation (current), 1026
input representation (voltage), 1026
output representation (current), 1027
output representation (voltage), 1027
status indicators, 138
step response times (SM), 1025
Analog inputs
analog type configuration, 193
rejection, 193
smoothing, 193
system block configuration, 193
Analog outputs
analog type configuration, 196
states at RUN/ STOP transition, 196
ANDB (STL-AND byte), 407
ANDD (STL-AND dword), 407
ANDW (STL-AND word), 407
ANI (STL-AND NOT immediate), 242
AR< (STL-AND compare real less than), 297
AR<= (STL-AND compare real less than or equal), 297
AR<> (STL-AND compare real not equal), 297
AR= (STL-AND compare real equal), 297
AR> (STL-AND compare real greater than), 297
AR>= (STL-AND compare real greater than or equal), 297
ARCCOS(arccosine), 368
ARCSIN(arcsine), 368
ARCTAN(Inverse tangent), 368
Article numbers, 1121, 1122
AS<> (STL-AND string compare not equal), 301
AS= (STL-AND string compare equal), 301
ASCII array conversion instructions, 305
Assigning
local variables, 162
local variables (local), 159
ATCH, 393
ATH, 305
ATT, 441
AVG_B, 375
AVG_DI, 375
AVG_I, 375
AVG_R, 375
AW< (STL-AND compare word less than), 297
AW<= (STL-AND compare word less than or equal), 297
AW<> (STL-AND compare word not equal), 297
AW= (STL-AND compare word equal), 297
AW> (STL-AND compare word greater than), 297
AW>= (STL-AND compare word greater than or equal), 297
Axes group 0 SM addresses (SMB800-SMB848), 1097
Axes group 1 SM addresses (SMB870-SMB928), 1098
Axes group configuration
buffer mode, 960
Axes group of motion
Axes group status, 809
Diagnose, 809
path, 957
segment, 957
Axes group of Motion
GRPx_2D_MOVELINEAR, 963
GRPx_3D_MOVELINEAR, 963
GRPx_MOVEPATH, 965

GRPx_RESET, 966
Programming, 955
AxesGroup of Motion
choosing an axis name, 807
Axis 0 motion control (SMB600-SMB649), 1094
Axis 1 motion control (SMB650-SMB699), 1094
Axis 2 motion control (SMB700-SMB749), 1094
Axis 3 motion control (SMB750-SMB799), 1094
Axis of motion
Axis status, 785, 796, 800
Diagnose, 785, 796, 800
Homing, 793
Modulo, 782, 788, 799
Original Point Offset, 794
Probe, 795, 799
Axis of Motion
ACCEL_TIME, 884
assigning the motor speed, 783, 790
AXISx_ABSPOS, 917
AXISx_CACHE, 915
AXISx_CFG, 914
AXISx_CTRL, 904
AXISx_DIS, 914
AXISx_GOTO, 907
AXISx_LDOFF, 911
AXISx_LDPOS, 912
AXISx_MAN, 906
AXISx_RDPOS, 916
AXISx_RSEEK, 910
AXISx_RUN, 909
AXISx_SRATE, 913
choosing an axis name, 780, 786, 797
choosing directional control and input configuration, 797
choosing directional control and output configuration, 786
configuring, 889
configuring reference point and seek parameters, 900
configuring the Backlash compensation, 899
configuring the input pin locations, 890
configuring the LMT- input, 789
configuring the LMT+ input, 789
configuring the RPS input, 793
configuring the STP input, 794
configuring the ZP input, 793
defining the motion profile, 902
displaying and controlling the operation of axes, 932
displaying and modifying the configuration of axes, 937
displaying the profile configuration for the axes, 937
eliminating backlash, 954
entering acceleration time, 897
entering jerk time, 898
entering jog parameters, 897
entering maximum start and stop speed, 896
error codes, 938
mapping the I/O, 891
phasing, 893
polarity, 895
programming, 888
RP Seek modes, 950
Selecting a measurement system, 781, 788, 799
setting the acceleration and deceleration times, 784, 792
setting the jerk time, 785, 792
SM locations, 948
subroutine guidelines, 904
subroutines, 903
AXISx_GOTO
override, 908

B

B_I, 302
BA01 battery signal board, 207
Baud rate
communications, 60, 183
setting, 552
switch selections:RS232/PPI Multi-Master cable, 570
BCD_I, 302
BCDI (STL-BCD to integer), 302
BGN_ITIME, 457
Biasing and terminating
CM01 signal board, 566
network cable, 565
Biasing PID loop, 379
BIR (STL-byte immediate read), 415
Bit logic instructions
AENO (STL-AND ENO), 244
ALT, 253
ALTP, 253
contacts, 241
contacts (Immediate), 242
edge detectors, 248
input examples, 253
NOP (No operation), 252
NOT, 247
output coils, 249
output examples, 255

set and reset bits, 250
set and reset dominant bistable, 251
STL logic stack instructions for inputs, 244
STL logic stack operations, 245
BITIM (STL-Begin interval timer), 457
BIW (STL-byte immediate write), 415
BLKMOV_B, 413
BLKMOV_D, 413
BLKMOV_W, 413
BMB (STL-block move byte), 413
BMD (STL-block move double word), 413
BMW (STL- block move word), 413
Bookmarks in programs, 737
BTI (STL-Byte to integer), 302
Buffer consistency
PROFIBUS, 534
Buffer mode
Aborting, 960
Buffered, 960
Building status charts, 742
Building your communication network, 562
Byte consistency
PROFIBUS, 534

C

Cable
expansion, 1051
Cables
USB/PPI Multi-Master, 1052
CAL_ITIME, 457
CALL, 458, 463
Cam table, 801
Card (memory), 132, 133
CE approval, 981
CFND (STL-Find character), 438
Character error received from Freeport (SMB3), 1071
Charts
building, 742
creating, 742
opening, 742
CHR_FIND, 438
CITIM (STL-Calculate interval time), 457
Clearance for airflow and cooling, 87
Cleared memory card, 210
Clearing PLC memory, 208
Cleasr, 67
Clock instructions
READ_RTC, 257
READ_RTCX, 259
SET_RTC, 257
SET_RTCX, 259
CLR_EVNT, 393
CM01 signal board
biasing and terminating, 566
connector pin assignments, 565
Coils
output, 249
output (immediate), 249
reset bits, 250
reset bits immediate, 250
set bits, 250
set bits immediate, 250
Cold junction compensation
thermocouple, 205
Thermocouple, 1030
Communication drivers, 478
Communication ports, 477
connector pin assignments, 564
Freeport mode, 568
system block configuration, 60, 183
Communications
dialog, 483, 557
Ethernet hardware connection, 71
Ethernet, RS485, and RS232, 29, 475
IP address, 483
locating MAC address on CPU, 490
network, 70
number of connections (Ethernet), 476
number of connections (RS232), 476
number of connections (RS485), 476
point-to-point interface (PPI) protocol, 551
restricting writes, 49
RS485 hardware connection, 74
RS485 network address, 557
Communications instructions
receive, 270
transmit, 270
Communications protocol
PROFIBUS, 530
Compare instructions
compare character strings, 301
compare number value, 297
Compatibility
EM DP01 PROFIBUS DP module, 1049
Compiling programs
downloading, 124
PLC information (errors), 169
PLC non-fatal program errors, 1062
Component properties, 63
Configuration
CPU, system block, 181
dynamic IP information, 483
dynamic PROFINET device name, 502

EM DP01 PROFIBUS DP, 534
Ethernet, 483
IP address, 483
MAC address, 490
profile table (Axis of Motion), 941
RS485 network, 557
RS485 network address, 557
static IP information, 485
Configuration drawings, 139
Connections
number of connections (Ethernet), 476
number of connections (RS232), 476
number of connections (RS485), 476
types of communication, 29, 475
Connector, 95
Contact information, 4
Contacts
ALT, 253
ALTP, 253
negative edge detector, 248
normally closed, 241
normally closed (immediate), 242
normally open, 241
normally open (immediate), 242
NOT, 247
positive edge detector, 248
Contamination level, 984
Convert instructions
ASCII array conversions, 305
ASCII sub-string to number value, 315
encode and decode, 317
number value to string, 311
standard conversion, 302
Cooling, 87
COS (cosine), 368
Counter instructions
high-speed counters, 325
standard counters, 322
CPU
accessing data, 111
clearing memory, 208
configuring communication to HMI, 491
connecting power, 69
CR40 wiring diagram, 992
CR60 wiring diagram, 1001
dimensions, 23, 89
DIN rail, 92
Ethernet communication, 480
Ethernet port, 483
expansion cable, 99
expansion modules supported, 27
fatal errors, 1065
Features, 23
installation, 90
installation on a panel, 91
IP address, 483
isolation guidelines, 101
LED indicators, 137
LEDs, 23
MAC address, 490
memory card, 132
non-fatal error memory locations, 1064
number of communication connections, 476
number of PPI connections, 552
RS485 network address, 557
RS485 network port, 557
setting the type, 80
SR40 specifications, 986
SR40 wiring diagram, 992
SR60 specifications, 994
SR60 wiring diagram, 1001
ST40 specifications, 986
ST40 wiring diagram, 992
ST60 specifications, 994
ST60 wiring diagram, 1001
system block, 181
types of communication, 29, 475
viewing PLC information, 169
wiring guidelines, 102
CPU hardware/firmware ID (SMB1000-SMB1049), 1099
CPU ID register (SMB6-SMB7), 1073
CPU role
Controller, 506
I-Device, 514
CRET (STL-Conditional return from function block), 463
CRET (STL-Conditional return from subroutine), 458
Cross reference, 737
CTD (count down), 322
CTU (count up), 322
CTUD (count up/down), 322
Customer support, 4

D

Data
receiving, 272
retention, 190
Data block
password protection, 63
Data block (DB), 142
Data consistency
PROFIBUS, 534

Data log
status (SMB480-SMB515), 1094
DC
isolation guidelines, 101
wiring guidelines, 102
DC inductive loads, 104
Debugging and monitoring
forcing values, 744
program editor status, 739
DEC_B, 370
DEC_DW, 370
DEC_W, 370
DECB (STL-Decrement byte), 370
DECD (STL-Decrement double word), 370
DECO, 317
DECW (STL-Decrement word), 370
Default gateway IP address, 482
Defining
local variables, 162
Deleting GSDML files, 501
Device configuration of CPU and modules, 181
DI_I, 302
DI_R, 302
DI_S, 311
Diagnostics
LED indicators, 137
Differential term, PID algorithm, 388
Digital input filter time, 186
Digital input filters, 187
Digital inputs, 186
Digital outputs, 189
Dimensions
CPU, 23
mounting, 89
DIN rail, 90, 92
DISI, 393
Displaying status, 739
DIV, 366
DIV_DI, 362
DIV_I, 362
DIV_R, 362
Downloading
programs, 124
sample program, 82
DP device
EM DP01 PROFIBUS DP, 532
Drive communication
calculating time requirement, 663
Drives, 887, 917
DTA, 305
DTCH, 393
DTI (STL-Double integer to integer), 302
DTR (STL-Double integer to real), 302
DTS (STL-Double integer to string), 311
Duplicate POU name assignment, 63
Dynamic IP information, 483

E

ED (STL-Edge Down), 248
Edge detectors, 248
Editors
FBD, 152
LAD, 151
STL, 152
Electromagnetic compatibility (EMC), 982
Element usage, 737
EM (expansion module) hardware/firmware ID (SMB1100-SMB1299), 1100
EM DE16 specifications, 1004
EM DP01 PROFIBUS DP module
status LEDs, 1050
wiring diagram, 1051
EM DP01 PROFIBUS DP module
data exchange mode, 541
EM DP01 PROFIBUS DP module
configuration options, 535
DP protocol, 532
on PROFIBUS network, 533
EM DP01 PROFIBUS DP module
additional configuration features, 544
GSD device database file, 545
ENCO, 317
END, 425
ENI, 393
Environmental
industrial environments, 982
operating conditions, 983
transport and storage conditions, 983
Error code for axes group, 978
Error code, axes group, 869
Error code, Cam and Gear, 869
Error code, single axis, 866
Errors
Axis of Motion, 938
character error received from Freeport communication (SMB3), 1071
compile and run-time errors (PLC program), 1062
data retention, 190
fatal (PLC), 1065
fatal error effect on run-time execution, 176
GET_TABLE and PUT_TABLE instructions, 263
I/O error status, 1072

I/O module ID and error registers (SMB8-SMB19), 1073
memory locations (PLC non-fatal errors), 1064
Modbus RTU master execution, 589
Modbus RTU slave execution, 595
Modbus TCP client execution, 604
Modbus TCP general communication exception codes, 615
Modbus TCP general exception codes, 614
Modbus TCP server execution, 606
Motion instruction, 939
non-fatal error effect on run-time execution, 175
PID auto-tune, 763
PID control, 1093
PLC information, 169
PWMx_RUN subroutine, 882
signal board ID and error registers (SMB28-SMB29), 1075
signal board ID and error registers (SMB32-SMB33), 1076
timestamp mismatch (PC/PLC program difference), 1061
Ethernet
configuring communication between CPU and HMI device, 491
GET, 262
IP address, 482
ISO-on-TCP protocol, 493
MAC address, 490
networks, 479
number of communication connections, 476
Ports, 493
TCP protocol, 492
TSAPs, 494
types of communication, 29, 475
UDP protocol, 492
Ethernet communications
STEP 7-Micro/WIN SMART V3 settings, 71
Ethernet network
configuring the IP address for a CPU, 483
searching for CPU, 489
EU STL-Edge Up), 248
Events, interrupts, 395
Example
Axes group of motion simple absolute move (pick-and-place application), 969
Example of creating a PROFINET network, 516
Examples
Axes group, 874
Axis of Motion AXISx_CTRL, AXISx_RUN, AXISx_SEEK, and AXISx_MAN subroutines application, 927
Axis of Motion simple relative move (cut-to-length application), 925
bit logic input, 253
bit logic output, 255
Cam, 876
configuring the PROFIBUS DP EM DP01 I/O, 538
conversion instructions, 304
count up/down counter instruction, 324
DISCONNECT instruction, 638
GET and PUT instructions, 267
high-speed counter initialization sequences, 340
high-speed counter programming, 330
installing the PROFIBUS DP EM DP01 GSD file, 536
ISO_CONNECT instruction, 624
Modbus RTU slave protocol, programming, 594
Open user communication (OUC) library, 640, 649, 651
PROFIBUS DP communications to a CPU, 549
Shift register bit (SHRB) instruction, 434
Single Axis, 871
subroutine for sampling the value of an analog input, 142
Table Find (TBL_FIND) instruction, 447
tables, 447
TCP_CONNECT instruction, 621
TCP_RECV instruction, 631
UDP_CONNECT instruction, 626
UDP_RECV instruction, 637
UDP_SEND instruction, 634
Using the AXISx_ABSPOS subroutine to read the absolute position from a SINAMICS V90 servo drive, 919
Executing
program, 109
single or multiple scans, 746
EXP (natural exponential), 368
Expansion and local I/O addressing, 120
Expansion cable, 1051
installation, 99
removal, 99
Expansion I/O bus - communication errors (SMW98), 1091
Expansion module
EM QR16, 1006
Expansion module (EM)
EM AE04 specifications, 1016
EM AE04 wiring diagram, 1018
EM AM06 specifications, 1021
EM AM06 wiring diagram, 1024
EM AQ02 specifications, 1019
EM AQ02, wiring diagram, 1020
EM AR02 (RTD) specifications, 1033

EM AR02 (RTD) wiring diagram, 1037
EM AT04 specifications, 1028
EM DE08 specifications, 1004
EM DE08 wiring diagram, 1005
EM DR08 specifications, 1006
EM DR08 wiring diagram, 1008
EM DR16 specifications, 1010
EM DR16 wiring diagram, 1013
EM DR32 specifications, 1010
EM DR32 wiring diagram, 1014
EM DT08 specifications, 1006
EM DT08 wiring diagram, 1008
EM DT16 specifications, 1010
EM DT16 wiring diagram, 1013
EM DT32 specifications, 1010
EM DT32 wiring diagram, 1014
installing and removing, 98
wiring diagram, 1028
Expansion module (SB)
SB AE01 specifications, 1040
Expansion module EM)
EM Q04, wiring diagram, 1020
expansion modules
EM DE08, 1004
Expansion modules, 27, 1072
dimensions, 89
module error status (SMB5), 1072
module ID and error registers (SMB8-SMB19), 1073
PLC information, 169
Expansion modules (EM)
analog input representation (current), 1026
analog input representation (voltage), 1026
analog output representation (current), 1027
analog output representation (voltage), 1027

F

Factory defaults memory card, 210
Factory reset, 67
Fatal error effect on run-time execution, 176
Fatal errors (PLC), 1065
FBD editor, 152
Features
CPU, 23
expansion modules supported, 27
Features of the new version, 24, 25
FIFO, 442
File menu
Download, 124
GSDML Management, 499
Upload, 127
FILL (STL-table fill), 444
FILL_N, 444
Filter time, 186
Filters, digital input configuration, 187
Find PROFINET Devices, 502
Firmware update, 130, 169
First scan flag (SMB0), 1069
First scan, executing, 746
Flag memory, 113
Floating point values, 389
FND=, <>, <, > (STL-table find), 445
FOR, 416
Force
writing and forcing outputs in STOP mode, 745
Forced value indicator (SMB4), 1071
Forgotten password, 49, 210
Freeport mode
character interrupt control, 280
enabling, 271
example, 568
Freeport character error (SMB3), 1071
Freeport configuration (SMB30-Port 0, SMB31-Port2 and SMB130-Port 1), 1077
Freeport receive character (SMB2), 1070
Freeport transmitter idle (SMB4), 1071
interrupts, 399
SMB405-SMB413 receive message control, SMB86-SMB94 and SMB186-SMB194 receive message control, 1087
Freeze outputs
analog output configuration, 196
digital output configuration, 189
Function Block Diagram editor, 152
function block instructions
CALL, RET, 463
Function block instructions
Call parameter and return examples, 465

G

General technical specifications, 981
GERR (STL-Get non-fatal error code), 426
GET, 262
GET_ADDR, 282
GET_ERROR, 426
GIP_ADDR, 283
Grounding, 102
GRPx_RESET
ErrorStop, 967
GSD file
EM DP01 PROFIBUS DP module, 545
GSDML, 498

Guidelines
grounding and circuit, 100
installation on a panel, 91
installation procedures, 90
isolation, 101
wiring guidelines, 102

H

Hardware troubleshooting, 748
HDEF (high-speed counter definition), 325
Heat, high voltage, and electrical noise, 86
High-speed counters, 330
High-vibration environment, 92
HMI
configuring Ethernet communication, 491
devices, 567
general guidelines for networks, 567
multi-master and multi-slave PPI networks, 556
single-master PPI networks, 555
supported devices, 28, 477
Hotline, 4
HSC (high-speed counter), 210, 325
HTA, 305

I

I/O
analog input representation (current), 1026
analog input representation (voltage), 1026
analog output representation (current), 1027
analog output representation (voltage), 1027
analog status indicators, 138
step response times (SM), 1025
I/O addressing, 120
I/O error status
PLC non-fatal error codes, 1062
PLC non-fatal error SM flags, 1064
SMB5, 1072
I/O expansion bus - communication errors (SMW98), 1091
I/O Module ID and error registers (SMB8-SMB19), 1073
I_B, 302
I_BCD, 302
I_DI, 302
I_S, 311
IBCD (STL-Integer to BCD), 302
I-Device, 496
I-Device with lower-level PROFINET IO system, 497
I-Device without lower-level PROFINET IO system, 497
I-Device without lower-level PROFINET IO system, 514
Immediate instructions
LAD, FBD, and STL, 242
Importing GSDML files, 499
INC_B, 370
INC_DW, 370
INC_W, 370
INCB (STL-Increment byte), 370
INCD (STL-Increment double word), 370
Incremental jog mode, 715
INCW (STL-Increment word), 370
Indirect addressing
creating pointers and using indirect address, 121
example of pointer offset to access data, 124
example of pointer to access data in a table, 123
global variable table, 158
Inductive loads, 104
Input filter time, 186
Input process image register, 108
inputs
ALTP contact, 253
Inputs
ALT contact, 253
edge detectors, 248
example bit logic, 253
NOT contact, 247
physical and in program, 107
pulse catch bits (system block), 187
reading, 108
standard contacts, 241, 242
STL logic stack, 244
Installation
clearance for airflow and cooling, 87
dimensions, 89
DIN rail, 92
expansion cable, 99
expansion module (EM), 98
grounding, 102
grounding and circuit, 100
guidelines, 85
high-vibration environment, 92
inductive loads, 104
isolation, 101
isolation guidelines, 101
lamp loads, 103
overview, 85, 90
panel, 91

separate the devices from heat, high voltage, and electrical noise, 86
signal board (SB), 94
terminal block connector, 95
wiring guidelines, 102
Instruction execution status bits (SMB1), 1070
Instruction libraries, 575
Instructions
GET, 262
GET_ADDR, 282
GIP_ADDR, 283
loop control (PID), 378
MBUS_CLIENT, 601
MBUS_SERVER, 605
PUT, 262
quick reference guide, 1110
SET_ADDR, 282
SIP_ADDR, 283
Insulation, 984
Integral term, PID algorithm, 387
Interrupt routines, 109
elements of a user program, 142
Interrupts
attach/detach, enable/disable, conditional return, and clear event instructions, 393
event support by CPU model, 395
example programs, 400
global interrupt enable state (SMB4), 1071
interrupt queue overflow (SMB4), 1071
nesting subroutines/FBs, 458, 464
overview, 395
priority and queuing, 400
programming guidelines, 397
time interval values for timed interrupts (SMB34-SMB35), 1078
types of interrupt events, 399
INV_B, 406
INV_DW, 406
INV_W, 406
INVB (STL-invert byte), 406
INVD (STL-invert double word), 406
INVW (STL-invert word), 406
IP address, 482
assigning, 481, 488
configuring, 483
MAC address, 490
IP router, 483
Isolation, 101
Isolation guidelines, 101
ITA, 305
ITB (STL-Integer to byte), 302
ITD (STL-Integer to double integer), 302
ITS (STL-Integer to string), 311

J

JMP, 417
Jog mode, 712

L

L memory, 159
LAD editor
features, 151
Lamp loads, 103
LBL, 417
LD (STL stack-Load NOT immediate), 244
LD (STL stack-Load), 244
LD (STL-Load), 241
LDB< (STL-Load compare byte less than), 297
LDB<= (STL-Load compare byte less than or equal), 297
LDB<> (STL-Load compare byte not equal), 297
LDB= (STL-Load compare byte equal), 297
LDB> (STL-Load compare byte greater than), 297
LDB>= (STL-Load compare byte greater than or equal), 297
LDD< (STL-Load compare double word less than), 297
LDD<= (STL-Load compare double word less than or equal), 297
LDD<> (STL-Load compare double word not equal), 297
LDD= (STL-Load compare double word equal), 297
LDD> (STL-Load compare double word greater than), 297
LDD>= (STL-Load compare double word greater than or equal), 297
LDI (STL stack-Load immediate), 244
LDI (STL-Load NOT immediate), 242
LDN (STL stack-Load NOT), 244
LDN (STL-Load NOT), 241
LDNI (STL-Load NOT immediate), 242
LDR< (STL-Load compare real less than), 297
LDR<= (STL-Load compare real less than or equal), 297
LDR<> (STL-Load compare real not equal), 297
LDR= (STL-Load compare real equal), 297
LDR> (STL-Load compare real greater than), 297
LDR>= (STL-Load compare real greater than or equal), 297
LDS (STL stack-Load), 245
LDS<> (STL-Load string compare not equal), 301
LDS= (STL-Load string compare equal), 301

LDW< (STL-Load compare word less than), 297
LDW<= (STL-Load compare word less than or equal), 297
LDW<> (STL-Load compare word not equal), 297
LDW= (STL-Load compare word equal), 297
LDW> (STL-Load compare word greater than), 297
LDW>= (STL-Load compare word greater than or equal), 297
LED indicators
    CPU status, 137
Library
    creating, 575
    types, 573
LIFO, 442
LN (natural logarithm), 368
Local and expansion I/O addressing, 120
local variable table, 159
Local variables, 159
Local/Partner connection, 479
Locks
    project tree, 63
Logic stack
    STL inputs, 244
    STL stack operations, 245
Logic, control, 107
Logical operation instructions
    A, 319
    Absolute, 376
    AND, OR, XOR (byte, word, and dword), 407
    invert, 406
    Low-pass filter, 377
    NAND, NOR, NXOR (byte, word, and dword), 407
Loop control (PID)
    adjusting bias, 391
    converting inputs, 389
    converting outputs, 390
    error conditions, 392
    forward/reverse, 391
    loop definition table, 755
    modes, 392
Loop table, 393
Lost password, 210
LPF, 377
LPF (STL-low pass filter), 377
LPP (STL stack-Logic Pop), 245
LPS (STL stack-Logic Push), 245
LRD (STL stack-Logic Read), 245
LSCR (STL-Load SCR), 418

M

MAC address, 490
Main entry, 1005
Main program, 141
Math instructions
    add, subtract, multiply and divide, 362
    divide integer with remainder, 366
    increment and decrement, 370
    multiply integers to double integer and divide integer with remainder, 366
    numeric functions, 368, 371
MBUS_CLIENT, 601
MBUS_CTRL (initialize Modbus master communication), 585
MBUS_INIT (initialize slave communication), 592
MBUS_MSG / MB_MSG2 (send message from Modbus master), 586
MBUS_SERVER, 605
MBUS_SLAVE (slave response to master message), 594
MC_AbortTrigger, 818
MC_CamOut, 843
MC_GearIn, 844
MC_GearOut, 846
MC_GroupDisable, 858
MC_GroupEnable, 857
MC_GroupReset, 859
MC_GroupStop, 860
MC_Halt, 823
MC_Home, 820
MC_ImmediateStop, 831
MC_MoveAbsolute, 825
MC_MoveJog, 832
MC_MoveLinearAbsolute, 861
MC_MoveLinearRelative, 864
MC_MoveRelative, 827
MC_MoveVelocity, 829
MC_PhasingRelative, 847
MC_Power, 816
MC_Reset, 819
MC_Stop, 822
MC_TouchProbe, 817
Memory, 1064
    addresses for non-fatal error indicators, 1064
    clearing PLC, 208
    retentive range configuration, 190
Memory card
    program transfer card, 132, 133
    reset to factory defaults, 210
    types of, 128
    using, 129
MIN_MAX_B, 371
MIN_MAX_DI, 371
MIN_MAX_I, 371

MIN_MAX_R, 371
Modbus general
addressing, 577
advanced user information, 597, 612
initialization and execution time for Modbus protocol, 582
library features, 579
requirements for using Modbus instructions, 580
Modbus RTU library overview, 573
Modbus RTU master
example program, 595
execution error codes, 589
MBUS_CTRL (initialize master communication), 585
MBUS_MSG / MB_MSG2 (send message from master), 586
using the instructions, 583
Modbus RTU slave
execution error codes, 595
MBUS_INIT (initialize slave communication), 592
MBUS_SLAVE (slave response to master message), 594
using the instructions, 591
Modbus TCP
general exception codes, 614, 615
MODBUS TCP
MBUS_CLIENT, 601
MBUS_SERVER, 605
Modbus TCP client
execution error codes, 604
Modbus TCP library
example, 607
Modbus TCP library features, 599
Modbus TCP library overview, 573
Modbus TCP server
execution error codes, 606
Modules
CPU SR40, 986
CPU SR60, 994
CPU ST40, 986
CPU ST60, 994
dimensions, 89
EM AE04, 1016
EM AM06, 1021
EM AQ02, 1019
EM AR02 (RTD), 1033
EM DE08, 1004
EM DE16 specifications, 1004
EM DP01 PROFIBUS DP, 1048
EM DR08, 1006
EM DR16, 1010
EM DR32, 1010
EM DT08, 1006
EM DT16, 1010
EM DT32, 1010
EM QR16, 1006
EM QT16, 1006
SB AE01, 1040
SB AQ01, 1043, 1044
SB CM01, 1045
SB DT04, 1038, 1040
Motion control
configuring reference point and seek parameters, 900
configuring the Backlash compensation, 899
configuring the input pin locations, 890
defining the motion profile, 902
entering acceleration time, 897
entering jerk time, 898
entering jog parameters, 897
entering maximum start and stop speed, 896
mapping the I/O, 891
Motion features, 886
phasing, 893
polarity, 895
Motion control panel
displaying and controlling the operation of axes, 932
displaying and modifying the configuration of axes, 937
displaying the profile configuration for the axes, 937
Motion inputs and outputs
CPU, 887
Motion instruction, error codes, 939
Motion profile
configuring, 884
creating steps, 886
defining, 885
mode of operation, 885
Motion wizard
configuration/profile table, 941
maximum and start/stop speeds, 883
open loop motion control, 879
SM locations, 948
Mounting
DIN rail, 92
expansion cable, 99
isolation, 101
overview, 90
panel, 91
wiring guidelines, 102
MOV_B, 412
MOV_BIR, 415

MOV_BIW, 415
MOV_DW, 412
MOV_R, 412
MOV_W, 412
MOVB (STL-move byte), 412
MOVD (STL-move double word), 412
Move instructions
  block move (byte, word, dword), 413
  move (byte, word, dword, real), 412
  move (byte, word, dword, real, UDT, array, string, FB instance), 411
  move byte immediate (read and write), 415
  SWAP (exchange byte data in a word), 414
MOVR (STL-move real), 412
MOVW (STL-move word), 412
MUL, 366
MUL_DI, 362
MUL_I, 362
MUL_R, 362
Multiple scans, 746

N

NAND_B, 407
NAND_DW, 407
NAND_W, 407
NANDB (STL-NAND byte), 407
NANDD (STL-NAND dword), 407
NANDW (STL-NAND word), 407
Nested subroutines/FBs, 458, 464
Network communications, 70
Networks (communication)
  addresses, 553
  biasing and terminating the network cable, 565
  biasing cable, 566
  calculating network distances, 562
  general guidelines for building, 562
  safety concerns, 562
  sample RS485 network configurations, 555
  selecting the network cable, 564
  single-master PPI, 556
  types of communication, 29, 475
New features, 24, 25
NEXT, 416
Non-fatal error effect on run-time execution, 175
Non-fatal PLC errors
  compile and run-time, 1062
  Special Memory locations, 1064
Non-volatile memory, 128
NOP, 252
NOR_B, 407
NOR_DW, 407
NOR_W, 407
NORD (STL-NOR dword), 407
Normally closed contact
  immediate, 242
  standard, 241
Normally open contact
  immediate, 242
  standard, 241
NORW (STL-NOR word), 407
NOT (STL), 247
Number value to string conversion instructions, 311
NXOR_B, 407
NXOR_DW, 407
NXOR_W, 407
NXORB (STL-NXOR byte), 407
NXORD (STL-NXOR dword), 407
NXORW (STL-NXOR word), 407

O

O (STL-OR), 241
OB< (STL-OR compare byte less than), 297
OB<= (STL-OR compare byte less than or equal), 297
OB<> (STL-OR compare byte not equal), 297
OB= (STL-OR compare byte equal), 297
OB> (STL-OR compare byte greater than), 297
OB>= (STL-OR compare byte greater than or equal), 297
OB1, 141
OD< (STL-OR compare double word less than), 297
OD<= (STL-OR compare double word less than or equal), 297
OD<> (STL-OR compare double word not equal), 297
OD= (STL-OR compare double word equal), 297
OD> (STL-OR compare double word greater than), 297
OD>= (STL-OR compare double word greater than or equal), 297
offline mode, 136
OI (STL-OR immediate), 242
OLD (STL stack-OR Load), 245
ON (STL-OR NOT), 241
ONI (STL-OR NOT immediate), 242
Open loop control, 879
Open Source Software, 34
  Use, 35
Open user communication (OUC)
  connection instructions, 493
  connection types, 493
Open user communication (OUC) library
  common library instruction parameters, 617
  DISCONNECT instruction, 637

instruction error codes, 639
ISO_CONNECT instruction, 622
overview, 573
TCP_CONNECT instruction, 619
TCP_RECV instruction, 629
TCP_SEND instruction, 627
UDP_CONNECT instruction, 625
UDP_RECV instruction, 635
UDP_SEND instruction, 632
Opening earlier STEP 7-Micro/WIN projects, 143
Operating mode
changing to offline, 136
changing to RUN, 82, 136
changing to STOP, 83, 136
read PLC information, 169
startup options, 192
Operator stations, 139
Options
STL status, 742
OR< (STL-OR compare real less than), 297
OR<= (STL-OR compare real less than or equal), 297
OR<> (STL-OR compare real not equal), 297
OR= (STL-OR compare real equal), 297
OR> (STL-OR compare real greater than), 297
OR>= (STL-OR compare real greater than or equal), 297
ORB (STL-NOR byte), 407
ORB (STL-OR byte), 407
ORD (STL-OR dword), 407
Ordering information, 1121
ORW (STL-OR word), 407
OS<> (STL-OR string compare not equal), 301
OS= (STL-OR string compare equal), 301
OUC library example
Active partner (client) program, 640
CheckErrors subroutine program, 649
Passive partner (server) program, 651
OUC library example Active partner (client) program
variable table, 650
OUC library example Passive partner (server) program
Variable table, 658
Output image register, 108
Outputs
coils, 249
example bit logic, 255
physical and in program, 107
set and reset bits, 250
set and reset dominant bistable, 251
writing, 109
Overvoltage, 984
OW< (STL-OR compare word less than), 297
OW<= (STL-OR compare word less than or equal), 297
OW<> (STL-OR compare word not equal), 297
OW= (STL-OR compare word equal), 297
OW> (STL-OR compare word greater than), 297
OW>= (STL-OR compare word greater than or equal), 297

P

Password
lost or forgotten, 210
privilege levels, 49
Password protection, 49
POUs and data blocks, 63
projects, 62
PID auto-tune
exception conditions, 763
prerequisites, 761
PV out-of-range, 764
sequence, 761
PID loop control
loop definition table, 755
PID Tune control panel, 765
PID loop instruction
alarm checking, 392
loop control types, 388
understanding, 386
PID Tune control panel, 765
Pin assignments for network connector, 564
Pipelining
PTO pulses, 353
PLC
clearing memory, 208
compile and run-time errors, 1062
expansion cable, 99
fatal errors, 1065
installation, 90
installation on a panel, 91
memory card, 132
non-fatal error memory locations, 1064
system block, 181
viewing PLC information, 169
PLC menu
Download, 124
Upload, 127
PLS
instruction, 351
Override, 352
Special Memory to monitor and control PTO and PWM outputs, 1080
PN Read Write library
PN_RD_REC instruction, 659
PN_WR_REC instruction, 659

PN Read Write Record library
overview, 573
Pointer
creating pointers and using indirect address, 121
example of pointer offset to access data, 124
example of pointer to access data in a table, 123
POUs
password protection, 63
viewing properties, 63
Power interruption (PLC), 190
Power requirements
calculating, 1059
CPU, 87, 1057
sample, 1058
Power supply, 87, 1057
PPI communication
changing to Freeport mode, 271
multi-master and multi-slave PPI networks, 556
port configuration in system block, 60, 183
single-master networks, 555
Prerequisite
Using SINA_PARA_S instruction, 727
Using SINA_POS instruction, 680
Using SINA_SPEED instruction, 720
Previous STEP 7-Micro/WIN projects, 143
PROFIBUS
DP device, 530
S7-200 SMART EM DP01 PROFIBUS DP
module, 531
Profile table values
PTO generators, 358
PROFINET
device naming and addressing, 501
PROFINET device status
SMB1800-SMB1999, 1104
PROFINET instruction
BLKMOV_BIR, 474
BLKMOV_BIW, 474
BMIR, 474
BMIW, 474
RDREC, 470
WRREC, 470
Program block, 141
Program control instructions
END, STOP, and WDR, 425
FOR-NEXT loop, 416
GET_ERROR, 426
JMP-LBL, 417
SCR (Sequence control relay), 418
Program editor
bookmarks, 737
debugging and monitoring, 739
STL status options, 742
types, 150
using, 77
Program instruction
bit logic, 253
Program instructions
bit logic, 241, 242, 244, 245, 247, 248, 249, 250, 251, 252, 253, 255
clock, 257, 259
compare, 297, 301
convert, 302, 305, 311, 315, 317
counters, 322, 325, 330, 340
frequency mode, 347
function block, 463
Function block, 465
interrupts, 393
logical operations, 319, 376, 377, 406, 407
math, 362, 366, 368, 370, 371
move, 411, 412, 413, 414, 415
program control, 416, 417, 418, 425, 426
shift and rotate, 428, 433
string, 437, 438
subroutine, 458, 459
table, 441, 442, 444
table find, 445
timer, 449, 457
Program status
building a status chart, 742
displaying in program editor, 739
executing a limited number of scans, 746
Program transfer card, 129
Programs
elements, 141
executing limited scans, 746
interrupt routines, 142
memory card, 132, 133
status charts, 742
subroutines, 141
Projects
Migrate, 143
password protection, 62
Properties
POUs and DBs, 63
Proportional term, PID algorithm, 387
Protection class, 984
Protocols
PROFIBUS DP, 531
PTO
overriding, 353

PTO0, PWM0, PTO1, PWM1, PTO2, PWM2, PTO3 and PWM3 high-speed outputs (SMB66-SMB85, SMB166-SMB169, SMB176-SMB179, SMB566-SMB579 and SMB580-SMB593), 1080
Pulse catch bits, digital input configuration in system block, 187
Pulse train output (PTO)
    cycle time, 352
    instruction, 211
    pulse output instruction (PLS), 351
Pulse width modulation (PWM)
    cycle time, 354
    instruction, 211
    output, 880, 881
    pulse output instruction (PLS), 351
    pulse outputs, 882
PUT, 262
PWM wizard
    PWMx_RUN subroutine, 882
PWMx_RUN, 882

Q

Queue interrupt overflow (SMB4), 1071
quick access toolbar, 30

R

R (STL-Reset), 250
R_S, 311
RAND_B, 373
RAND_DI, 373
RAND_I, 373
RAND_R, 373
Rated voltages, 984
RCV (receive message control SMB86-SMB94, SMB405-SMB413 and SMB186-SMB194), 1087
READ_RTC, 257
READ_RTCX, 259
Real number values, 119
Receive instruction
    break detection, 277
    end character detection, 278
    end conditions, 275
    idle line detection, 275
    intercharacter timer, 278
    maximum character count, 279
    message timer, 279
    parity errors, 279
    start character detection, 276
    user termination, 279
Recursion in subroutines/FBs, 458, 464
Recycling and disposal, 67
Referencing (active referencing) mode, 703
Referencing (set reference point) mode), 706
Relative positioning mode, 691
Relay electrical service life, 985
Repeaters, 563
Reset-to-factory-defaults memory card, 210
Restoring data after power-on, 136
RET, 458, 463
Retentive memory, 128
Retentive ranges, system block configuration, 190
RETI, 393
RI (STL-Reset immediate), 250
ROUND, 302
RS (LAD/FBD Reset dominant bistable), 251
RS232, 570
    Freeport mode, 569
    number of communication connections, 476
    types of communication, 29, 475
RS232/PPI cable, 569
RS485
    communication ports configuration, 60, 183
    number of communication connections, 476
    sample network configurations, 555
    setting baud rate and port network address, 553
    types of communication, 29, 475
RS485 address
    assigning, 558
RS485 communications
    STEP 7-Micro/WIN SMART settings, 75
RS485 network
    address, 557
    configuring the RS485 network address for a CPU, 557
    searching for CPU, 561
RS485 network address
    configuring, 557
RTA, 305
RTD analog inputs
    coefficient, 198
    rejection, 198
    resistor, 198
    RTD type, 198
    scale, 198
    smoothing, 198
    system block configuration, 198
RTS (STL-Real to string), 311
RUN mode, 82, 136
RUN to STOP transition
    analog output states, 196
    digital output states, 189

Run-time and PLC compile errors, 1062

S

S (STL-Set), 250
S_DI, 315
S_I, 315
S_ITR, 319
S_R, 315
S_RTI, 319
S_RTR, 319
S7-200 SMART security
  Network security, 45
S7-200 SMART
  as PROFIBUS slave device, 532
Safety circuits, 139
Sample control program, 77
Sample network configurations, RS485 devices, 555
Saving project, 81
SB (signal board) hardware/firmware ID (SMB1050-SMB1099), 1099
Scan cycle
  executing a single scan, 746
  executing multiple scans, 746
  Scan rates (PLC information), 169
  scan times (SMW22-SMW26), 1075
SCR, 418
SCRE, 418
SCRT, 418
Security, 49
SEG, 302
Selecting the network cable, 564
Service and support, 4
Set and reset dominant bistable instructions, 251
Set and reset immediate instructions, 250
SET_ADDR, 282
SET_RTC, 257
SET_RTCX, 259
Setting the CPU type, 80
Setup mode, 699
SFND (STL-Find string), 438
Shift and rotate instructions
  bit (SHRB), 433
  byte, word, dword, 428
SHL_B, 428
SHL_DW, 428
SHL_W, 428
SHR_B, 428
SHR_DW, 428
SHR_W, 428
SHRB, 433
SI (STL-set immediate), 250
Siemens technical support, 4
Siemens-supplied libraries, 573
Signal board (SB
  SB BA01 specifications, 1047
Signal board (SB)
  installing and removing, 94
  SB AQ01 specifications, 1043
  SB AQ01 wiring diagram, 1044
  SB BA01 wiring diagram, 1047
  SB CM01 specifications, 1045
  SB CM01 wiring diagram, 1046
  SB DT04 specifications, 1038
  SB DT04 wiring diagram, 1040
Signal board ID and error registers (SMB28-SMB29), 1075
Signal board ID and error registers (SMB32-SMB33), 1076
Signal boards (SB)
  analog output representation (current), 1027
  analog output representation (voltage), 1027
  input representation (current), 1026
  input representation (voltage), 1026
Signal modules (SM)
  step response times, 1025
SIN (sine), 368
SINA_PARA_S instruction
  example, 734
SINA_POS instruction
  Absolute positioning mode example, 696
  Incremental jog mode example, 716
  Jog mode example, 713
  operating mode selection, 690
  Referencing (active referencing) mode example, 704
  Referencing (set reference point) mode example, 707
  Relative positioning mode example, 692
  Setup mode example, 701
  Traversing blocks mode example, 710
SINA_SPEED instruction
  example, 725
SINAMICS library
  overview, 573
  SINA_PARA_S instruction, 730
  SINA_POS instruction, 683
  SINA_SPEED instruction, 723
  SINAMICS_Control, 678
  SINAMICS_Parameter, 679
SINAMICS library features, 678
SIP_ADDR, 283
SLB (STL-Shift left byte), 428
SLD (STL-Shift left dword), 428

SLW (STL-Shift left word), 428
SM (Special memory) assignments and functions, 1067
SM locations (Axis of Motion), 948
SM memory, PTO/PWM operation, 355
SMB0 system status bits, 1069
SMB1 instruction execution status bits, 1070
SMB1000-SMB1049 CPU hardware/firmware ID, 1099
SMB1050-SMB1099 SB (signal board) hardware/firmware ID, 1099
SMB1100-SMB1299 EM (expansion module) hardware/firmware ID, 1100
SMB130 Port 1 configuration, 1077
SMB136-SMB145 (HSC3) high-speed counter 3, 1078
SMB146-SMB155 (HSC4) high-speed counter 4, 1078
SMB156-SMB165 (HSC5) high-speed counter 5, 1078
SMB1800-SMB1999 PROFINET device status, 1104
SMB186-SMB194 receive message control, 1087
SMB195 to SMB210
    HSC, 1092
SMB2 Freeport receive character, 1070
SMB2000-SMB2014: PTO4/PWM4 high-speed outputs, 1081
SMB28-SMB29 signal board ID and error registers, 1075
SMB3 Freeport character error, 1071
SMB30 (Port 0), SMB31(Port2) and SMB130 (Port 1) configuration, 1077
SMB32-SMB33 signal board ID and error registers, 1076
SMB34-SMB35 time interval values for timed interrupts, 1078
SMB360-SMB369 (HSC6) high-speed counter 6, 1078
SMB36-SMB45 (HSC0) high-speed counter 0, 1078
SMB370-SMB379 (HSC7) high-speed counter 7, 1078
SMB380 to SMB404
    HSC, 1093
SMB4 interrupt queue overflow, run-time program error, interrupts enabled, Freeport transmitter idle, value forced, 1071
SMB46-SMB55 (HSC1) high-speed counter 1, 1078
SMB480-SMB515 Data log status, 1094
SMB5 I/O error status bit, 1072
SMB566-SMB579: PTO2 and PWM2 high-speed outputs, 1080
SMB56-SMB65 (HSC2) high-speed counter 2, 1078
SMB580-SMB593: PTO3 and PWM3 high-speed outputs, 1080
SMB580-SMB593: PTO3/PWM3 high-speed outputs, 1080
SMB600-SMB799 Axis 0,1,2 and 3 motion control,
SMB66-SMB85, SMB166-SMB169, and SMB176-SMB179: PTO0, PWM0, PTO1, and PWM1 high-speed outputs,
SMB66-SMB85, SMB166-SMB169, SMB176-SMB179, SMB566-SMB579 and SMB580-SMB593: PTO0, PWM0, PTO1, PWM1, PTO2, PWM2, PTO3 and PWM3 high-speed outputs, 1080
SMB6-SMB7 CPU ID register, 1073
SMB86-SMB94 and SMB186-SMB194 receive message control, 1087
SMB8-SMB19 I/O module ID and error registers, 1073
SMB950-SMB998 SB (signal board) hardware/firmware ID, 1099
SMW100-SMW120 System alarms, 1091
SMW22-SMW26 scan times, 1075
SMW98 Expansion I/O bus - communication errors, 1091
Software debugging, 737
SORT_B, 374
SORT_DI, 374
SORT_I, 374
SORT_R, 374
Special memory assignments and functions, 1067
Special memory bytes
    EM DP01 PROFIBUS DP, 542
Specifications, (SB BA01)
    analog input representation (current), 1026
    analog input representation (voltage), 1026
    analog output representation (current), 1027
    analog output representation (voltage), 1027
    CE approval, 981
    contamination level, 984
    CPU SR40, 986
    CPU SR60, 994
    CPU ST40, 986
    CPU ST60, 994
    electromagnetic compatibility (EMC), 982
    EM AE04, 1016
    EM AM06, 1021
    EM AQ02, 1019
    EM AR02 (RTD), 1033
    EM AT04, 1028
    EM DP01 PROFIBUS DP, 1048
    EM DR08, 1006
    EM DR16, 1010
    EM DR32, 1010
    EM DT08, 1006
    EM DT16, 1010
    EM DT32, 1010
    EM QR16, 1006
    EM QT16, 1006
    environmental conditions, 983

General technical specifications, 981
industrial environments, 982
insulation, 984
overvoltage, 984
protection, 984
rated voltages, 984
relay electrical service life, 985
SB AE01, 1040
SB AQ01, 1043, 1044
SB CM01, 1045
SB DT04, 1038, 1040
step response times (SM), 1025
UK Conformity Assessed marking, 981
SQRT (square root), 368
SR (LAD/FBD Set dominant bistable), 251
SRB (STL-Shift right byte), 428
SRD (STL-Shift right dword), 428
SRW (STL-Shift right word), 428
SSCPY (STL-Copy substring), 437
SSTR_CPY, 437
Starting
startup options, 192
Starting Ethernet communications
STEP 7-Micro/WIN SMART, 71
Starting RS485 communications
STEP 7-Micro/WIN SMART, 75
Statement List editor (STL), 152
Static IP information, 483
Status
building a status chart, 742
displaying in program editor, 739
executing a limited number of scans, 746
Status error (timestamp mismatch), 1061
Status LEDs
CPU, 137
EM DP01 PROFIBUS DP, 543
Expansion modules (EMs), 137
PROFINET network, 504
STD (STL-Sub-string to double integer), 315
STEP 7-Micro/WIN (earlier versions), 143
STEP 7-Micro/WIN SMART
connecting with the CPU, 73, 75, 76, 484, 559
equipment requirements, 30
Ethernet port configuration, 483
RS485 network port configuration, 557
RUN and STOP mode, 82, 136
Steppers in motion control, 882
STI (STL-Sub-string to integer), 315
STL
logic stack operations, 245
status options, 742
STL editor, 152
STOP (instruction), 425
STOP mode, 83, 136
analog output states, 196
digital output states, 189
writing and forcing outputs, 745
STR (STL-Sub-string to real), 315
STR_FIND, 438
String instructions
copy substring, 437
Find string / character, 438
Strings
format, 119
representation, 120
SUB_DI, 362
SUB_I, 362
SUB_R, 362
Subroutine instructions
Call parameter and return examples, 459
CALL, RET, 458
Subroutines
Axes group of Motion, 962
Axis of Motion, 903
element of user program, 141
guidelines, 904
PWMx_RUN, 882
Subroutines/FBs
nesting depth, 458, 464
total number, 458, 464
Support, 4
Suppression circuits, 104
SWAP, 414
Synchronous updates (PWM instruction), 355
System alarms (SMW100-SMW120), 1091
System block, 142
BA01 battery signal board, 207
CPU configuration, 181
digital input filters, 187
IP address of CPU, 485
password and security, 49
RS485 network address of CPU, 557
RS485/RS232 CM01 communications signal board, 205
RTD analog input module, 198
startup options, 192
TC analog input module, 202
System status bits (SMB0), 1069

T

Table instructions
ATT, 441
FIFO/LIFO, 442

FILL_N, 444
TBL_FIND, 445
TAN (tangent), 368
TBL_FIND, 445
TC analog input module, 202
TC analog inputs
rejection, 202
scale, 202
smoothing, 202
system block configuration, 202
TC type, 202
Technical specifications, 981
Technical support, 4
Terminal block connector, 95
Thermal zone, 87
Thermocouple
basic operation, 205, 1030
cold junction compensation, 205, 1030
EM AT04 Thermocouple filter selection table, 1031
EM AT04 Thermocouple selection table, 1031
Timed interrupt configuration (SMB34-SMB35), 1078
Time-of-day
clock instructions, 257
extended clock instructions, 259
protection for reads and writes, 49
Timer instructions
BITIM, CITIM, 457
interrupts, 399
programming tips and examples, 451
TON, TONR, TOF, 449
Timestamp mismatch (PC/PLC program difference), 1061
TODR (STL-Read time-of-day clock), 257
TODRX (STL-Read time-of-day clock extended), 259
TODW (STL-Write time-of-day clock), 257
TODWX (STL-Write time-of-day clock extended), 259
TOF (Off-delay timer), 449
TON (On-delay timer), 449
TONR (On-delay timer retentive), 449
Tools (options)
execution status coloring, 739
STL status, 742
Tools menu
Find PROFINET Devices, 502
PID Tune control panel, 765
Transfer card, 129
Transmission rate, 562
Transmit instruction
example, 281
transmitting data, 272
Traversing blocks mode, 709
Troubleshooting
LED indicators, 137
Troubleshooting S7-200 SMART hardware, 748
TRUNC, 302

U

UK Conformity Assessed marking, 981
Updating firmware, 169
Uploading programs, 127
User-defined libraries, 575
USS protocol instructions
example program, 675
using, 664
USS_CTRL, 667
USS_INIT, 665
USS_RPM_x, 670
USS_WPM_x, 673
USS protocol library
calculating time for communications, 663
execution errors, 677
overview, 573, 662
requirements, 663
using the USS protocol instructions, 664

V

V90 drives, 887, 917
variables (variable addressing)
indirect addressing, 158
Version (firmware), 169
Vibration, 92

W

WAND_B, 407
WAND_DW, 407
WAND_W, 407
WDR (watchdog timer reset), 425
Web API, 239
Web server
Watch table, 216
Wiring diagram
EM AT04, 1028
wiring diagrams
EM DE16, 1005
Wiring diagrams
CPU CR40, 992
CPU CR60, 1001
CPU SR40, 992
CPU SR60, 1001

CPU ST40, 992
CPU ST60, 1001
EM AE04, 1018
EM AM06, 1024
EM AQ02, 1020
EM AQ04, 1020
EM AR02 (RTD), 1037
EM DE08, 1005
EM DP01 PROFIBUS DP module, 1051
EM DR08, 1008
EM DR16, 1013
EM DR32, 1014
EM DT08, 1008
EM DT16, 1013
EM DT32, 1014
SB AQ01, 1044
SB BA01, 1047
SB CM01, 1046
Wiring guidelines, 102
clearance for airflow and cooling, 87
DIN rail, 92
grounding, 102
grounding and circuit, 100
inductive loads, 104
installation, 85
isolation, 101
lamp loads, 103
separate the devices from heat, high voltage, and electrical noise, 86
terminal block connector, 95
Wizards
high-speed counter (HSC), 330
Text Display, 28
WOR_B, 407
WOR_DW, 407
WOR_W, 407
Word access, 112
Word consistency
PROFIBUS, 534
Writing values
outputs, 109
writing and forcing outputs in STOP mode, 745
WXOR_B, 407
WXOR_DW, 407
WXOR_W, 407

X

XORB (STL-XOR byte), 407
XORD (STL-XOR dword), 407
XORW (STL-XOR word), 407