SIEMENS

S7-200 SMART G2 System Manual

System Manual

Preface

Product overview 1

Security information 2

Getting started 3

Installation 4

PLC concept 5

Programming concepts 6

PLC device configuration 7

Web server 8

Program instructions 9

Communication 10

Libraries 11

Debugging and troubleshooting 12

PID loops and tuning 13

Motion control(PLCopen) 14

Open loop motion control(Legacy) 15

Technical specifications A

Calculating a power budget B

Error code C

Special memory (SM) and system symbol names D

Reference E

Ordering information F

03/2026
A5E55110929-AB

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 persons are those who, because of their training and experience, are familiar with the installation, assembly, commissioning, operation, decommissioning and disassembly of the product and can recognize risks and avoid possible hazards.

Proper use of Siemens products

Note the following:

⚠ WARNING

Siemens products may only be used for the application described in the catalog and the associated usage information. 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

A5E55110929-AB
Ⓟ 03/2026 Subject to change

Copyright © Siemens 2026.
All rights reserved

Preface

Preface

Purpose of the documentation

The S7-200 SMART G2 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 G2 a perfect solution for controlling small applications. The wide variety of S7-200 SMART G2 models and the Windows-based programming tool give you the flexibility you need to solve your automation problems.

To meet diverse application needs, the S7200 SMART G2 CPU series will offer multiple models, each with unique features and functions designed to help users build efficient solutions. Additional models will be introduced gradually.

This manual provides information about installing and programming the S7-200 SMART G2 CPUs and is designed for engineers, programmers, installers, and electricians who have a general knowledge of programmable logic controllers.

Basic knowledge required

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

Validity of this document

This document applies to the S7-200 SMART G2 CPU series and expansion modules.

For details on the individual CPUs and modules in the S7-200 SMART G2 series, please refer to technical specifications (Page 1146).

ID Link for the digital nameplate



The ID link is a globally unique identifier according to IEC 61406-1 and is displayed on the product as a QR code.

The figure shows an example of an ID Link for the CPU .

The ID link can be identified by the box with a black border in the lower right corner. The product's digital nameplate can be accessed via an ID link.

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

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

Service and support

Certificates and certifications For detailed information on certificates and certifications, please refer to the technical specifications.

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.

Certifications and approvals

For detailed information on certificates and certifications, please refer to the technical specifications (Page 1146).

Table of contents

Preface **3**

1 Product overview **23**

1.1 S7-200 SMART G2 CPU 24

1.2 What's new? 26
1.2.1 What's new in S7-200 SMART V3.2? 26
1.2.2 What's new in S7-200 SMART V3.1? 28

1.3 S7-200 SMART G2 expansion module 29

1.4 HMI devices for S7-200 SMART G2 31

1.5 Communications options 32

1.6 Programming software 33

2 Security information **35**

2.1 Safety information 35
2.1.1 Warnings in this document 35
2.1.2 Safety-related symbols 35
2.1.2.1 Devices without explosion protection 35
2.1.2.2 Devices with explosion protection 36
2.1.3 Intended use 37
2.1.4 Changes to the device and spare parts 37
2.1.5 Knowledge and personnel qualification 38
2.1.6 Personal protective equipment 38
2.1.7 Open Source Software 38
2.1.8 Safe working 39
2.1.8.1 Working on electrical parts 39
2.1.9 Behavior in case of emergency 40
2.1.10 Residual risks 40
2.1.10.1 Residual risks 40
2.1.10.2 Live parts 40
2.1.10.3 External power supply to modules 41
2.1.10.4 Voltage dangerous to touch due to permissible electric potential difference 41
2.1.10.5 Conductive pollution 42
2.1.10.6 Overheating 42
2.1.10.7 Uncontrolled movements 42
2.1.10.8 Unsafe operating states 42
2.1.11 Material damage 43
2.1.11.1 Transport and storage 43
2.1.11.2 Installation and connection 43

2.2 Industrial cybersecurity 43
2.2.1 Cybersecurity information 43
2.2.2 Safety instructions 44
2.2.3 Security overview 46
2.2.4 Network security 47

2.2.5 Communications portocols and port numbers used for Ethernet communication 49
2.2.6 Configuring communication 52
2.2.7 CPU access protection 54
2.2.8 Configuring PLC security 55
2.2.8.1 PLC protection (Legacy) 56
2.2.8.2 Enable User Management 59
2.2.8.3 Communicaiton setting 63
2.2.9 Protect the project 64
2.2.9.1 Set Project Password 64
2.2.9.2 POU / data page password protection 65
2.2.10 Secure decommissioning 69
2.2.10.1 Decomissioning introduction 69
2.2.10.2 Securely removing data 69
2.2.10.3 Recycling and disposal 69

3 Getting started 71

3.1 Connecting to the CPU 71
3.1.1 Configuring the CPU for communication 72
3.1.1.1 Overview 72
3.1.1.2 Establishing the Ethernet hardware communication connection 73
3.1.1.3 Setting up Ethernet communication with the CPU 73

3.2 Creating a sample program 76
3.2.1 Network 1: Starting the timer 77
3.2.2 Network 2: Turning the output on 78
3.2.3 Network 3: Resetting the timer 79
3.2.4 Setting the CPU type and version for your project 79
3.2.5 Saving the sample project 80

3.3 Downloading the sample program 81

3.4 Changing the operating mode of the CPU 81

4 Installation 83

4.1 Guidelines for installing S7-200 SMART G2 devices 83

4.2 Power budget 87

4.3 Installation and removal procedures 89
4.3.1 S7-200 SMART G2 devices installation dimensions 89
4.3.2 Installing and removing the CPU 90
4.3.3 Installing and removing a signal board 93
4.3.4 Installing and removing an expansion module 94
4.3.5 Labeling S7200 SMART G2 96
4.3.5.1 Attaching the identification label 96

4.4 Wiring guidelines 97
4.4.1 Prerequisites 97
4.4.2 I/O terminal wiring 104

4.5 Maintenance 106
4.5.1 Replacing terminal block 106

5 PLC concept 109

5.1 Execution of the control logic 109
5.1.1 Reading the inputs and writing to the outputs 110

5.1.2 Immediate reading or writing of the I/O ... 111
5.1.3 Executing the user program ... 111

5.2 accessing data ... 113
5.2.1 Accessing memory areas ... 114
5.2.2 Format for Real numbers ... 121
5.2.3 Format for Strings ... 121
5.2.4 Assigning a constant value for instructions ... 122
5.2.5 Addressing the local and expansion I/O ... 122
5.2.6 Using Pointers for indirect addressing ... 123
5.2.7 Pointer examples ... 125

5.3 Saving and restoring data ... 126
5.3.1 Downloading project components ... 126
5.3.2 Uploading project components ... 129
5.3.3 Type of energy storage ... 130
5.3.4 Using a memory card ... 131
5.3.5 Inserting a memory card in a standard CPU ... 133
5.3.6 Transferring your program with a memory card ... 133
5.3.7 Restoring data after power-on ... 138

5.4 Changing the operating mode of the CPU ... 138

5.5 Status LEDs ... 138

6 Programming concepts ... 141

6.1 Guidelines for designing a PLC system ... 141

6.2 Elements of the user program ... 143

6.3 Creating your user program ... 145
6.3.1 Earlier versions of STEP 7-Micro/WIN projects ... 145
6.3.2 Using STEP 7-Micro/WIN SMART user interface ... 149
6.3.3 Using STEP 7-Micro/WIN SMART to create your programs ... 152
6.3.4 Using wizards to help you create your control program ... 153
6.3.5 Ladder logic editor ... 154
6.3.6 Function block diagram editor ... 155
6.3.7 Statement list editor ... 155

6.4 Data block (DB) editor ... 157

6.5 Global variable table ... 159

6.6 Local variable table ... 163

6.7 User-defined data types ... 167
6.7.1 UDT table ... 169

6.8 PLC error reaction ... 172
6.8.1 System information ... 173
6.8.1.1 System ... 173
6.8.1.2 CPU ... 174
6.8.1.3 PROFINET device ... 175
6.8.2 Event Log ... 177
6.8.3 PROFINET Alarm ... 178
6.8.4 Scan rates ... 179
6.8.5 Non-fatal errors and I/O errors ... 179
6.8.6 fatal errors ... 181

6.9 Program edit in RUN mode ... 181
6.10 Features for debugging your program ... 184
6.11 Trace ... 184
6.11.1 Introduction ... 184
6.11.2 User interface ... 186
6.11.3 Trace timing diagram interface ... 188
6.11.4 Configuring the trace ... 192
6.11.4.1 New trace ... 193
6.11.4.2 Configuring a signal table ... 193
6.11.4.3 Configuring sampling conditions ... 195
6.11.5 Managing traces in the project ... 199
6.11.5.1 Opening a trace ... 199
6.11.5.2 Importing/export inga trace ... 200
6.11.6 Managing traces in PLC ... 201
6.11.6.1 Downloading/uploading traces ... 201
6.11.6.2 Activating/deactivating a trace ... 202
6.11.6.3 Trace status table ... 203
6.11.6.4 Deleting the trace configuration and data in the PLC ... 204

7 PLC device configuration ... 205

7.1 Configuring the operation of the PLC system ... 205
7.1.1 System block ... 205
7.1.2 Configuring communication ... 208
7.1.3 Configuring the digital inputs ... 211
7.1.4 Configuring the digital outputs ... 214
7.1.5 Configuring the retentive ranges ... 215
7.1.6 Configuring the startup options ... 217
7.1.7 Motion control ... 218
7.1.8 Configuring the analog inputs ... 220
7.1.9 Configuring the analog outputs ... 223
7.1.10 Configuring the RTD analog inputs ... 225
7.1.11 Configuring the TC analog inputs ... 229
7.1.12 Configuring the RS485/RS232 CM01 communications signal board ... 232
7.1.13 Clearing PLC memory ... 233
7.1.14 Creating a reset-to-factory-defaults memory card ... 235
7.2 High-speed I/O ... 235

8 Web server ... 237

8.1 Web server Overview ... 237
8.2 Downloading and installing the certificate of CA ... 239
8.2.1 Get the certificate of CA ... 239
8.2.2 Installing the certificate for Windows ... 240
8.2.3 Installing the certificate for Mac OS ... 241
8.2.4 Installing the certificate for mobile devices ... 241
8.3 Accessing Web server from PC or mobile device ... 241
8.4 Standard Web pages ... 243
8.4.1 Layout of web pages ... 243
8.4.2 Introduction ... 244
8.4.3 Start Page ... 245

8.4.4 Event Log ... 246
8.4.5 Status chart ... 246
8.4.6 Watch table ... 247
8.4.7 Data log ... 247
8.4.8 Set clock ... 248
8.4.9 Module information ... 249

8.5 User defined Web pages ... 250
8.5.1 Downloading user-defined Web resource files ... 250

8.6 Web API (Application Programming Interface) ... 252

9 Program instructions ... 255

9.1 Bit logic ... 255
9.1.1 Standard inputs ... 255
9.1.2 Immediate inputs ... 256
9.1.3 Logic stack overview ... 257
9.1.4 STL logic stack instructions ... 259
9.1.5 NOT ... 261
9.1.6 Positive and negative transition detectors ... 262
9.1.7 Coils: output and output immediate instructions ... 263
9.1.8 Set, reset, set immediate, and reset immediate functions ... 263
9.1.9 Set and reset dominant bistable ... 265
9.1.10 NOP (No operation) instruction ... 266
9.1.11 Alternating output and alternating output on pulse ... 267
9.1.12 Edge detector ... 268
9.1.13 Bit logic input examples ... 269
9.1.14 Bit logic output examples ... 270

9.2 Clock ... 272
9.2.1 Read and set real-time clock ... 272
9.2.2 Read and set real-time clock extended ... 274

9.3 Communications ... 277
9.3.1 GET and PUT (Ethernet) ... 277
9.3.2 Transmit and receive (Freeport on RS485/RS232) ... 285
9.3.3 Get port address and set port address (PPI protocol on RS485/RS232) ... 297
9.3.4 Get IP address and set IP address (Ethernet) ... 298
9.3.5 Open user communication ... 300
9.3.5.1 OUC instructions ... 300
9.3.5.2 OUC instruction error codes ... 311

9.4 Compare ... 312
9.4.1 Compare number values ... 312
9.4.2 Compare character strings ... 315

9.5 Convert ... 317
9.5.1 Standard conversion instructions ... 317
9.5.2 ASCII character array conversion ... 320
9.5.3 Number value to ASCII string conversion ... 326
9.5.4 ASCII sub-string to number value conversion ... 330
9.5.5 Encode and decode ... 332
9.5.6 Scale ... 334

9.6 Counters ... 336
9.6.1 Counter instructions ... 336

9.6.2 Counter instructions (IEC) ... 340
9.6.3 High-speed counter instructions ... 344
9.6.4 High-speed counter summary ... 346
9.6.5 Noise reduction for high-speed inputs ... 346
9.6.6 High-speed counter programming ... 348
9.6.7 Example initialization sequences for high-speed counting mode ... 359
9.6.8 Example initialization sequences for high-speed frequency mode ... 366

9.7 Pulse output ... 370
9.7.1 Pulse output instruction (PLS) ... 370
9.7.2 Pulse train output (PTO) ... 371
9.7.3 Pulse width modulation (PWM) ... 373
9.7.4 Using SM locations to configure and control the PTO/PWM operation ... 374
9.7.5 Calculating the profile table values ... 377

9.8 Math ... 381
9.8.1 Add, subtract, multiply, and divide ... 381
9.8.2 Multiply integers to double integer and divide integer with remainder ... 385
9.8.3 Trigonometry, natural logarithm/exponential, and square root ... 387
9.8.4 Increment and decrement ... 389
9.8.5 Get the maximum or minimum value ... 390
9.8.6 Get a random value ... 392
9.8.7 Sort the array ... 393
9.8.8 Get the average of a array ... 394
9.8.9 Get absolute values ... 395
9.8.10 Low-pass filter ... 396
9.8.11 PID loop instruction ... 397

9.9 Interrupt ... 398
9.9.1 Interrupt instructions ... 398
9.9.2 Interrupt routine overview and CPU model event support ... 400
9.9.3 Interrupt programming guidelines ... 402
9.9.4 Types of interrupt events that the S7-200 SMART CPU supports ... 404
9.9.5 Interrupt priority, queuing, and example program ... 405

9.10 Logical operations ... 413
9.10.1 Invert ... 413
9.10.2 AND, OR, XOR, NAND, NOR, and NXOR ... 414

9.11 Move ... 418
9.11.1 Move ... 418
9.11.2 Move byte, word, double word, or real ... 419
9.11.3 Block move ... 420
9.11.4 Swap bytes ... 421
9.11.5 Move byte immediate (read and write) ... 422

9.12 Program control ... 422
9.12.1 FOR-NEXT loop ... 422
9.12.2 JMP (jump to label) ... 424
9.12.3 SCR (sequence control relay) ... 425
9.12.4 END, STOP, and WDR (watchdog timer reset) ... 432
9.12.5 GET_ERROR (Get non-fatal error code) ... 433

9.13 Shift and rotate ... 435
9.13.1 Shift and rotate ... 435
9.13.2 Shift register bit ... 440

9.14 String ... 442
9.14.1 String (Get length, copy, and concatenate) ... 442
9.14.2 Copy substring from string ... 444
9.14.3 Find string and first character within string ... 445

9.15 Table ... 448
9.15.1 Add to table ... 448
9.15.2 First-in-first-out and last-in-first-out ... 449
9.15.3 Memory fill ... 451
9.15.4 Table find ... 452

9.16 Timers ... 456
9.16.1 Timer instructions ... 456
9.16.2 Timer programming tips and examples ... 458
9.16.3 Timer instructions (IEC) ... 464
9.16.4 Timer programming tips and examples (IEC) ... 465
9.16.5 Interval timers ... 467

9.17 Subroutine ... 468
9.17.1 CALL (subroutine) and RET (conditional return) ... 468
9.17.2 Subroutine call parameters and return examples ... 470

9.18 Function block ... 474
9.18.1 CALL (function block) and RET (conditional return) ... 474
9.18.2 Function block call parameters ... 476

9.19 PROFINET ... 481
9.19.1 Features of the programming instruction "PROFINET" ... 481
9.19.2 Read and Write data record ... 481
9.19.2.1 Input and output interface of RDREC and WRREC instruction ... 481
9.19.2.2 System-defined error code of the instructions RDREC and WRREC ... 483
9.19.3 Read and Write multiple bytes between physical PROFINET and memory address ... 485
9.19.3.1 Input and output interface of BLKMOV_BIR and BLKMOV_BIW ... 485
9.19.3.2 Error code of the instructions BLKMOV_BIR and BLKMOV_BIW ... 485

10 Communication ... 487

10.1 Overview ... 487

10.2 CPU communication connections ... 488

10.3 CPU communication ports ... 489

10.4 Ethernet ... 489
10.4.1 Local/partner connection ... 489
10.4.2 Sample Ethernet network configurations ... 490
10.4.3 Assigning Internet Protocol (IP) addresses ... 491
10.4.3.1 Assigning IP addresses to programming and network devices ... 491
10.4.3.2 Configuring or changing an IP address for a CPU or device in your project ... 492
10.4.3.3 Searching for CPUs and devices on your Ethernet network ... 499
10.4.4 Locating the Ethernet (MAC) address on the CPU ... 500
10.4.5 HMI-to-CPU communication ... 501
10.4.6 Open user communication ... 503
10.4.6.1 Protocols ... 503
10.4.6.2 Connections ... 504
10.4.6.3 Ports and TSAPs ... 504
10.4.7 PROFINET Communication ... 506

10.4.7.1 Introduction ... 506
10.4.7.2 I-Device ... 506
10.4.7.3 Device database file in XML: GSDML ... 509
10.4.7.4 PROFINET device naming ... 509
10.4.7.5 LED status indicators for PROFINET network ... 509
10.4.7.6 Configuration ... 509

10.5 RS485 ... 526
10.5.1 Communication selection ... 526
10.5.1.1 Overview ... 526
10.5.1.2 PPI protocol ... 527
10.5.2 Communication basics ... 528
10.5.2.1 Basics of S7-200 SMART network communications ... 528
10.5.2.2 Definition of baud rate and network address ... 528
10.5.2.3 Setting the baud rate and network address for the S7-200 SMART CPU ... 529
10.5.3 Sample RS485 network configurations ... 531
10.5.3.1 Single-master PPI networks ... 531
10.5.3.2 Multi-master and multi-slave PPI networks ... 531
10.5.4 Assigning RS485 addresses ... 532
10.5.4.1 Configuring or changing an RS485 address for a CPU or device in your project ... 532
10.5.5 Building your network ... 535
10.5.5.1 General guidelines ... 535
10.5.5.2 Building a network ... 535
10.5.5.3 Determining the distances, transmission rates, and cable lengths for your network ... 535
10.5.5.4 Repeaters on the network ... 536
10.5.5.5 Specifications for RS485 cable ... 537
10.5.5.6 connector pin assignments ... 537
10.5.5.7 biasing and terminating the network cable ... 538
10.5.5.8 Biasing and terminating the CM01 signal board ... 538
10.5.6 Freeport mode ... 539
10.5.6.1 Creating user-defined protocols with Freeport mode ... 539
10.5.6.2 Using the RS232/PPI Multi-Master cable and Freeport mode with RS232 devices ... 541
10.5.6.3 Switches of the RS232/PPI Multi-Master cable ... 542

10.6 RS232 ... 545

11 Libraries ... 547

11.1 Library types (Siemens and user-defined) ... 547

11.2 Overview of Modbus communication ... 549
11.2.1 Modbus addressing ... 549
11.2.2 Modbus read and write functions ... 551

11.3 Modbus RTU library instructions ... 552
11.3.1 Modbus RTU communication overview ... 552
11.3.1.1 Modbus RTU library features ... 552
11.3.1.2 Requirements for using Modbus instructions ... 553
11.3.1.3 Initialization and execution time for Modbus protocol ... 555
11.3.2 Modbus RTU master ... 555
11.3.2.1 Using the Modbus RTU master instructions ... 555
11.3.2.2 MBUS_CTRL / MB_CTRL2 / MB_CTRL3 instruction (initialize master) ... 557
11.3.2.3 MBUS_MSG / MB_MSG2 / MB_MSG3 instruction ... 559
11.3.2.4 Modbus RTU master execution error codes ... 562
11.3.3 Modbus RTU slave ... 564

11.3.3.1 Using the Modbus RTU slave instructions ... 564
11.3.3.2 MBUS_INIT instruction (initialize slave) ... 565
11.3.3.3 MBUS_SLAVE instruction ... 567
11.3.3.4 Modbus RTU slave execution error codes ... 568
11.3.4 Modbus RTU master example program ... 568
11.3.5 Modbus RTU advanced user information ... 570

11.4 Modbus TCP library ... 572
11.4.1 Modbus TCP library features ... 572
11.4.2 Modbus TCP client ... 574
11.4.2.1 MBUS_CLIENT instruction ... 574
11.4.2.2 Modbus TCP client execution error codes ... 577
11.4.3 Modbus TCP server ... 578
11.4.3.1 MBUS_SERVER instruction ... 578
11.4.3.2 Modbus TCP server execution error codes ... 579
11.4.4 Example: Modbus TCP application ... 580
11.4.5 Modbus TCP advanced user information ... 585
11.4.6 Modbus TCP general exception codes ... 587
11.4.7 Modbus TCP general communication exception codes ... 588

11.5 Open user communication library ... 589
11.5.1 Overview ... 589
11.5.2 Parameters common to the OUC library instructions ... 590
11.5.3 Open user communication library instructions ... 592
11.5.3.1 TCP_CONNECT instruction ... 592
11.5.3.2 ISO_CONNECT instruction ... 595
11.5.3.3 UDP_CONNECT instruction ... 598
11.5.3.4 TCP_SEND instruction ... 600
11.5.3.5 TCP_RECV instruction ... 602
11.5.3.6 UDP_SEND instruction ... 605
11.5.3.7 UDP_RECV instruction ... 608
11.5.3.8 DISCONNECT instruction ... 610
11.5.4 Open user communication library instruction error codes ... 612
11.5.5 Open user communication library example ... 613
11.5.5.1 Active partner (client) ... 613
11.5.5.2 CheckErrors subroutine ... 622
11.5.5.3 Active partner variable table ... 623
11.5.5.4 Passive partner (server) ... 624
11.5.5.5 CheckErrors subroutine ... 631
11.5.5.6 Passive partner variable table ... 631

11.6 PN Read Write Record library ... 632
11.6.1 PN Read Write Record features ... 632
11.6.2 Input and output interface of PN Read Write Record library ... 632
11.6.3 Definition of parameters for input signal "STATUS" ... 633
11.6.4 System_defined error code of the library PN Read Write Record ... 634

11.7 USS Protocol library instructions ... 635
11.7.1 USS communication overview ... 635
11.7.1.1 USS protocol overview ... 635
11.7.1.2 Requirements for using the USS protocol ... 636
11.7.1.3 Calculating the time required for communicating with the drive ... 636
11.7.2 USS program instructions ... 637
11.7.2.1 Using the USS protocol instructions ... 637

11.7.2.2 USS_INIT instruction 638
11.7.2.3 USS_CTRL instruction 640
11.7.2.4 USS_RPM_x instruction 643
11.7.2.5 USS_WPM_x instruction 646
11.7.2.6 USS protocol example program 648
11.7.3 USS protocol execution error codes 650

11.8 SINAMICS Library 651
11.8.1 SINAMICS library features 651
11.8.2 SINA_POS instruction 652
11.8.2.1 Prerequisite of using the SINA_POS instruction 652
11.8.2.2 Input and output interface of SINA_POS instruction 655
11.8.2.3 Mode selection of SINAMICS with SINA_POS instruction 662
11.8.2.4 Relative positioning 663
11.8.2.5 Absolute positioning 666
11.8.2.6 Setup mode 671
11.8.2.7 Referencing (active referencing) 675
11.8.2.8 Referencing (set reference point) 678
11.8.2.9 Traversing blocks 681
11.8.2.10 Jog 684
11.8.2.11 Incremental jog 687
11.8.2.12 Encoder adjustment (for telegram 112 only) 691
11.8.3 SINA_SPEED instruction 692
11.8.3.1 Prerequisite of using the SINA_SPEED instruction 692
11.8.3.2 Input and output interface of SINA_SPEED instruction 694
11.8.3.3 Definition of "ConfigAxis" parameters 696
11.8.3.4 Example of SINA_SPEED instruction 696
11.8.4 SINA_PARA_S instruction 698
11.8.4.1 Prerequisite of using the SINA_PARA_S instruction 698
11.8.4.2 Input and output interface of SINA_PARA_S instruction 700
11.8.4.3 Example of the SINA_PARA_S instruction 704

11.9 User-defined library programming tips 707

11.10 How to reload libraries 714

11.11 User-defined library example 715
11.11.1 Example library for scaling analog values 715
11.11.2 Subroutine S_ITR: integer to real 716
11.11.3 Subroutine S_RTI: real to integer 718
11.11.4 Subroutine S_RTR: real to real 721
11.11.5 Creating the user-defined library 722
11.11.6 Using the library in your project 727
11.11.7 Changing the version of your library 729

12 Debugging and troubleshooting 731

12.1 Debugging your program 731
12.1.1 Bookmark functions 731
12.1.2 Cross reference table 731
12.1.3 Byte usage table 733

12.2 Displaying program status 733
12.2.1 How the program editor displays status 733
12.2.2 Configuring the STL status options 736

12.3 Using a status chart to monitor your program ... 737
12.4 Forcing specific values ... 739
12.5 Writing and forcing outputs in STOP mode ... 740
12.6 How to execute a limited number of scans ... 740
12.7 Hardware troubleshooting guide ... 742
13 PID loops and tuning ... 745
13.1 Introduction to PID self-tuning function ... 745
13.2 Pretuning ... 746
13.3 Fine tuning ... 748
13.4 PID loop definition table ... 749
13.5 Prerequisites ... 756
13.6 Auto-tune sequence ... 756
13.7 Exception conditions ... 758
13.8 Notes concerning PV out-of-range (result code 3) ... 759
13.9 PID tune control panel introduction ... 760
14 Motion control(PLCopen) ... 765
14.1 Basic concept ... 766
14.2 Motion control resources ... 768
14.3 Motion control overview ... 769
14.3.1 Control mode ... 769
14.3.1.1 Position-controlled mode (closed loop) ... 769
14.3.1.2 Speed control mode ... 771
14.3.1.3 Single-axis torque control ... 780
14.3.2 Task Mechanism ... 782
14.3.3 Process reaction ... 785
14.3.4 Compatibility ... 786
14.3.4.1 Changes to motion control in V3.1 and V3.2 ... 786
14.3.4.2 New motion control fully replaces the legacy motion wizard ... 788
14.3.4.3 Convert to STEP 7-Micro/WIN SMART V3 (PLCopen) Motion Control ... 788
14.4 Configuration ... 790
14.4.1 Setting steps ... 790
14.4.2 Hardware Network Topology View ... 791
14.4.2.1 Hardware Network Topology ... 791
14.4.2.2 Topology View ... 793
14.4.2.3 Device view ... 795
14.4.3 Editing devices and networks ... 797
14.4.3.1 Adding a device to PROFINET network ... 797
14.4.3.2 Go online ... 798
14.4.3.3 Device connections ... 799
14.4.3.4 Copying/pasting/deleting a device ... 800
14.4.3.5 Moving/swapping/inserting a device ... 801
14.4.4 Configuring the device ... 802

14.4.4.1 Configuring a controller ... 802
14.4.4.2 Configuring IRT devices ... 803
14.4.4.3 Configuring RT device ... 806
14.4.4.4 Add/remove/move RT modules ... 808
14.4.5 Topology Comparison ... 810
14.4.6 Motion control component configuration window ... 811
14.4.7 Configure the axis ... 813
14.4.7.1 Configuring the Virtual Axis ... 815
14.4.7.2 Configuring the PTO Axis ... 820
14.4.7.3 Configuring the Encoder Axis ... 834
14.4.7.4 Configure PN IRT axis ... 842
14.4.8 Configuring Axes Group ... 863
14.4.8.1 Axes Group Basic Setting ... 863
14.4.8.2 Diagnose information ... 864
14.4.9 Configuring the Cam ... 865
14.4.9.1 Configuring the cam table ... 865
14.4.9.2 Set the parameters ... 868
14.4.9.3 View the curve ... 869
14.4.9.4 Import/Export cam table ... 870

14.5 Commissioning (for PTO axis and PN IRT axis only) ... 872
14.5.1 Control Panel ... 872
14.5.2 Parameter tuning ... 875

14.6 General mechanism ... 879
14.6.1 Instruction restart ... 880
14.6.2 State Machine ... 884
14.6.2.1 Single axis state machine ... 884
14.6.2.2 Axes group state machine ... 886
14.6.3 Buffered mode ... 887
14.6.3.1 Single-axis command buffered mode ... 887
14.6.3.2 Axes group command buffered mode ... 891
14.6.4 Overview of variable ... 895
14.6.5 Variable overview table ... 897
14.6.6 Achieve multi-axis control with arrays ... 905

14.7 Motion Control Instructions ... 906
14.7.1 Basic Instruction ... 906
14.7.1.1 MC_Power ... 906
14.7.1.2 MC_TouchProbe ... 908
14.7.1.3 MC_AbortTrigger ... 912
14.7.1.4 MC_Reset ... 913
14.7.1.5 MC_Home ... 915
14.7.1.6 MC_Stop ... 920
14.7.1.7 MC_Halt ... 922
14.7.1.8 MC_MoveAbsolute ... 928
14.7.1.9 MC_MoveRelative ... 933
14.7.1.10 MC_MoveVelocity ... 940
14.7.1.11 MC_ImmediateStop ... 944
14.7.1.12 MC_MoveJog ... 947
14.7.1.13 MC_TorqueControl ... 952
14.7.1.14 MC_TorqueRange ... 960
14.7.1.15 MC_TorqueLimiting ... 966
14.7.2 Cam and Gear Instruction ... 970

14.7.2.1 MC_CamIn ... 970
14.7.2.2 MC_CamOut ... 985
14.7.2.3 MC_GearIn ... 989
14.7.2.4 MC_GearOut ... 993
14.7.2.5 MC_PhasingRelative ... 997
14.7.2.6 MC_GenerateCamTable ... 1000
14.7.3 Axes Group instructions ... 1005
14.7.3.1 MC_GroupEnable ... 1005
14.7.3.2 MC_GroupDisable ... 1006
14.7.3.3 MC_GroupReset ... 1007
14.7.3.4 MC_GroupStop ... 1008
14.7.3.5 MC_MoveLinearAbsolute ... 1009
14.7.3.6 MC_MoveLinearRelative ... 1011

14.8 Troubleshooting and optimization for insufficient motion control resources ... 1012

14.9 Error codes ... 1015
14.9.1 Error code for single axis motion ... 1015
14.9.2 Error codes for electronic cam and gear ... 1020
14.9.3 Error code for axes group ... 1021

14.10 Example programs ... 1022
14.10.1 Motion control example ... 1022
14.10.2 Single Axis example ... 1028
14.10.3 Axes group example ... 1031
14.10.4 Cam example ... 1033

15 Open loop motion control(Legacy) ... 1039

15.1 Using the PWM output ... 1040
15.1.1 Using the PWM (pulse width modulation) output ... 1040
15.1.2 Configuring the PWM output ... 1041
15.1.3 PWMx_RUN subroutine ... 1042

15.2 Using motion control ... 1042
15.2.1 Maximum and start/stop speeds ... 1043
15.2.2 Entering the acceleration and deceleration times ... 1044
15.2.3 Configuring the motion profiles ... 1044

15.3 Features of motion control ... 1046

15.4 Using Axis of motion ... 1048
15.4.1 Programming an Axis of Motion ... 1048
15.4.2 Configuring an Axis of Motion ... 1049
15.4.3 Subroutines created by the Motion wizard for the Axis of Motion ... 1063
15.4.3.1 Guidelines for using the Motion subroutines ... 1064
15.4.3.2 AXISx_CTRL subroutine ... 1064
15.4.3.3 AXISx_MAN subroutine ... 1066
15.4.3.4 AXISx_GOTO subroutine ... 1067
15.4.3.5 AXISx_RUN subroutine ... 1069
15.4.3.6 AXISx_RSEEK subroutine ... 1070
15.4.3.7 AXISx_LDOFF subroutine ... 1071
15.4.3.8 AXISx_LDPOS subroutine ... 1072
15.4.3.9 AXISx_SRATE subroutine ... 1073
15.4.3.10 AXISx_DIS subroutine ... 1074
15.4.3.11 AXISx_CFG subroutine ... 1074

15.4.3.12 AXISx_CACHE subroutine ... 1075
15.4.3.13 AXISx_RDPOS subroutine ... 1076
15.4.3.14 AXISx_ABSPOS subroutine ... 1077
15.4.4 Using the AXISx_ABSPOS subroutine to read the absolute position from a SINAMICS servo drive ... 1078
15.4.4.1 AXISx_ABSPOS and AXISx_LDPOS subroutines usage examples ... 1079
15.4.4.2 Interconnections ... 1080
15.4.4.3 Commissioning ... 1081
15.4.4.4 Important facts to know ... 1084
15.4.5 Axis of Motion example programs ... 1085
15.4.5.1 Axis of Motion simple relative move (cut-to-length application) example ... 1085
15.4.5.2 Axis of Motion AXISx_CTRL, AXISx_RUN, AXISx_SEEK, and AXISx_MAN example ... 1087
15.4.6 Monitoring the Axis of Motion ... 1092
15.4.6.1 Displaying and controlling the operation of the Axis of Motion ... 1092
15.4.6.2 Displaying and modifying the configuration of the Axis of Motion ... 1097
15.4.6.3 Displaying the profile configuration for the Axis of Motion ... 1097
15.4.6.4 Error codes for the Axis of Motion (WORD at SMW620, SMW670, SMW720 or SMW770) ... 1098
15.4.6.5 Error codes for the Axis of Motion instruction (seven LS bits of SMB634, SMB684, SMB734 or SMB784)) ... 1099
15.4.7 Advanced topics ... 1101
15.4.7.1 Understanding the configuration/profile table for the Axis of Motion ... 1101
15.4.7.2 Special memory (SM) locations for the Axis of Motion ... 1108
15.4.8 Understanding the RP Seek modes of the Axis of Motion ... 1110
15.4.8.1 Selecting the work zone location to eliminate backlash ... 1114

15.5 Using Axes group of Motion ... 1115
15.5.1 Programming an Axes group of Motion ... 1115
15.5.2 Configuring an Axes group of Motion ... 1115
15.5.3 Subroutines created by the Motion wizard for the Axes group of Motion ... 1122
15.5.3.1 Guidelines for using the Motion subroutines ... 1123
15.5.3.2 GRPx_2/3D_MOVELINEAR subroutine ... 1123
15.5.3.3 GRPx_MOVEPATH subroutine ... 1125
15.5.3.4 GRPx_RESET subroutine ... 1126
15.5.3.5 GRPx_STOP subroutine ... 1128
15.5.4 Axes group of Motion example programs ... 1129
15.5.4.1 Example of using GPRx_STOP subroutine to stop axes group movement ... 1136

15.6 Stopping the Axes group movement ... 1137

15.7 Error codes for the Axes group of Motion (WORD at SMW820 or SMW890) ... 1138

15.8 Error codes for the Axes group of Motion instruction (seven LS bits of SMB823 or SMB893) ... 1138

15.9 Interactions between Axis and Axis group ... 1139

A Technical specifications ... 1141

A.1 General technical specifications ... 1141

A.2 S7-200 SMART CPU ... 1146
A.2.1 CPU ST32 ... 1146
A.2.1.1 General specifications and features ... 1146
A.2.1.2 Digital inputs and outputs ... 1150
A.2.1.3 Wiring diagram ... 1152

A.3 Digital inputs and outputs expansion modules (EMs) ... 1153
A.3.1 EM DE16 digital input specifications ... 1153

A.3.2 EM QT16 digital output specifications ... 1155

A.4 Analog inputs and outputs expansion modules (EMs) ... 1156
A.4.1 EM AE04 and EM AE08 analog input specifications ... 1156
A.4.2 EM AQ04 Analog output module specifications ... 1159
A.4.3 EM AM06 analog input/output module specifications ... 1162
A.4.4 Step response of the analog inputs ... 1165
A.4.5 Sample time and update times for the analog inputs ... 1165
A.4.6 Measurement ranges of the analog inputs for voltage and current (SB and SM) ... 1165
A.4.7 Measurement ranges of the analog outputs for voltage and current (SB and EM) ... 1166

A.5 Thermocouple expansion modules (EMs) ... 1168
A.5.1 EM AT08 thermocouple specifications ... 1168

A.6 RTD expansion modules (EMs) ... 1172
A.6.1 EM AR04 RTD specifications ... 1172

A.7 RS485/RS232 signal board ... 1176
A.7.1 SB RS485/RS232 specifications ... 1176

B Calculating a power budget ... 1179

B.1 Power budget ... 1179

B.2 Calculating a sample power requirement ... 1180

B.3 Calculating your power requirement ... 1181

C Error code ... 1183

C.1 Timestamp mismatch ... 1183

C.2 PLC non-fatal error codes ... 1184

C.3 PLC non-fatal error SM flags ... 1186

C.4 PLC fatal error codes ... 1187

C.5 PROFINET Alarm ... 1189

D Special memory (SM) and system symbol names ... 1191

D.1 SM (Special Memory) overview ... 1191

D.2 SMB0: System status ... 1193

D.3 SMB1: Instruction execution status ... 1194

D.4 SMB2: Freeport receive character ... 1195

D.5 SMB3: Freeport character error ... 1195

D.6 SMB4, SM20.0: Interrupt queue overflow, run-time program error, interrupts enabled, freeport transmitter idle, and value forced ... 1196

D.7 SMB5: I/O error status ... 1196

D.8 SMB6-SMB7: CPU ID, error status, and digital I/O points ... 1197

D.9 SMB8-SMB19, SMB131-SMB134, SMB2030-SMB2045: I/O module ID and errors ... 1197

D.10 SMW22-SMW26: Scan times ... 1199

D.11 SMB28-SMB29, SMB32-SMB33: Signal board ID and errors ... 1199

D.12 SMB30: (Port 0), SMB130: (Port1) and SMB31: (Port 2) ... 1201

D.13 SMB34-SMB35: Time intervals for timed interrupts 1202
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 1202
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 1204
D.16 SMB86-SMB94, SMB186-SMB194 and SMB405-SMB413: Receive message control..... 1211
D.17 SMW98: Expansion I/O bus communication errors 1215
D.18 SMW100-SMW120, SMW2014-SMW2028 system alarms 1215
D.19 SMB195-SMB210: HSC Type and Frequency measurement sampling period 1217
D.20 SMB220-SMB235: PID control Error code 1217
D.21 SMB380-SMB404: HSC User-defined pin mapping enable 1218
D.22 SMB480-SMB515: Data log status 1218
D.23 SMB600-SMB799: Axis (0, 1, 2 and 3) open loop motion control (for Motion(V2.x)) 1219
D.24 SMB600-SMB649: Axis 0 open loop motion control 1221
D.25 SMB650-SMB699: Axis 1 open loop motion control 1221
D.26 SMB700-SMB749: Axis 2 open loop motion control 1221
D.27 SMB750-SMB799: Axis 3 open loop motion control 1221
D.28 SMB800-SMB858: Axes group 0 open loop motion control..... 1221
D.29 SMB870-SMB928: Axes group 1 open loop motion control..... 1222
D.30 SMB929-SMB936: Trace control and trace status 1223
D.31 SMB1000-SMB1049: CPU hardware/firmware ID 1224
D.32 SMB950-SMB998, SMB1050-SMB1099: SB (signal board) hardware/firmware ID 1224
D.33 SMB1100-SMB1399, SMB250-SMB349, SMB2050-SMB2449: EM (expansion module) hardware/firmware ID..... 1225
D.34 SMB1400-SMB1799, SMB2450-2999: EM (expansion module) module-specific data 1231
D.35 SMB1800-SMB1939: PROFINET device status..... 1232
D.36 SMB1940-SMB1946: Web server status 1233
E Reference 1235
E.1 Often-used special memory bits 1235
E.2 Interrupt events in priority order..... 1236
E.3 High-speed counter summary 1238
E.4 STL instructions 1238
E.5 CPU memory ranges and features..... 1245

F Ordering information 1249
F.1 CPU module 1249
F.2 Expansion modules (EMs) and signal boards (SBs) 1249
F.3 Programming software 1249
F.4 Spare parts and other hardware 1249
Index 1251

Product overview

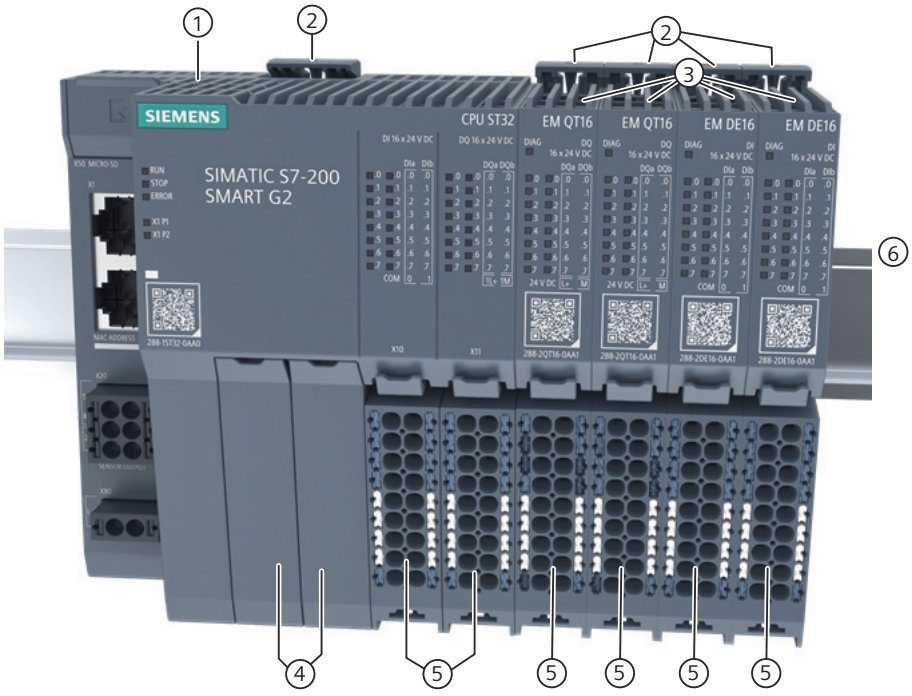
1

The S7-200 SMART G2 series is the second generation of the S7-200 SMART, featuring a significantly narrower footprint compared to the first generation. The CPU and expansion modules offer high flexibility, supporting a wide range of automation application scenarios.

The CPU monitors inputs and changes outputs as controlled by the user program, which can include Boolean logic, counting, timing, complex math operations, and communications with other intelligent devices. The compact design, flexible configuration, and powerful instruction set combine to make the S7-200 SMART G2 a perfect solution for controlling a wide variety of applications.

S7-200 SMART G2 system

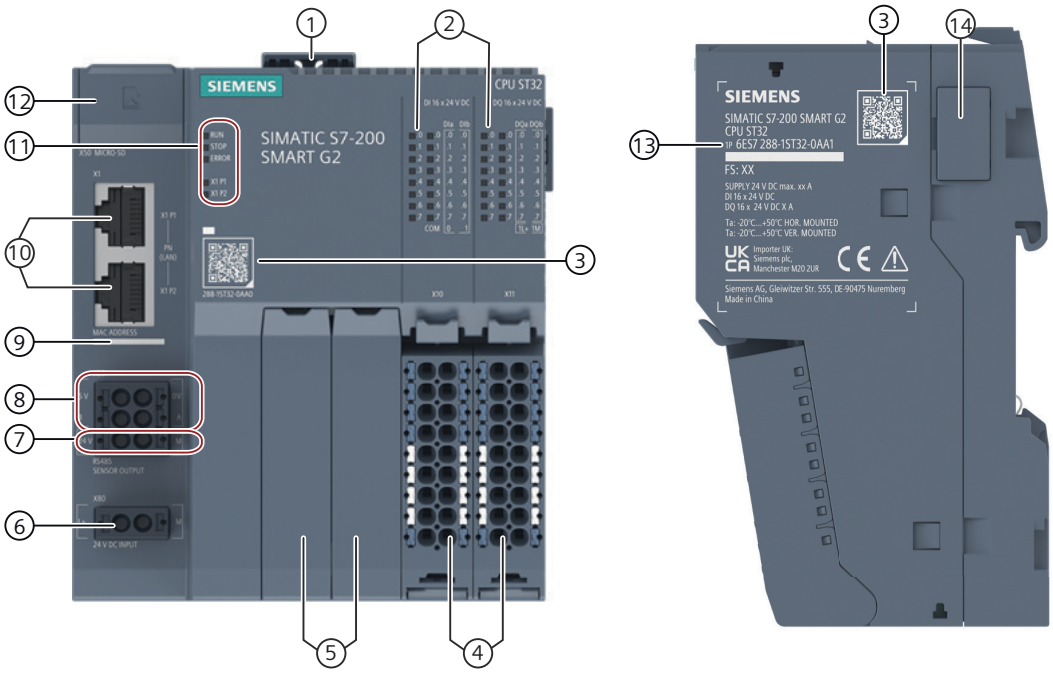
The S7-200 SMART G2 modules can be installed on a single mounting rail, with support for up to 19 modules (CPU, 2 signal boards, and 16 I/O modules) on the rail.



1. CPU module
2. Mounting rail release button
3. I/O modules
4. Optional signal board
5. Terminal connectors
6. Mounting rail

1.1 S7-200 SMART G2 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	Mounting rail release button	⑧	RS485 port
②	Status LEDs for the on-board I/O	⑨	MAC address
③	ID link	⑩	Dual-port Ethernet interface
④	Terminal connectors	⑪	Status LED of the CPU
⑤	Optional signal board	⑫	Memory card slot (below the cover)
⑥	Power input connector	⑬	Article number
⑦	24 V DC Sensor Power Out	⑭	I/O bus connector cover

The CPU provides different models with a diversity of features and capabilities that help you create effective solutions for your varied applications. For detailed information about a specific CPU, see the technical specifications (Page 1146).

The relay output type CPU will be released at a later date.

Refer to the technical specifications (Page 1146) for the power requirements of the CPU and the expansion modules. Use the worksheets in Appendix B, Calculating a power budget (Page 1179) to calculate your power budget (Page 1179).

S7-200 SMART G2 CPU

	ST32
Standard, expandable	✓
Relay outputs	
Transistor output (DC)	✓
I/O points (built-in)	32

Standard expandable CPUs

Features		**CPU ST32**
Dimensions: W x H x D (mm)		100 x 118 x 74.5
User memory	Program	120 KB[1]
	User data	100 KB
	Retentivity	40 KB max.[2]
	Memory for unbinded global variables and FB instances (DB2)	1 MB
On-board digital I/O	• Inputs • Outputs	• 16 DI • 16 DQ
Expansion module		max. 16
Signal board		2
High-speed counters (8 total)	Single phase / Dual phase	8 at 200 kHz
	A/B phase	8 at 200 kHz 4x frequency support up to 800 kHz
Pulse outputs [3]		5 at 200 kHz
PID loops		16
Real-time clock with 7-day back-up		✓

[1] The user program size for CPU ST32 (V3.1) is 100 KB.

[2] 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.

[3] 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.

1.2 What's new?

1.2.1 What's new in S7-200 SMART V3.2?

Only the following CPU models with firmware V3.2 support the new features described in this chapter:

Table 1-1 CPU models affected by firmware V3.2

CPU model	Article number
CPU ST32, DC/DC/DC	6ES7288-1ST32-0AA0

Note

CPU models with firmware V3.2 cannot be downgraded to V3.1 or any previous versions.

Motion control (PLCOpen)

The new motion control brings bus-level real-time synchronization and higher-precision multi-axis interpolation capabilities, while the addition of torque control-related instructions improves torque management capabilities.

- New IRT motion control
  - New IRT axis
  - Supports for a minimum PROFINET bus motion control cycle of 1 ms
  - Supports for bus configuration, commissioning, and diagnostics
  - Supports for single-axis, cam, gear motion
  - Supports for various control modes: Position mode, velocity mode, torque mode
- New motion instructions:
  - MC_TorqueControl (Page 952)
  - MC_TorqueRange (Page 960)
  - MC_TorqueLimiting (Page 966)

Hardware Network Topology

New hardware network topology is provided for configuring the PROFINET network

- Topology View (Page 793): Configuring a PROFINET network topology
- Device View (Page 795): Allows you to manage and configure RT devices
- Topology Comparison: Able to compare the differences between the current online network and the configured network.

When you use STEP 7-Micro/WIN SMART (3. 2) to open a project created in a previous version that contains PROFINET RT configurations, these PROFINET wizard configurations will be automatically migrated and integrated into the Hardware Network Topology.

Trace

Supports the IRT sampling cycle.

Programming

- Import and export of UDTs, FBs, and variable tables.

New instructions

- R_TRIG (Page 268)
- T_TRIG (Page 268)
- CTU_DINT (Page 340)
- CTD_DINT (Page 340)
- CTUD_DINT (Page 340)
- IEC_TON (Page 464)
- IEC_TOF (Page 464)
- IEC_TONR (Page 464)

Package project

An "Package project" button has been added to the File menu, allowing you to package the project files along with the following content into a .zip archive:

- The GSDML file used by the project (included by default)
- Trace (optional)

PLC information

The scan rate for MC_USER_TASK has been added to the Scan Rate (Page 179) page, along with monitoring for the maximum/minimum and timeout of motion tasks.

SmartBot

Note

SmartBot is only available in mainland China.

STEP 7- Micro/WIN SMART V3 (Version 3.2) introduces a new AI assistant "SmartBot" for FAQ.

Deletion Wizard

- PROFINETSTEP 7-Micro/WIN SMART (Version 3.2) no longer provides the legacy RPOFINET wizard.
- **Text Display**: STEP 7-Micro/WIN SMART (Version 3.2) no longer provides **Text Display** Wizard for configuring the TD400.
  When you use STEP 7-Micro/WIN SMART (Version 3.2) to open a project created in a lower version that contains a **Text Display** configuration, the program remains usable. However, it is not possible to add new configurations or modify existing ones.

1.2.2 What's new in S7-200 SMART V3.1?

Full support for S7-200 SMART G2 modules

The ST32 supports connection of up to 16 expansion modules.

Module type	Module description	Article number
CPU	CPU ST32 (DC/DC/DC)	6ES7288-1ST32-0AA0
Signal board	SB RS485/RS232 (SB CM01)	6ES7288-5CM01-0AA1
Expansion module	EM Digital 16 x Inputs (EM DE16)	6ES7288-2DE16-0AA1
	EM Digital 16 x Outputs Transistor (EM QT16)	6ES7288-2QT16-0AA1
	EM Analog 4 x Inputs (EM AE04)	6ES7288-3AE04-0AA1
	EM Analog 4 x Outputs (EM AQ04)	6ES7288-3AQ04-0AA1
	EM Analog 8 x Inputs (EM AE08)	6ES7288-3AE08-0AA1
	EM Analog 4 x Inputs / Analog 2 x Outputs (EM AM06)	6ES7288-3AM06-0AA1
	EM RTD 4 x 16 bit (EM AR04)	6ES7288-3AR04-0AA1
	EM TC 8 x 16 bit (EM AT08)	6ES7288-3AT08-0AA1

Migrate

Migrate (Page 145): STEP 7-Micro/WIN SMART V3 (Version 3.1) allows you to migrate your STEP 7-Micro/WIN SMART (Version 2.8) or STEP 7-Micro/WIN SMART V3 (Version 3.0) projects and configurations to the S7-200 SMART G2 series.

During the project migration process, the CPU used in the original project will be migrated to ST32 by default; the signal board (SB) will be migrated to the signal board of the same name in the S7-200 SMART G2 series. If there is no signal board of the same name, its configuration will be cleared; The expansion module (EM) will be automatically replaced with S7-200 SMART G2 expansion module (EM) according to the matching rules.

Trace

The trace function (Page 184) is used to monitor and record variable values in a program in real time.

Once a trace is activated, it will start recording variable values with timestamps. These recorded data can be continuously monitored by STEP 7-Micro/WIN SMART V3 (Version 3.1) to provide a basis for your debugging. Once a trace is activated, the trace function will be executed independently of the main program.

Trace data is temporarily stored in RAM. To retain the records, you must upload them in STEP 7-Micro/WIN SMART V3 (Version 3.1) and then export and save them.

Programming

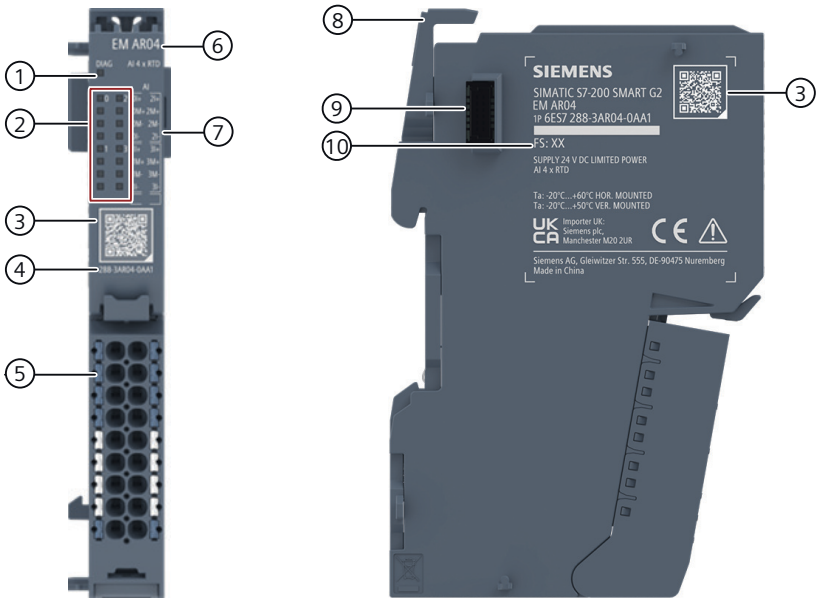
- User-defined libraries (Page 707) support user-defined data types (UDT) and function blocks (FB).
- User-defined libraries (Page 707) support the use of DB2.
- When creating a user-defined library (Page 722), you are allowed to define a namespace and configure library information.
- Updating parameters at call locations (including Subroutines SBR and Function Blocks FB) are supported when a block's input/output parameter declarations are modified.

1.3 S7-200 SMART G2 expansion module

To better meet your application requirements, the S7-200 SMART G2 family includes a wide variety of expansion modules and signal boards. You can use these expansion modules with the standard CPU model (ST32) to add additional functionality to the CPU.

Expansion modules (EMs)

The ST32 CPU supports up to 16 expansion modules.



①	LED indicators for diagnostics	⑥	Module type and name
②	Channel status LED	⑦	Terminal port definitions
③	ID link	⑧	Mounting rail release button
④	Article number	⑨	IO bus connector
⑤	Terminal connectors	⑩	FS number

Signal board (SB)

Signal boards (such as the CM01) can be added to enhance the communication capability of the CPU.

- The ST32 CPU supports two plug-in expansion boards.

In the following diagram, the plug-in expansion board is highlighted, and its main features are labeled:



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 1141).

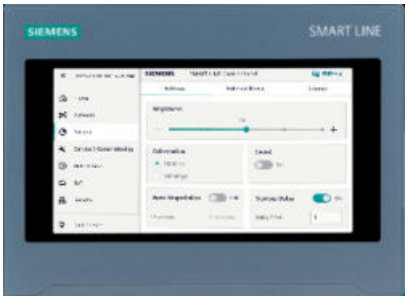
Table 1-2 Expansion modules and signal boards

Type	Input only	Output only	Combination In/Out	Others
Digital expansion module	• 16 x DC In	• 16 x DC Out		
Analog expansion modules	• 4 x Analog In • 8 x Analog In • 4 x RTD In • 8 x thermocouple inputs	• 4 x Analog Out	• 4 x Analog In / 2 x Analog Out	
Signal board				• RS485/ RS232

1.4 HMI devices for S7-200 SMART G2

The S7-200 SMART G2 supports Comfort HMIs, SMART HMIs, Basic HMIs and Micro HMIs. The SMART LINE Touch Panel is shown below.

Table 1-3 HMI device

	**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 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:

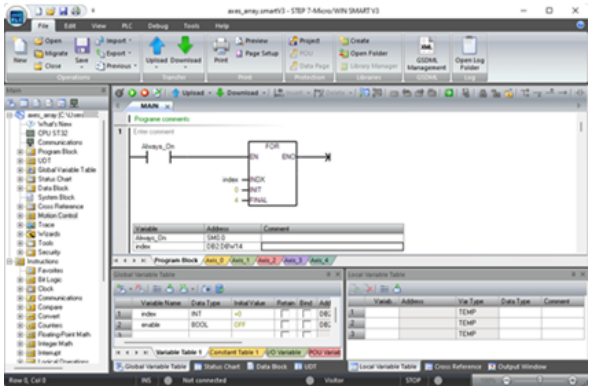
- Ethernet:
  - Exchange of data from the programming device to the CPU
  - Exchange of data between HMIs and the CPU
  - S7 peer-to-peer communication with other S7-200 SMART CPUs
  - Open User Communication (OUC) with other Ethernet-capable devices
  - PROFINET communication with PROFINET devices
- RS485:
  - Provides a STEP 7-Micro/WIN SMART V3 connection for programming when using a USB-PPI cable

    Note

    A direct connection to the USB-PPI cable is not supported by the S7-200 SMART G2 CPU. To connect an S7-200 SMART G2 CPU with this cable, you will need to make your own RS485 connection wire.

  - 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-Micro/WIN SMART 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!

Security information

2

2.1 Safety information

2.1.1 Warnings in this document

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

2.1.2 Safety-related symbols

2.1.2.1 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
	Note that connected mains lines must be designed according to the expected minimum and maximum ambient temperature.
	Note that the device must be constructed and connected in accordance with EMC regulations.
	Note that a 230 V device can be exposed to electrical voltages which can be dangerous. ANSI Z535.2
	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

2.1.2.2 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
	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"
	Be aware that the device is only approved for the industrial field and only for indoor use.
	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.
	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.

2.1.3 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 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.

2.1.4 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.

2.1.5 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.

2.1.6 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

2.1.7 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.

2.1.8 Safe working

2.1.8.1 Working on electrical parts

- Only work on electrical parts if you are a qualified specialist (see sectionKnowledge and personnel qualification (Page 38)).
- 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.

2.1.9 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.

2.1.10 Residual risks

2.1.10.1 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.

2.1.10.2 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 38)).
- Adhere to the protective measures for safe working on electrical parts (see section Working on electrical parts (Page 39))
- 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 38)).
- Adhere to the protective measures for safe working on electrical parts (see section Working on electrical parts (Page 39))

2.1.10.3 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

2.1.10.4 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

2.1.10.5 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.

2.1.10.6 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.

2.1.10.7 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.

2.1.10.8 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).
- 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.

2.1.11 Material damage

2.1.11.1 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 section.

2.1.11.2 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

2.2 Industrial cybersecurity

2.2.1 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://www.siemens.com/cert.

2.2.2 Safety instructions

Disclaimer

Please note that the following list of recommended risk-minimizing security measures is not intended to be exhaustive. Therefore, please consult your security expert for final assessment and configuration. Further, as already mentioned in Industry security (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.

Note on 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 allowlist tool on the PC. Then use this 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.

2.2.3 Security overview

Overview

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

Security features	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 55)
Communications protocols and port (Page 49)	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 PLC security (Page 55)
POU access protection	The POU protection method can help you protect the POU from being leaked. For more information, refer to POU / data page password protection (Page 65)

Security features	Description
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 64)
Certificate	The certificate management wizard is used to configure the HTTPS certificates for the Web server. For more information, refer to Downloading and installing the certificate of CA (Page 239)

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 the device in a cabinet.

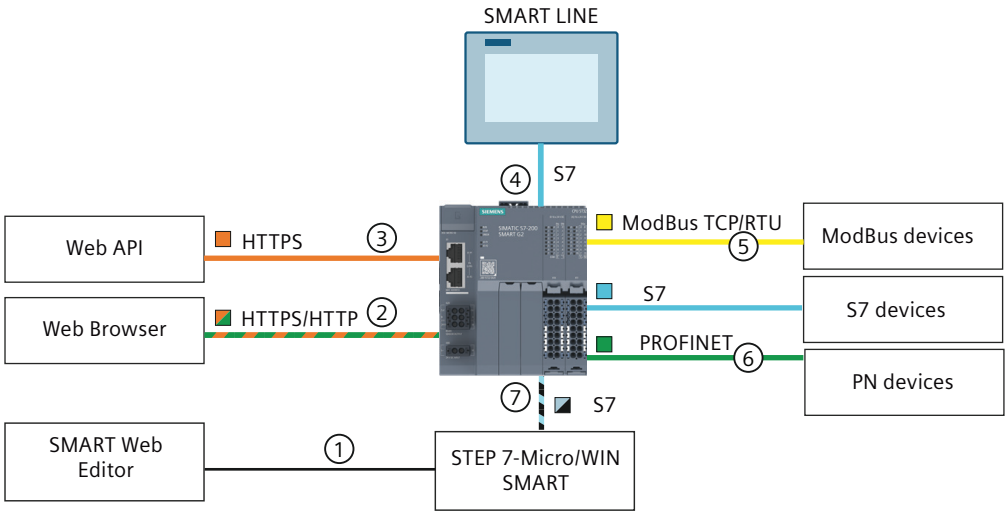
See also

Configuring communication (Page 52)

2.2.4 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.



Connection	Supported APP/ Device	Port	Protocol	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	TCP 102	S7	
	S7 Device			
⑤	Modbus Device	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.
⑦	STEP 7-Micro/WIN SMART V3	TCP 102	S7	

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.

2.2.5 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 2-1 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. HTTPS is used for communication with the CPU-internal web server.	Default: activated.

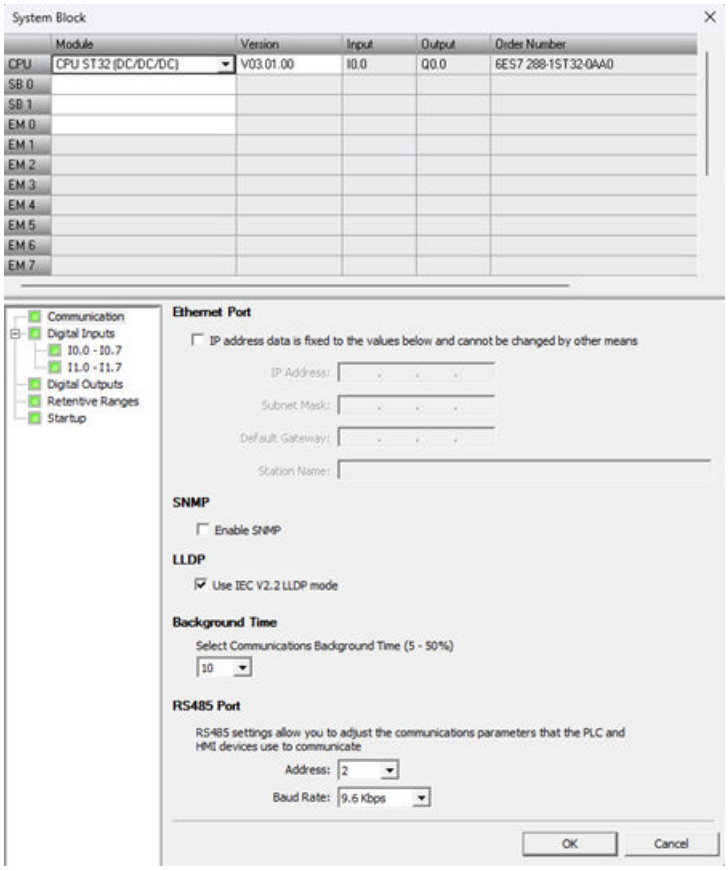
Protocol / Role	Port number	(2) Link layer (4) Transport layer	Description / function	Default setting / notes
HTTPS Server	443	(4) TCP	Hypertext Transfer Protocol Secure. HTTPS is used for communication with the CPU-internal Web server via Secure Socket Layer (SSL).	Default: Deactivated. Can be enabled 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 (Complies with 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 63). 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.

2.2.6 Configuring communication

Click the Communication node of the system block (Page 205) dialog to configure the Ethernet port, SNMP, background time, and RS485 port.



Ethernet ports

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 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.
- 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.

Note

A direct connection to the USB-PPI cable is not supported by the S7-200 SMART G2 CPU. To connect an S7-200 SMART G2 CPU with this cable, you will need to make your own RS485 connection wire.

2.2.7 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 55)".

2.2.8 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.

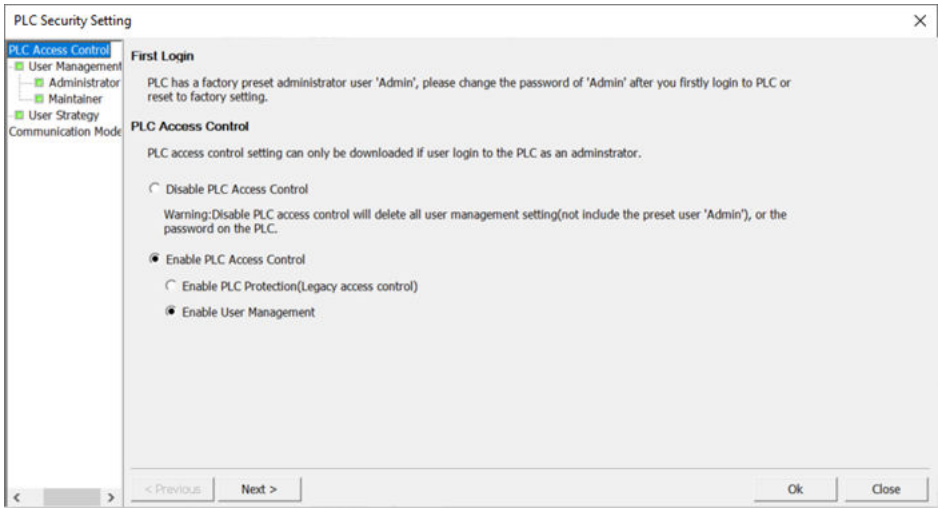


- Open the Security folder in the project tree, select the "PLC Security Setting" node and press Enter; or double-click the "PLC Security Setting" node.



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

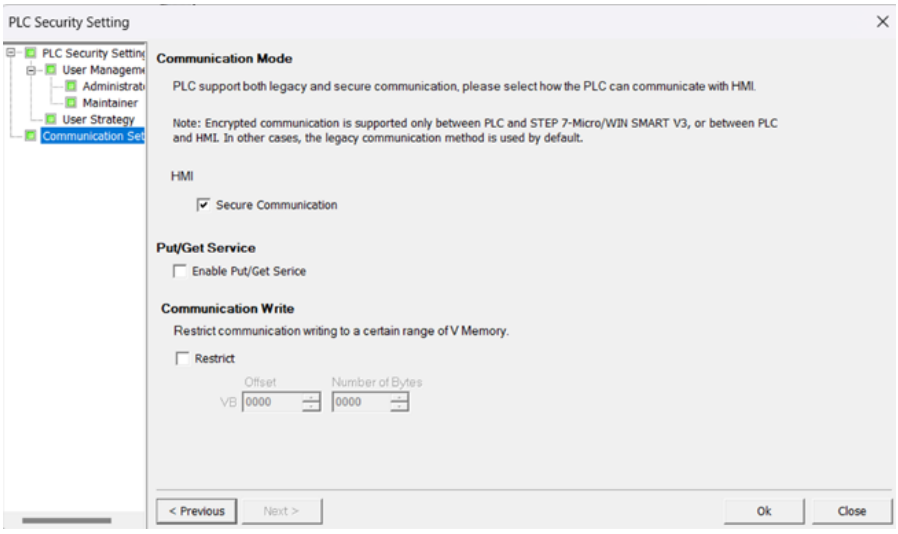
The default and recommended condition for the S7-200 SMART is "Enable User Management".

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.



2.2.8.1 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 2-2 S7-200 SMART CPU password-protected privilege levels

Operating instructions	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	Not permitted
Download the project	Permitted	Restricted	Restricted	Restricted
reset to factory defaults	Permitted	Restricted	Restricted	Restricted
Delete of 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	Restricted	Restricted	Not permitted
Project compare	Permitted	Permitted	Restricted	Not permitted
Download trace configuration	Permitted	Restricted	Restricted	Restricted
Delete trace configuration	Permitted	Restricted	Restricted	Restricted
Upload trace	Permitted	Permitted	Restricted	Restricted
Activate/deactivate trace	Permitted	Restricted	Restricted	Restricted
Trace status polling	Permitted	Permitted	Permitted	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.)

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:

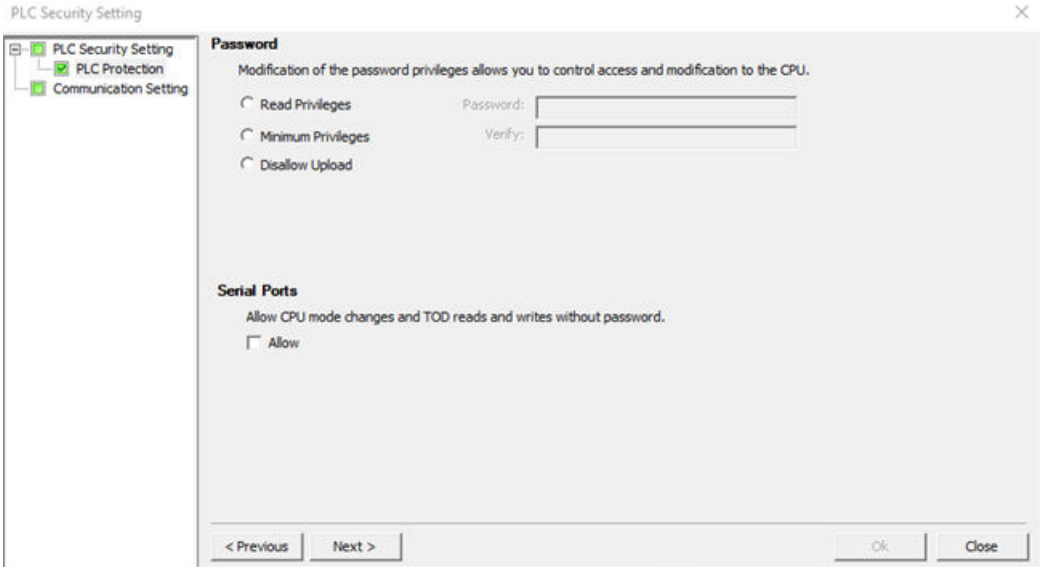
1. Select "Disable PLC Access Control" in the PLC Security Setting panel.
2. Click **Ok**.

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 63).

2.2.8.2 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.

Operating instructions	Admin	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 the 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

Operating instructions	Admin	Maintainer	Visitor
Change current user password	Permitted	Permitted	Restricted
Flash LED	Permitted	Permitted	Permitted
Download trace configuration	Permitted	Restricted	Restricted
Delete trace configuration	Permitted	Restricted	Restricted
Upload trace	Permitted	Permitted	Restricted
Activate/deactivate trace	Permitted	Permitted	Restricted
Trace status polling	Permitted	Permitted	Restricted

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.

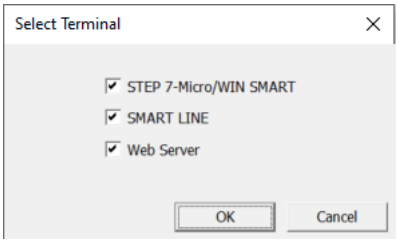
Note

- 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.



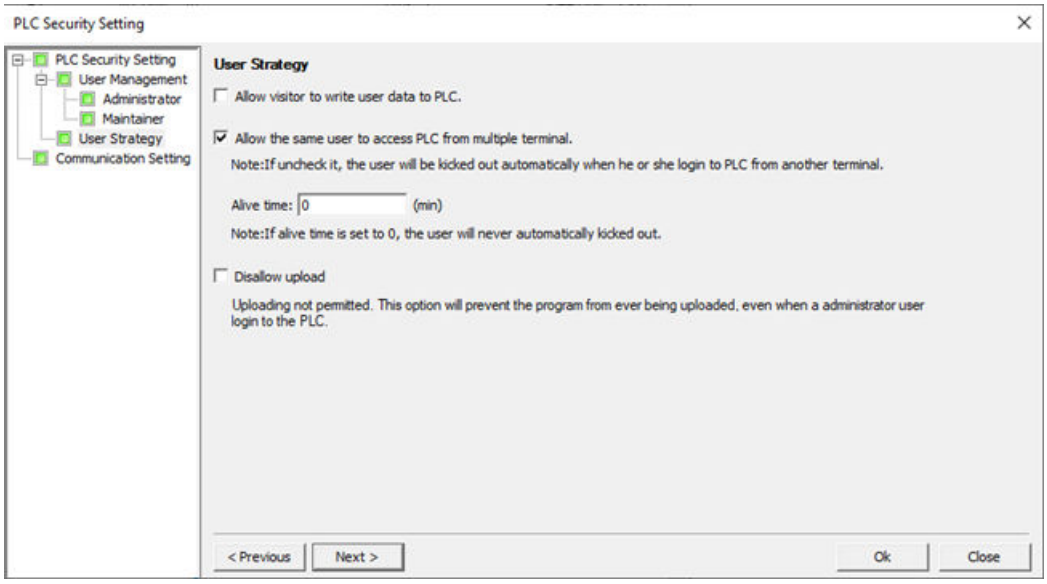
- **Comments:** you can add comments for the user role. Not required.

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.
  - "Active time" is used to set the duration for which a logged-in user can access the PLC. When the time expires, the user logged in on the terminal will be logged 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.

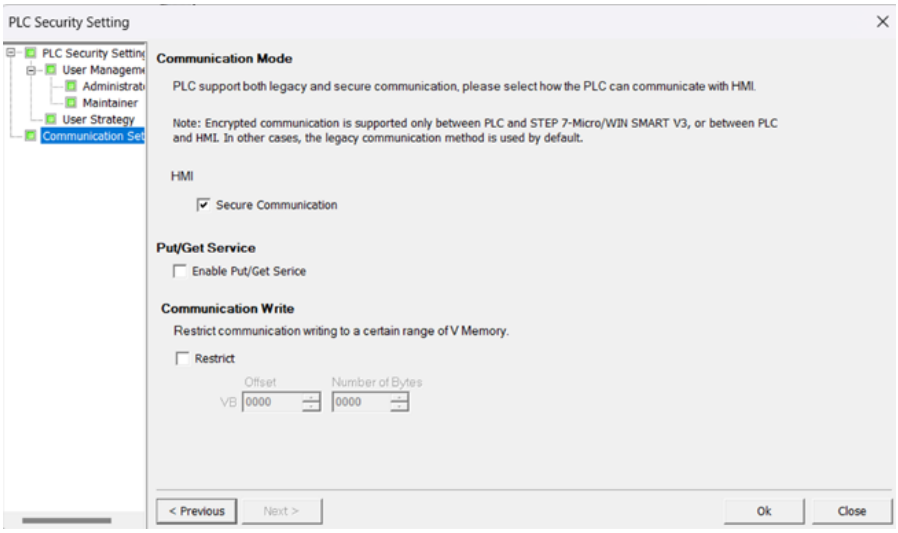
Communication setting

Setting the communication setting (Page 63)

2.2.8.3 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.

Note

Enabling the Put/Get server functionality introduces potential network security risks. Please ensure the security of your local area network (LAN) environment.

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, AQ and DB2.

2.2.9 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.

Note

Unencrypted project files are only accessible to users with customer administrator privileges.

2.2.9.1 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.



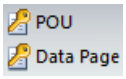
2. Select the "Password-protect this project" checkbox.
3. Enter and verify a password.

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

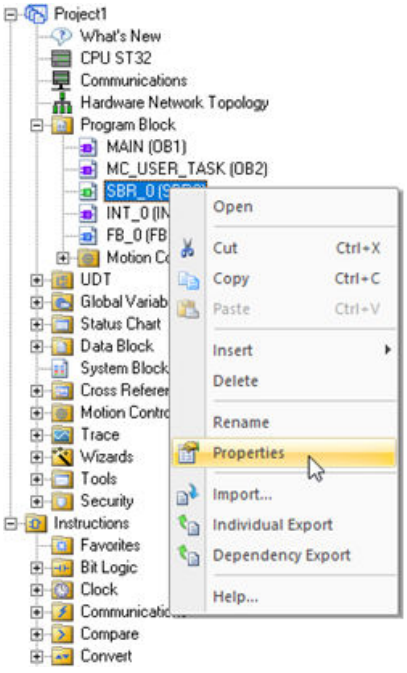
2.2.9.2 POU / data page password protection

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 (Page 145) or data block in the project tree (Page 149) and select **Properties** from the context menu.

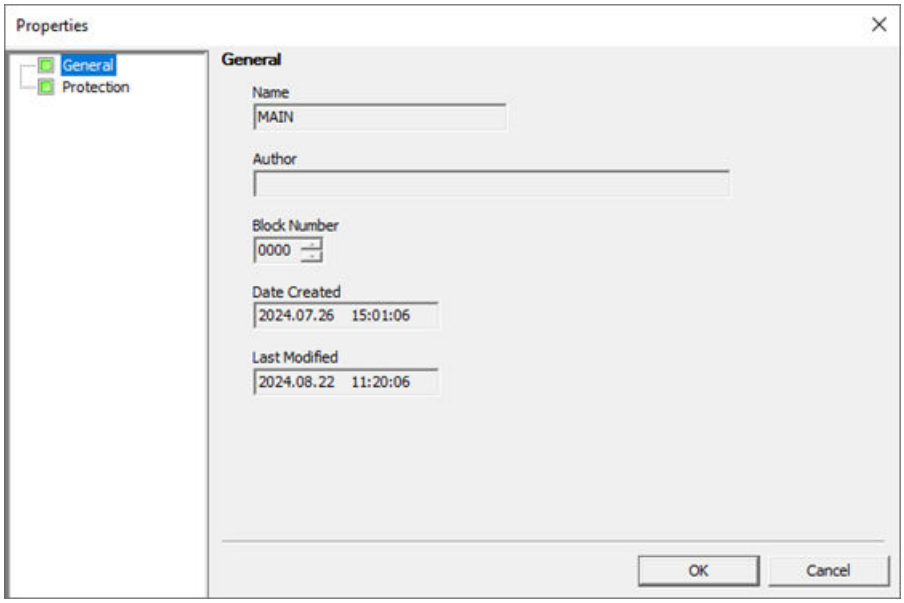


- Right-click a POU (Page 145) or data block page tab in the editor window and click the "Properties" button.



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

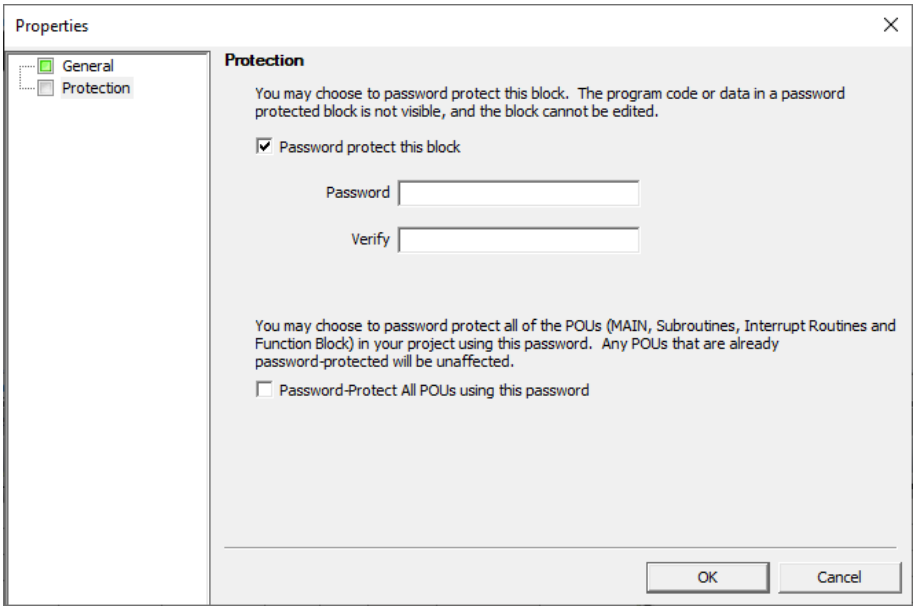
You cannot use the default name (the POU address assigned by STEP 7-Micro/WIN SMART V3, such as SBR_1 for a subroutine , a INT_1 for an interrupt routine or FB_1 for a function block) as the global variable name for another POU because that would constitute a duplicate assignment. If you violate the 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 10-32 characters
- Must include numbers, uppercase and lowercase letters, and special characters !@#$ %^&*()-+~_={}[]|\: ;,.<>
- Not words that can be found in a dictionary,
- cannot be a name or identifier 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:

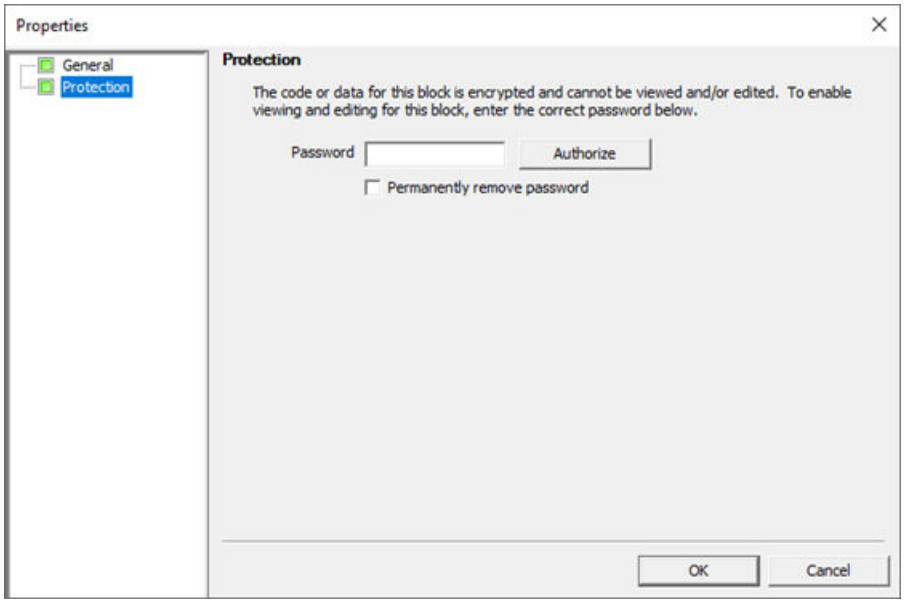


Unlocking a POU

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. Input 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, but 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

Project password protection (Page 64)

Set Project Password (Page 64)

Application user reference (Page 149)

Getting started (Page 64)

See also

Configuring PLC security (Page 55)

Getting started (Page 71)

2.2.10 Secure decommissioning

2.2.10.1 Decomissioning introduction

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.

2.2.10.2 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 steps in the following links to clear PLC. The function deletes all information that was saved internally on the module.

- Clearing PLC memory (Page 233)
- Creating a reset-to-factory-defaults memory card (Page 235)

2.2.10.3 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

3

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 RS485 communication cable, a CPU, and a programming device running the STEP 7-Micro/WIN SMART V3 programming software.

3.1 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 communication cable between your programming device and the CPU.

Power supply unit

Only use a SELV/PELV type power supply unit with safe electrical isolation and an extra-low voltage (≤ 28.8 VDC).

Connecting Power to the CPU (X80)

⚠ WARNING

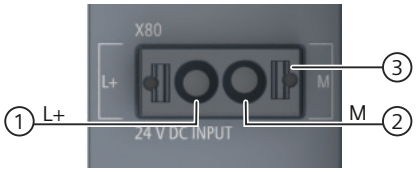
Ensure power is off prior to installing, wiring or removing devices

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

Attempts to install or wire 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.

The following diagram lists the signal names for a 24 V DC power supply.



① Supply voltage +24 V DC
② Supply voltage ground
③ Spring release

Refer to the wiring regulations (Page 97) table for information on the maximum conductor cross-section.

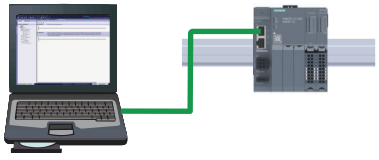
Disconnecting the wires

Using the screwdriver, press the spring release down as far as it will go and pull out the wire.

3.1.1 Configuring the CPU for communication

3.1.1.1 Overview

A CPU can communicate with a STEP 7-Micro/WIN SMART programming device in the network:



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

Note

- Support installing STEP 7-Micro/WIN SMART (Version 2.8) and STEP 7-Micro/WIN SMART V3 on the same computer.
- STEP 7-Micro/WIN SMART (Version 2.8) can connect to CPU of Version 2.8 or earlier, STEP 7-Micro/WIN SMART V3 can only connect to CPU of version 3.0 or later.
- 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:

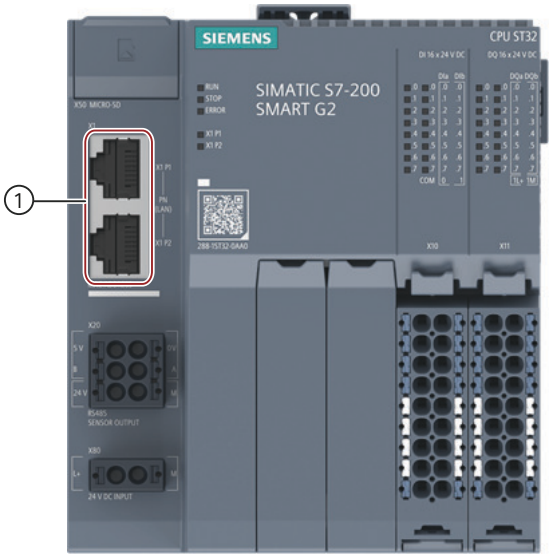
- Configuration/Setup: No hardware configuration is required for a single CPU. If you want multiple CPU's on the same network, then you must change the default IP addresses to new, unique IP addresses.
- Dual network ports, multiple can be daisy-chained

3.1.1.2 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. Plug the Ethernet cable into the Ethernet port on the top left of the CPU as shown below.
3. Connect the Ethernet cable to the programming device.



① PROFINET (LAN) ports

3.1.1.3 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.

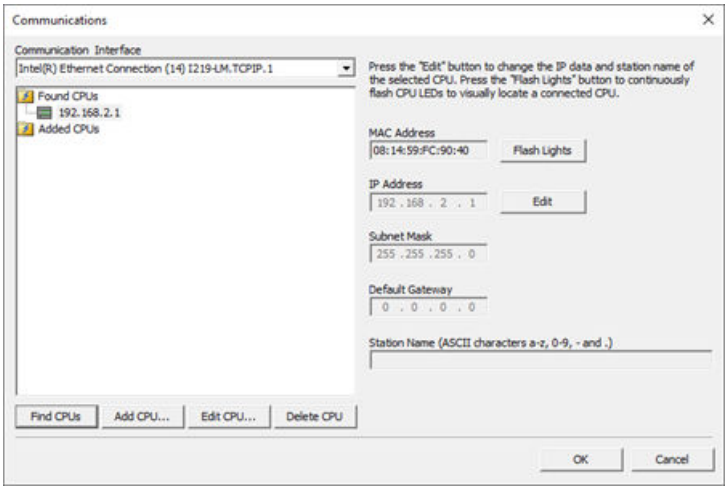
- From the project tree, double-click the "Communications" node.
- Click the "Communications" button from the navigation bar.
- Select "Communications" from the "Component" drop-down list in the Windows area of the "View" menu ribbon strip.

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 CPU" and is retained.

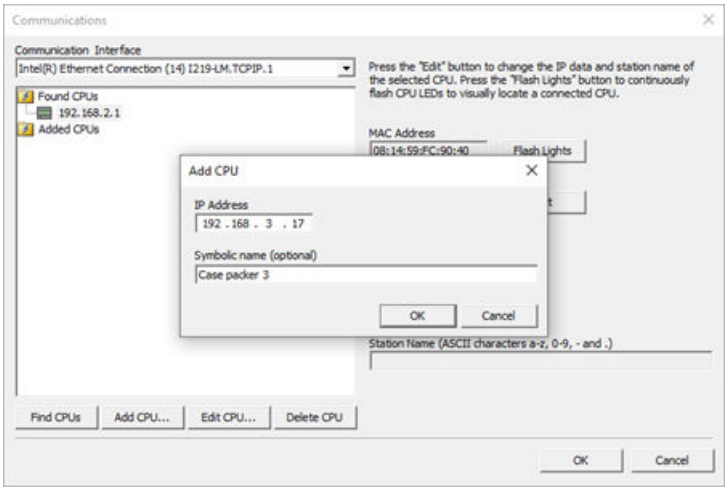
Note

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



For "Found CPUs" (CPUs located on your local network), use the "Communications" dialog to connect with your CPU:

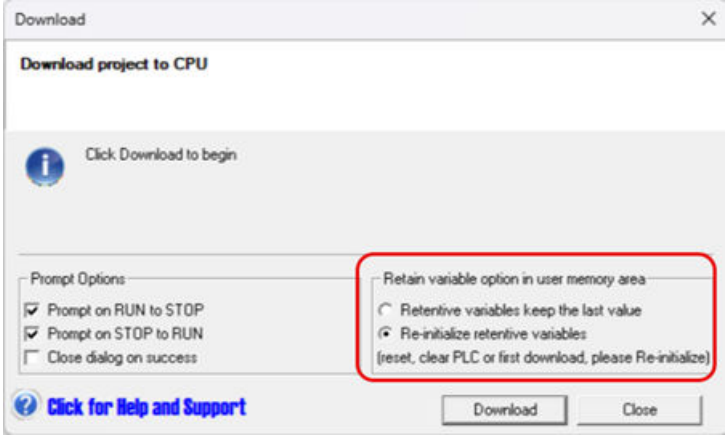
- Select TCP/IP for your Communication Interface.
- Click the "Find CPU" button to display all operational CPUs ("Found CPU") on the local Ethernet network. All CPUs have a default IP address. See the "Note" below.
- Highlight a CPU, and then click "OK".



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.

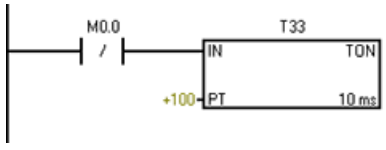
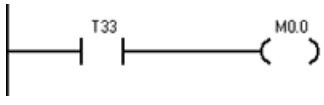
See the "Configuring or changing an IP address for a CPU or device in your project" (Page 492) for information about how to accomplish this.

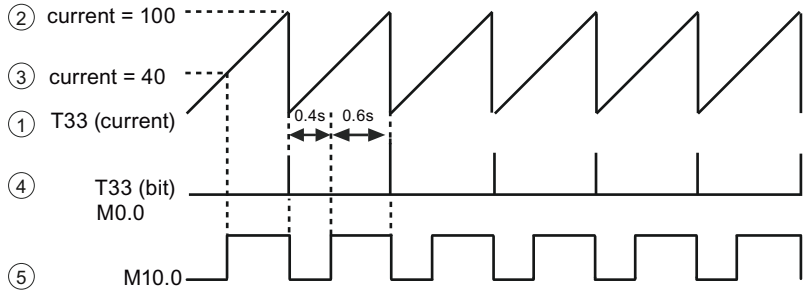
3.2 Creating a sample program

Entering this example of a control program will help you understand how easy it is to use STEP 7-Micro/WIN SMART. 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.

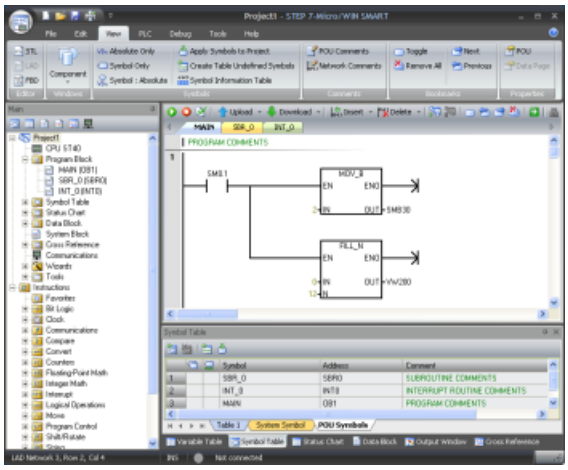
Table 3-1 Sample program for getting started with STEP 7-Micro/WIN SMART

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.

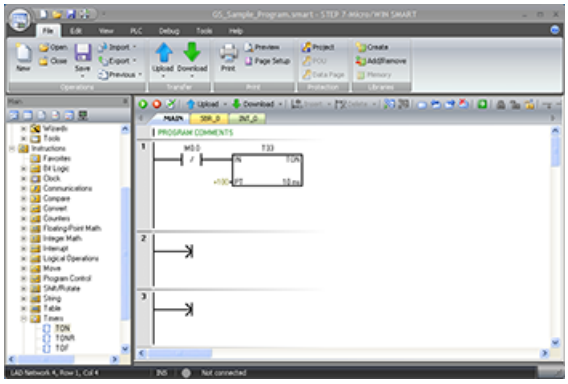
The Program Block folder in the project tree contains all of the blocks of your program.

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.

3.2.1 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.

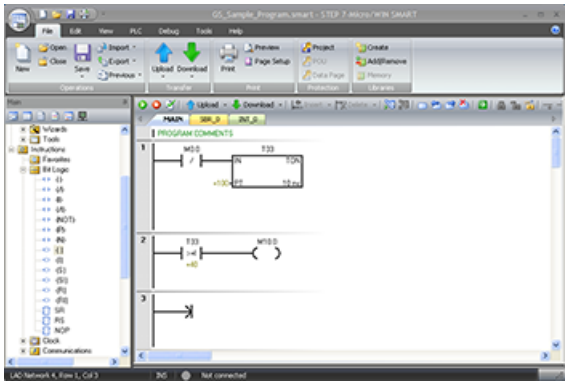
To enter the timer instruction for T33:

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.

3.2.2 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:

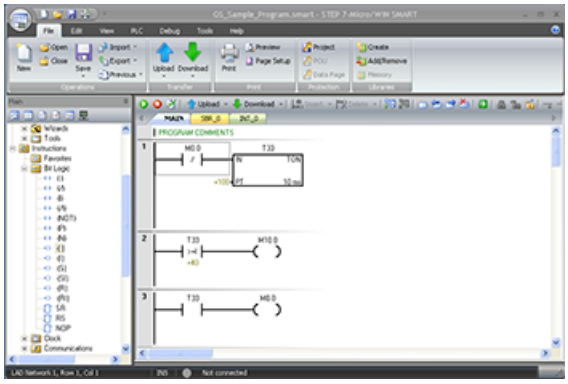
1. Double-click the Compare icon to display the compare instructions. Select the ">=I" instruction (greater-than-or-equal-to-integer).
2. Hold down the left mouse button and drag the compare instruction onto the second network.
3. Click "???" above the contact and enter the address for the timer value: 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.

3.2.3 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.

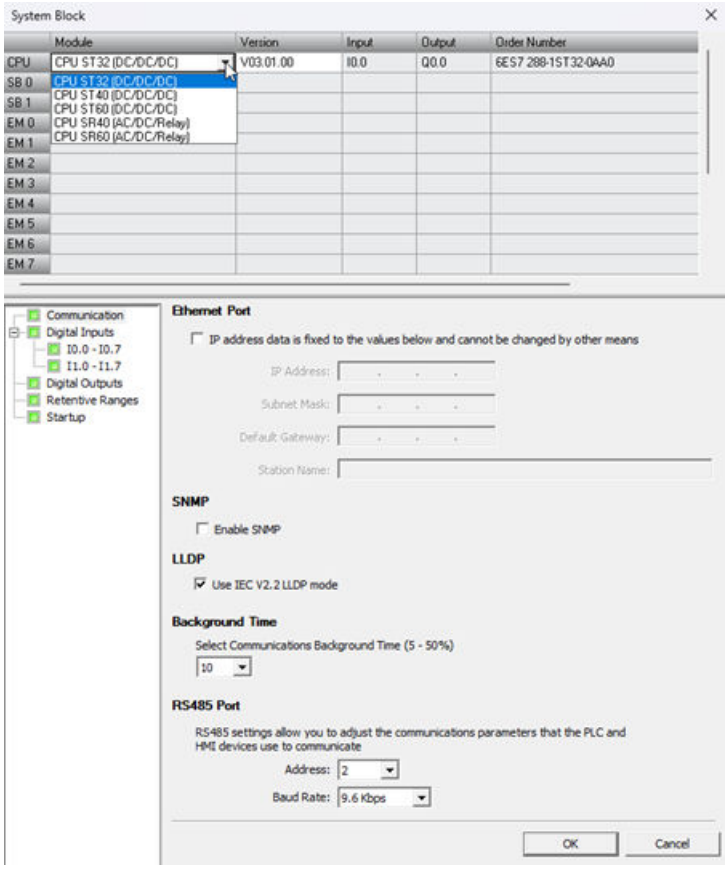
To enter the coil for turning on M0.0:

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.

3.2.4 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.



3.2.5 Saving the sample project

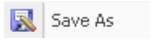
Saving the sample project

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.

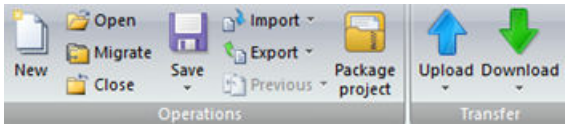


3. Enter a name for the project in the "Save As" dialog.
4. Browse to a location where you want to save your project.
5. Click "Save" to save the project.

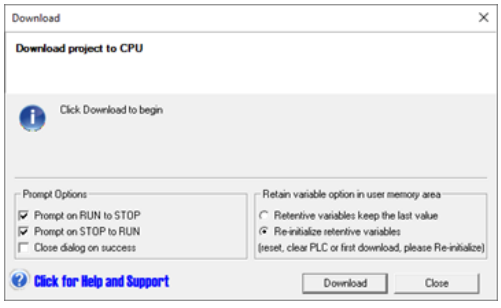
After saving the project, you can download the program to the CPU.

3.3 Downloading the sample program

First, ensure that your network hardware and PLC connector cable for either Ethernet (Page 73) (standard CPUs only) or RS485 (Page 532) 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.

3.4 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.

place 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 432) in your program logic to put the CPU in STOP mode.

Installation

4

4.1 Guidelines for installing S7-200 SMART G2 devices

The S7-200 SMART G2 devices are designed to be easy to install. The S7-200 SMART G2 can be installed horizontally or vertically on a standard DIN rail. The small size of the S7-200 SMART G2 allows you to make efficient use of space.

⚠ WARNING

Safety requirements for installing S7-200 SMART G2 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 G2. 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 G2 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.

Secure connection

In applications with significant environmental vibration or during vertical installation, it is recommended to install DIN rail end caps (8WA1 805) on both sides of the system to ensure the modules remain securely connected.

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.

Allow sufficient space for heat dissipation and wiring

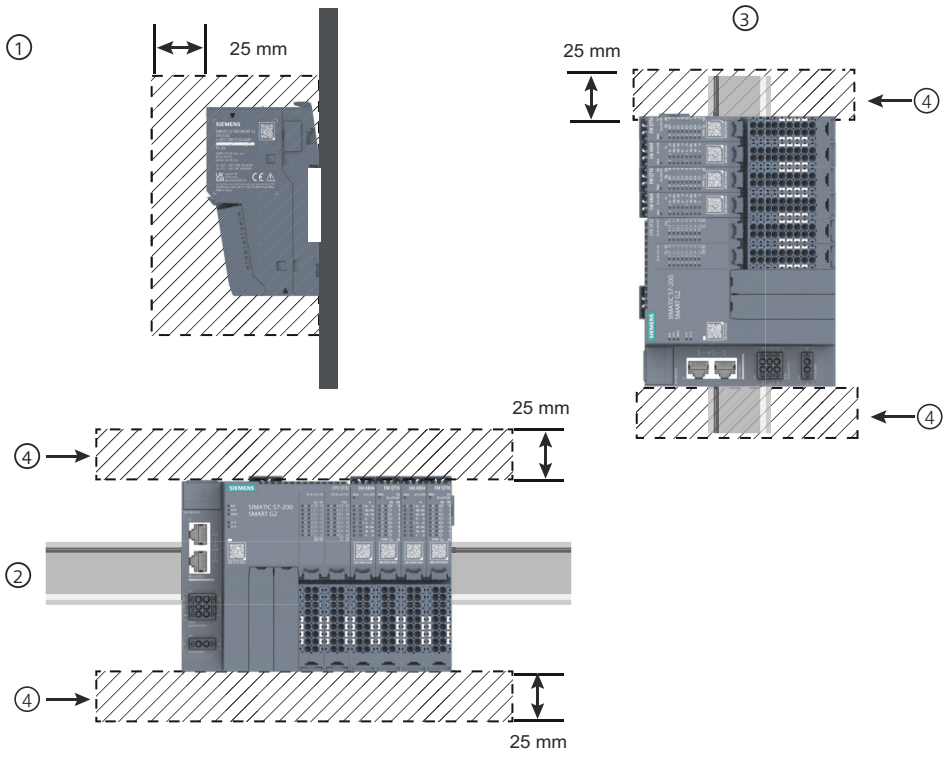
The S7-200 SMART G2 devices are designed for natural convection cooling. For proper cooling, you must provide a clearance of at least 25 mm above and below the devices. Also, allow at least 25 mm of depth between the front of the modules and the inside of the enclosure.

⚠ CAUTION

Temperature considerations

For vertical mounting, the maximum allowable ambient temperature is reduced by 10 degrees C. Operating with large temperature fluctuations outdoors could result in erratic process operation and could result in minor personal injury.

If your installation includes expansion modules, it is recommended to 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
② Horizontal installation
③ Vertical installation
④ Clearance area

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

Installation rules for reducing thermal load

The following rules help reduce the thermal stress on the S7-200 SMART G2 system within the control cabinet.

- Horizontal installation should be prioritized
- During vertical installation, high-power consumption modules should be inserted at the top, and low-power consumption modules at the bottom
- Install CPU stations with high-power consumption modules in the lower section of the control cabinet
- In multi-layer configurations, high-power consumption modules should be inserted on both sides to allow waste heat to dissipate upward unobstructed

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

There is a risk of electric shock when installing or removing the equipment

Performing live installation or wiring work on the S7-200 SMART G2 or related equipment may result in electric shock or unintended activation of the device.

Always follow appropriate safety precautions and ensure the power to the S7-200 SMART G2 or related equipment is disconnected before installing or removing any devices.

Failure to disable all power during installation or removal of the S7-200 SMART G2 or related equipment can result in death, severe personal injury and/or property damage due to electric shock or unexpected equipment operation.

Always ensure that whenever you replace or install an S7-200 SMART G2 device you use the correct module or equivalent device.

Always follow the grounding guidelines (Page 97).

WARNING

Risks associated with replacing S7-200 SMART G2 device

Improper installation of the S7-200 SMART G2 module may cause abnormal operation of the program in the S7-200 SMART G2.

When replacing S7-200 SMART G2 equipment, use the same model, orientation, and sequence.

Otherwise, unintended operation of the equipment may occur, resulting in death, serious personal injury, and/or property damage.

WARNING

Installing or removing equipment in a flammable or combustible environment

Disconnecting equipment in a flammable or combustible environment may cause a fire or explosion.

Do not disconnect equipment in a flammable or combustible environment; when working in such environments, follow appropriate safety precautions.

A fire or explosion can result in death, serious injury and/or property damage.

NOTICE

Risks associated with electrostatic discharge

Electrostatic discharge can damage the memory card or the receptacle on the CPU. If damaged, the card slot or memory card may malfunction or become inoperable.

To protect memory card and receptacle from electrostatic discharge, perform the following operations:

- Make contact with a grounded conductive pad or wear a grounded wrist strap whenever you handle the memory card.
- Store the memory card in a conductive container.

Receptacle or memory card malfunction can result in property damage.

4.2 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 of the specific CPU (Page 24) to determine the 24 V DC sensor power budget. Refer to the Calculating power budget (Page 1179) to determine how much power the CPU can provide for your configuration.

The CPU provides the 5 V DC logic power needed for any expansion in your system.

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 CPU also provides a 24 V DC sensor provide 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.

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 G2 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 G2 system are connected to the same reference potential.

Refer to the technical specifications of the specific CPU (Page 24) to determine the 24 V DC sensor power budget.

4.3 Installation and removal procedures

4.3.1 S7-200 SMART G2 devices installation dimensions

Use the following installation dimensions to ensure proper installation of the equipment. All installation dimensions are in millimeters (mm).

CPU ST32

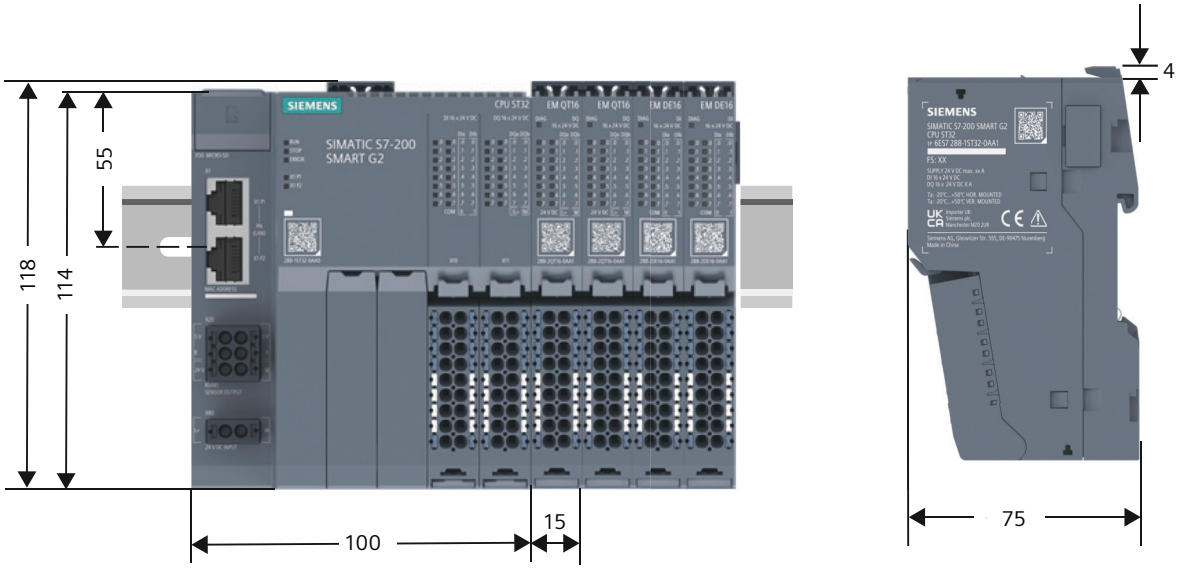


Table 4-1 Mounting dimensions (mm)

S7-200 SMART G2 device		Height A (mm)	Width B (mm)
CPU	CPU ST32, DC/DC/DC	118	100
Signal modules	CM01	30	15
Expansion module	DE16 (16DI)	118	15
	QT16 (16DQ)	118	15
	AE04 (4AI)	118	15
	AQ04 (4AQ)	118	15
	AE08 (8AI)	118	15
	AM06 (4AI 2AQ)	118	15
	AR04 (4RTD)	118	15
	AT08 (8TC)	118	15

Each CPU and EM supports installation on a DIN rail. Use the DIN rail clips on the module to secure the device on the rail. These clips also snap into an extended position to provide screw mounting positions to mount the unit directly on a panel.

Always follow the grounding guidelines (Page 97).

4.3.2 Installing and removing the CPU

The CPU can be easily installed on a standard DIN rail.

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

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 exchange

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.

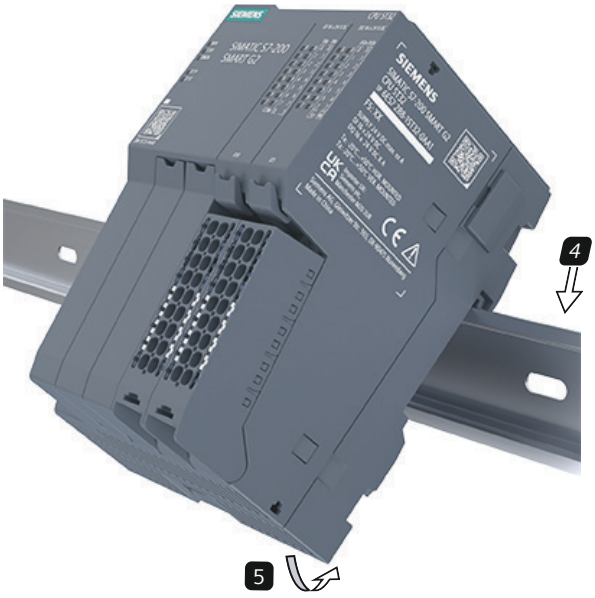
Replace the device with the same model, and be sure to orient and position it correctly.

Installing the CPU on a DIN rail

Follow the steps below to install a CPU on a DIN rail.

1. Install the DIN rail; secure the rail to the mounting panel every 75 mm.
2. Ensure that the CPU and all S7-200 SMART G2 devices are powered off. Confirm that the mounting rail release button on the CPU is not pressed.

3. Hook the CPU over the top of the DIN rail.
4. Rotate the CPU downward until a "click" is heard, indicating that the mounting rail release button has locked into place.



Removing a CPU from a DIN rail

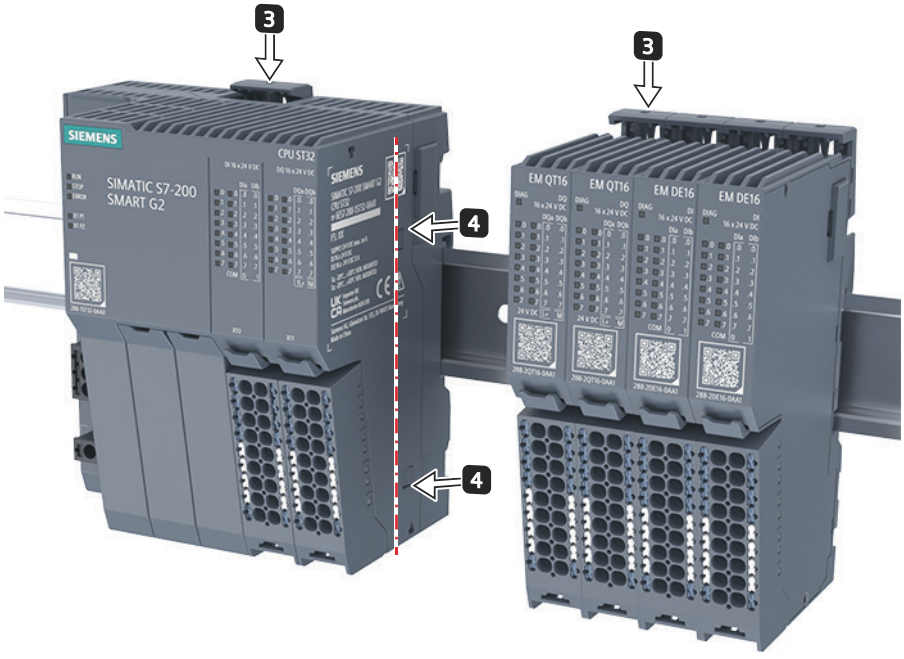
Follow the steps below to remove a CPU from a DIN rail.

1. Remove power from the CPU and any attached expansion modules.
2. Disconnect all the wiring and cabling that is attached to the CPU.
3. Press the mounting rail release buttons on the CPU and its adjacent modules to release them from the rail.

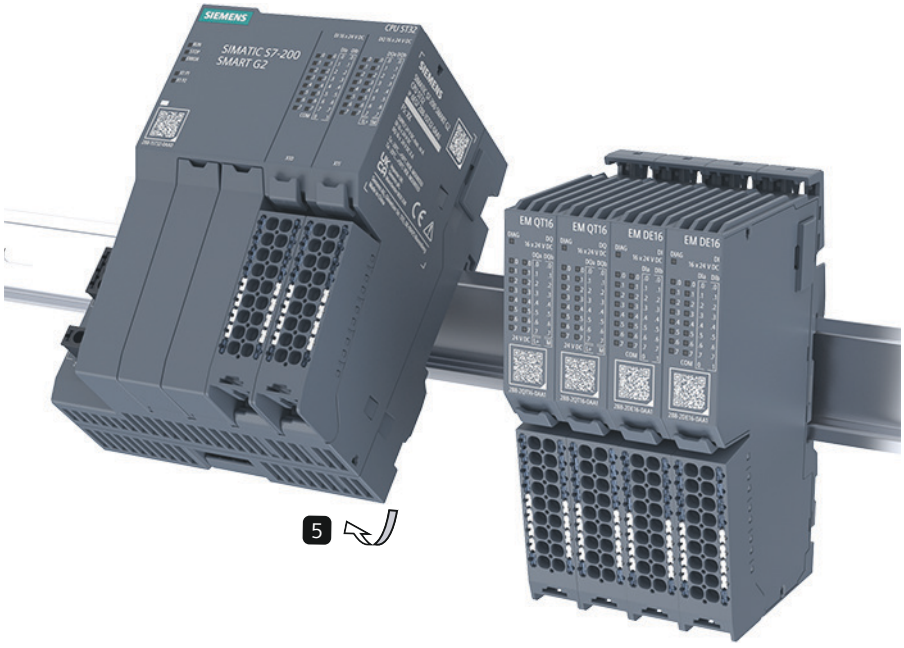
Note

The mounting rail release button is located above the CPU or expansion module.

4. If you have expansion modules connected, slide the CPU to the left to disengage it from the expansion module connector.



5. Rotate the CPU up and off the rail, and remove the CPU from the system.

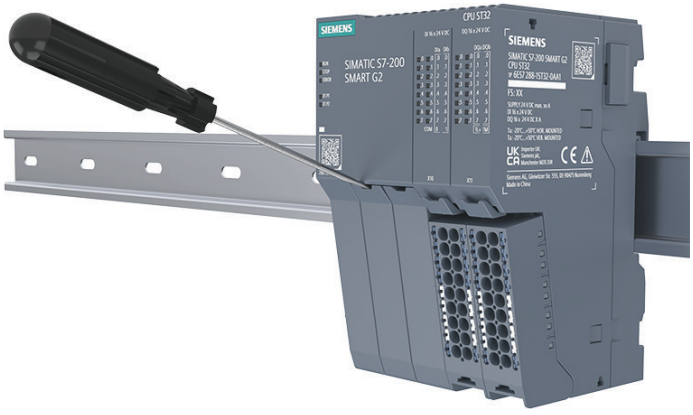


4.3.3 Installing and removing a signal board

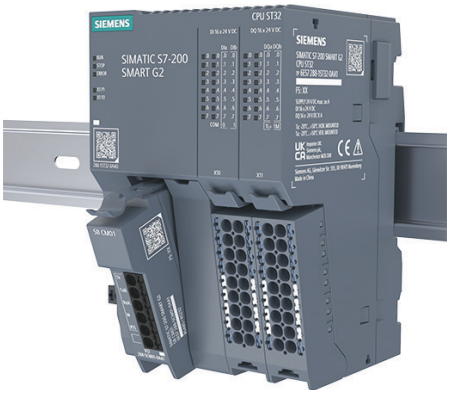
Installing a signal board on a CPU

Follow the steps below to install a signal board

1. Ensure that the CPU and all S7-200 SMART G2 equipment are disconnected from electrical power.
2. Disconnect all the wiring and cabling that is attached to the CPU.
3. Insert an M3 screwdriver into the reserved slot above the signal board, then lever open the blank cover plate and pull it away from the CPU to remove it. Retain the board cover for reuse.



4. Hook the lower end of the signal board into the lower end of the CPU expansion slot, then press down until it clicks into place.



Removing a signal board on a CPU

Follow the steps below to remove a signal board

1. Ensure that the CPU and all S7-200 SMART G2 equipment are disconnected from electrical power.
2. Remove the module straight up from its mounting position in the top of the CPU. Please refer to Replacing terminal block (Page 106).
3. Replace the cover onto the CPU.

4.3.4 Installing and removing an expansion module

Install expansion modules separately after the CPU has been installed.

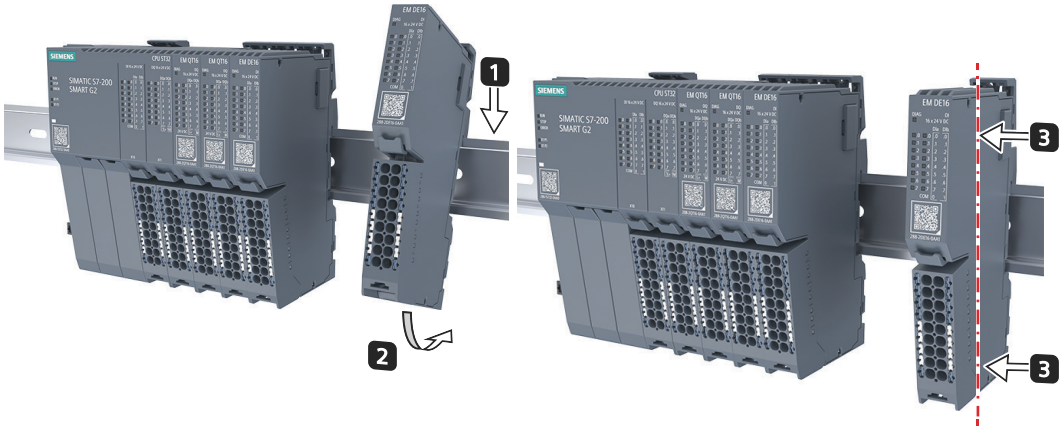
Installing expansion modules

Follow the steps below to install an expansion module:

1. Ensure that the CPU and all S7-200 SMART G2 equipment are disconnected from electrical power.
2. Remove the cover for the I/O bus connector from the right side of the CPU or expansion module. Insert the screwdriver into the slot above the cover and gently lever to release and remove the cover. Retain the cover for reuse.

Connect the expansion module to the CPU.

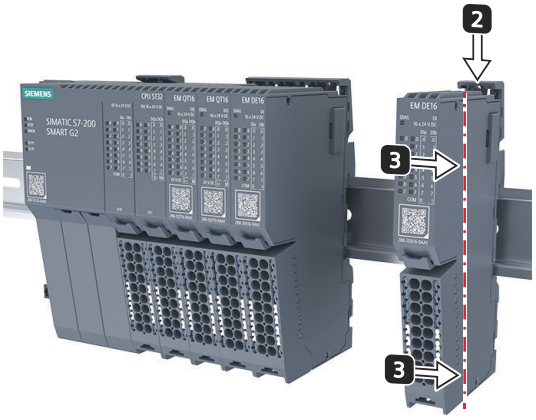
1. Install the expansion module on the mounting rail. Position the expansion module to the right of the CPU. Confirm that the mounting rail release button on the expansion module is not pressed.
2. Rotate the expansion module downward until a "click" is heard, indicating that the mounting rail release button has locked into place.
3. Align the pins of the module's I/O bus connector with the holes on the CPU or module, then firmly press the assembly until the pins click into place.



Removing an expansion module

Follow the steps below to remove an expansion module:

1. Ensure that the CPU and all S7-200 SMART G2 equipment are disconnected from electrical power.
2. On the expansion module to be removed, firmly press the mounting rail release button.



3. Slide the expansion module to the right.
4. Rotate the expansion module upward and remove it from the rail.

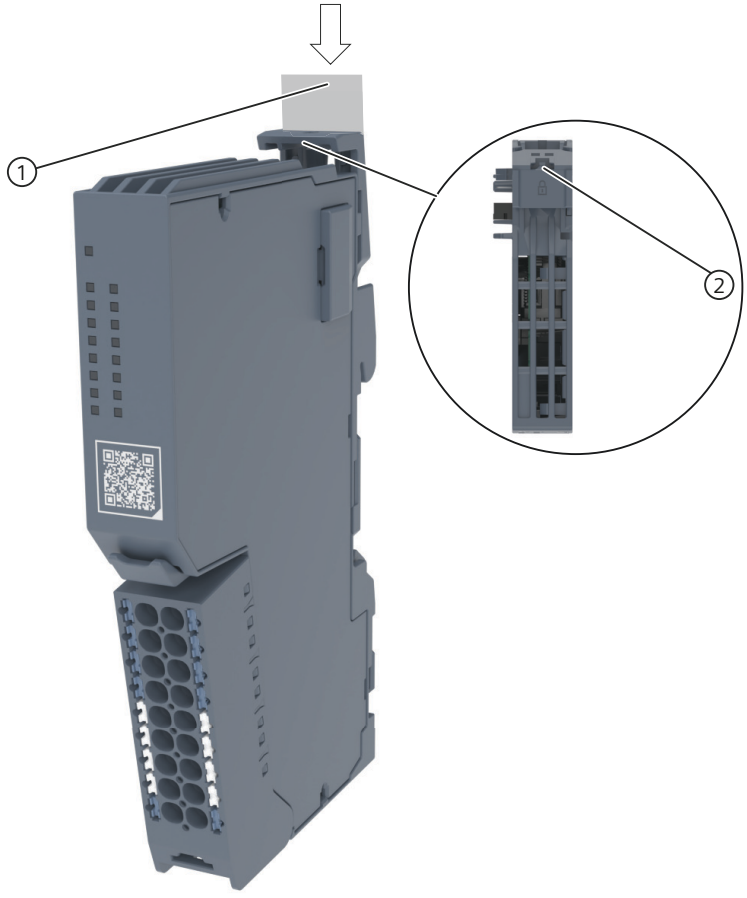
4.3.5 Labeling S7200 SMART G2

4.3.5.1 Attaching the identification label

Procedure

To attach the identification label, follow these steps:

1. Break off the reference identification labels from the sheet.
2. Attach the reference identification labels to the CPU and expansion modules. This slot is located on the top of the CPU or expansion module.



① Reference identification labels
② Label insertion slot

Note

Reference identification labels

The printable side of the reference identification label must be facing forward.

4.4 Wiring guidelines

4.4.1 Prerequisites

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 S7-200 SMART G2. For the wiring diagram of the S7-200 SMART G2, please refer to the technical specifications (Page 1146).

Prerequisites

Before grounding or wiring any electrical equipment, ensure that the power to the device is turned off. Also, ensure that the power to all related equipment has been turned off.

⚠ WARNING

There is a risk of electric shock when installing or removing the equipment

Performing live installation or wiring work on the S7-200 SMART G2 or related equipment may result in electric shock or unintended activation of the device.

Always follow appropriate safety precautions and ensure the power to the S7-200 SMART G2 or related equipment is disconnected before installing or removing any devices.

Failure to disable all power during installation or removal of the S7-200 SMART G2 or related equipment can result in death, severe personal injury and/or property damage due to electric shock or unexpected equipment operation.

Always take safety into consideration as you design the grounding and wiring for the S7-200 SMART G2 system. Follow all applicable electrical codes when wiring the S7-200 SMART G2 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

Risk of unintended equipment operation

Control devices may malfunction under unsafe conditions, leading to unintended equipment operation.

Use an emergency stop function, electromechanical overrides or other redundant safeguards that are independent of the S7-200 SMART G2.

Such unexpected operations could result in death, severe personal injury and/or property damage.

The S7-200 SMART G2 AC power supply boundaries and I/O boundaries to AC circuits ensure safe isolation between the AC line voltage 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 that use AC line voltage are safely isolated from other circuits. The isolation boundaries between 24 V DC circuits serve a limited function and should not be relied upon for safety.

The sensor supply output, communications circuits, and internal logic circuits of an S7-200 SMART G2 with included AC power supply are sourced as SELV (safety extra-low voltage) according to EN 61131-2.

To ensure safe operation, certified power supplies that comply with SELV/PELV, Class 2, limited voltage, or limited power requirements must be used when providing power to the following:

- Communication ports
- Analog circuits
- 24V DC nominal power supply
- Digital I/O circuits

⚠ WARNING

Using non-isolated or single-layer insulated power supplies poses a risk of electric shock

The use of non-isolated or single-layer insulated power supplies may cause circuits that are typically safe to touch, such as communication circuits and low-voltage sensor lines, to carry hazardous voltages.

Do not use non-isolated or single-layer insulated power supplies to power low-voltage circuits from AC line voltage.

Only use high voltage to low voltage power converters that are approved as sources of touch safe, limited voltage circuits.

The use of non-isolated or single-layer insulated power supplies may result in accidental electric shock, leading to fatalities, serious injuries, and/or property damage.

⚠ 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.

When the S7-200 SMART G2 is mounted on a DIN rail, the device uses a spring to establish a proper grounding connection with the DIN rail. Ensure the DIN rail is properly grounded to the system's single-point ground.

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

When planning the wiring for the S7-200 SMART G2, provide a disconnect switch that can simultaneously remove power from the S7-200 SMART G2 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.

The transistor outputs of the CPU do not have integrated overcurrent protection. If a short circuit occurs at the output or the load current exceeds its rated capacity, the absence of external overcurrent protection can easily lead to damage of the MOSFET. Therefore, it is recommended to add external overcurrent protection in the Q-point power supply 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 appropriately sized copper end sleeves to connect the wire terminals to the connector.

To reduce electrical noise, use shielded cables.

In addition to using shielded cables, the following measures can be taken to further reduce electrical noise:

- Connect cable shield to ground at both ends.
- Use clamps or copper tape around the shielding to provide a larger grounding contact surface and ground the shields of other cables.

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 G2 because the sensor supply is already current-limited.

All S7-200 SMART G2 modules have removable connectors for user wiring. To prevent loose connections, ensure that the connector is seated properly, locked in place and that the wire is installed securely into the connector.

Refer to the insulation guidelines (Page 97) and technical specifications (Page 1146) for the amount of isolation provided and the location of the isolation boundaries.

The following table summarizes the wiring rules for the S7-200 SMART G2 CPU, EM, and SB.

Table 4-2 Wiring rules for S7-200 SMART G2 CPUs, EMs, and SBs

Wiring rules for...	**CPU**		**SB**	**EM**
	RS485(2x3)	**Power in(2x1)**		
Conductor cross-sectional area for solid cables (copper)	IEC: 0.2 to 1.5 mm^2			
	UL: 24 to 14 AWG*			

Wiring rules for...		CPU		SB	EM
		RS485(2x3)	Power in(2x1)		
Cross-sectional area of flexible cable (copper cable)	Without end sleeve	IEC: 0.2 to 1.5 mm²**			
		UL: 24 to 14 AWG*			
	With end sleeve (with plastic collar)*** Compliant with 46228-4 or UL486F	IEC: 0.2 mm to 1.5 mm²			
		UL: 24 to 16 AWG*			
	With TWIN end sleeve***	IEC: 0.2 mm to 0.5 mm²	IEC: 0.2 mm to 0.75 mm²	IEC: 0.2 mm to 0.5 mm²	IEC: 0.2 mm to 0.75 mm²
	In accordance with UL486F	UL: 24 to 20 AWG*	UL: 24 to 18 AWG*	UL: 24 to 20 AWG*	UL: 24 to 18 AWG*
Stripped length of conductors		10 mm			
End sleeve with plastic collar***	0.2 mm²≤ cable cross-section ≤0.5 mm²	8 to 10 mm in length			
	0.75 mm²≤ cable cross-section ≤1.5 mm²	10 mm long			
TWIN end sleeve		12 mm long			

End sleeve, conforming to DIN 46228 and UL486F, with plastic collar***				Maximum cable stripping length (mm)
Conductor cross-section		Crimping sleeve length (mm)	Type of crimping sleeve	
mm²	AWG*			
0.14	26	8	AI 0.25-8	10
0.25	24	8	AI 0.25-8	10
		10	AI 0.25-10	10
0.34	22	8	AI 0.34-8	10
		10	AI 0.34-10	10
0.5	20	8	AI 0.5-8	10
		10	AI 0.5-10	10
0.75	18	10	AI 0.75-10	13
1		10	AI 1-10	13
1.5	16	10	AI 1.5-10	13

* AWG: American wire gauge

** End sleeve, without plastic collar: 0.25 to 2.5 mm²/AWG: 24 to 13

*** Please refer to the notes for the end sleeve

⚠ DANGER

Use copper wires with terminal connectors

Use copper (Cu) cables for all power wires connected to the device via terminals, for example, the 24 V DC power cable connected to the 24 V DC power connector.

Utilisez des câbles en cuivre sur les connexions à bornes

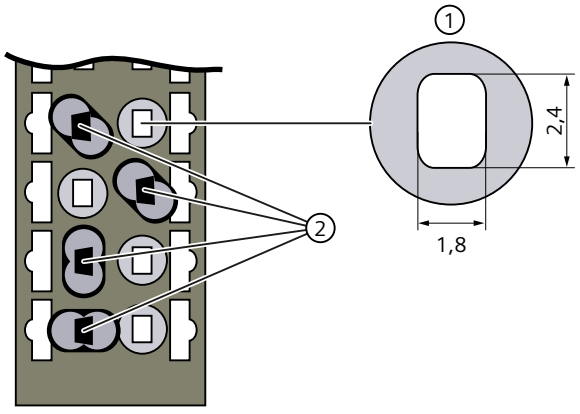
Utiliserz des câbles en cuivre (Cu) pour tous les câbles d'alimentation qui sont raccordés à l'appereil par des bornes, par example les câbles d'alimentation 24 V CC sur le connecteur d'alimentation 24 V CC.

Note

Using copper terminal sleeves on standard copper conductors can reduce the risk of short circuits caused by loose wires. Ends longer than the recommended stripping length must include an insulation ring to prevent short circuits caused by lateral movement of the conductor. The cross-sectional area limitations for bare conductors also apply to end pieces.

TWIN end sleeve for I/O module through-hole terminal cable

When crimping TWIN end sleeves with a cross-sectional area of 0.75 mm^2, ensure the cables are arranged at an appropriate angle to achieve optimal alignment, as the TWIN end sleeves require a certain amount of space.



① Cross-sectional area of the terminal compartment
② Crimp the TWIN end sleeve at the appropriate angle.

Figure 4-1 Twin end sleeve

Cable temperature measurement threshold

Note

Cable temperature measurement threshold

When selecting cables, note that the operating temperature of the cable can be up to 45 °C higher than the ambient temperature of the S7-200 SMART G2 system. (For example, when the ambient temperature is 60 °C, the connected conductors must have a temperature rating of at least 90 °C.)

You should specify other connection types and required materials based on the electrical characteristics of the circuits used and the installation environment.

Guidelines for lamp loads

The inrush current when a tungsten filament lamp is switched on is typically 10 to 15 times the steady-state current. A replaceable interposing relay or surge limiter is recommended for lamp loads requiring frequent switching operations.

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 generated by inductive loads without effective suppression measures can interrupt PLC operation. 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 G2 DC outputs include internal suppressor circuits that are adequate for inductive loads in most applications. Since S7-200 SMART G2 relay output contacts can be used to switch either a DC or an AC load, internal protection is not provided.

For suppressing electromagnetic interference caused by inductive loads (such as contactors), the following solutions are recommended: Prioritize products that come with built-in suppression circuits from the equipment manufacturer, or configure dedicated suppression modules provided by them as optional accessories. 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.

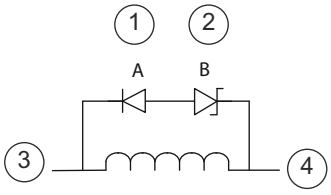
A well-controlled turn-off transient will have a ring frequency of no more than 10 kHz, with less than 1 kHz preferred. Peak voltage for AC lines should be within +/- 1200 V of ground. Negative peak voltage for DC loads using the PLC internal suppression will be ~40 V below

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.

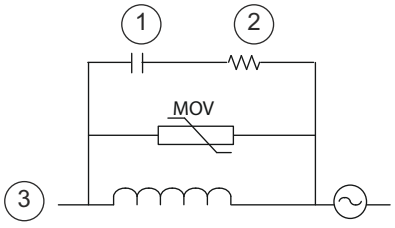
Typical suppressor circuit for DC or relay outputs that switch DC inductive loads



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 4-3 AC suppressor circuit resistor and capacitor values

Inductive load			**Suppressor values**		
I rms	230 V AC	120 V AC	Resistor		Capacitor
A	VA	VA	Ω	W (power rating)	nF
0.02	4.6	2.4	15000	0.1	15

Inductive load			Suppressor values		
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

The values in the table satisfy the following conditions:
Maximum turn-off transient 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.

4.4.2 I/O terminal wiring

Introduction

Note

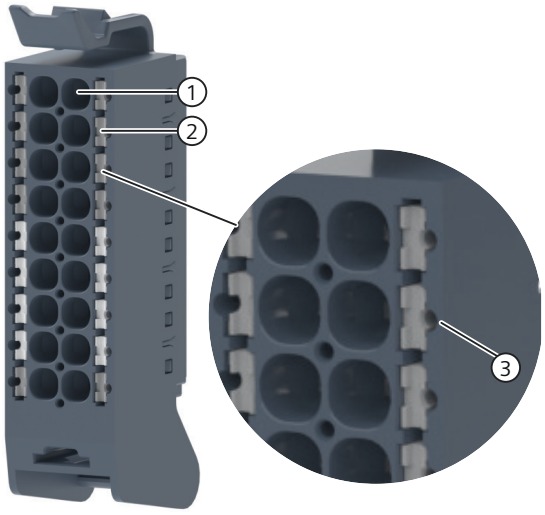
The pin assignment of the expansion module depends on the specific expansion module being used. You can find information about the corresponding expansion module in the technical specifications (Page 1153).

Requirements

- The supply voltage is switched off.
- 3 to 3.5 mm flathead screwdriver

Push-in system

The following diagram shows a view of the push-in terminal.



① Push-in terminals
② Spring release
③ Test probe opening (applicable test probe: diameter 1 mm, length ≥ 10 mm, compliant with the permitted voltage category)

Tool-free cable connection: Single-wire cables without conductor end sleeves, multi-wire cables with conductor end sleeves or ultrasonically sealed

To connect wires without end sleeves, follow these steps:

1. Strip 8 to 10 mm of the wire.
2. Only standard conductors (excluding those with a cross-sectional area of 2.5 mm^2): Seal or crimp the wires using end sleeves.
3. Insert the cable as deep as possible into the push-in terminal.

Connection of cables: Stranded cable, without end sleeves, rough cut

To connect wires without end sleeves, follow these steps:

1. Strip 8 to 10 mm of the wire.
2. Use a screwdriver to push the wire into the spring release.
3. Insert the cable as deep as possible into the push-in terminal.
4. Remove the screwdriver from the spring release.

Remove wiring

Use a screwdriver to press down the spring release of the terminal as far as possible and pull out the wire.

Note

Do not pull the wire/cable while pressing the spring release to avoid damaging the terminal.

4.5 Maintenance

4.5.1 Replacing terminal block

Introduction

The terminal block is a component of the CPU and expansion modules. If necessary, you can replace the terminal block. For this, you do not need to remove the CPU or expansion modules. When replacing the terminal block, the power supply bus for the load current of the potential group will not be interrupted.

Requirements

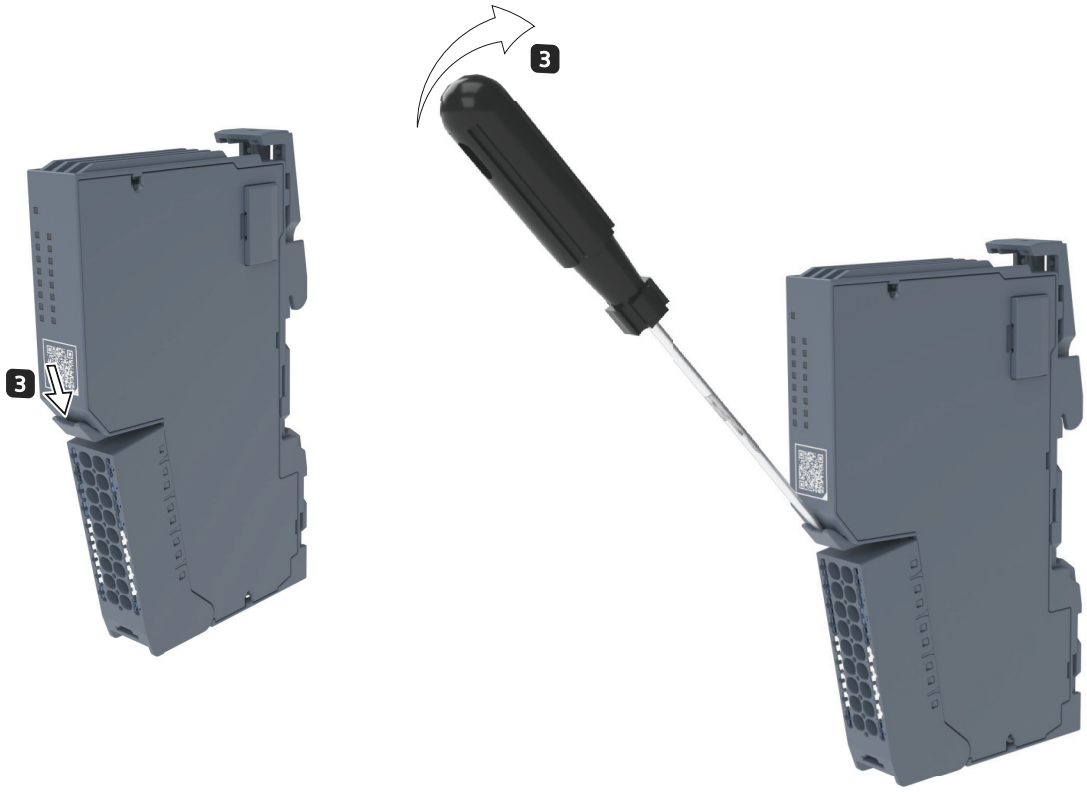
- The terminal block may only be replaced when the supply voltage is switched off.
- The CPU or expansion module is installed and wired.

Procedure

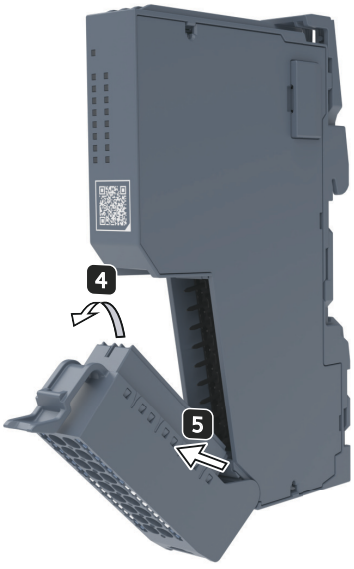
To replace the terminal block, follow these steps:

1. Disconnect the power supply voltage on the module.
2. Disconnect the wiring on the terminal block.

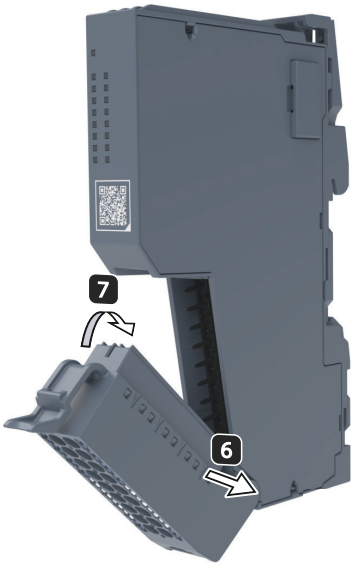
3. Press the release mechanism with your finger. The release mechanism of the terminal block is located at the top of the terminal block.



4. Tilt the released terminal block forward.
5. Remove the terminal block from the expansion module.



6. Insert the new terminal block into the expansion module.
7. Rotate the new terminal block backward into the expansion module until it clicks into place.
8. Wire the terminal block.



9. Turn on all existing power supply voltages on the expansion module.

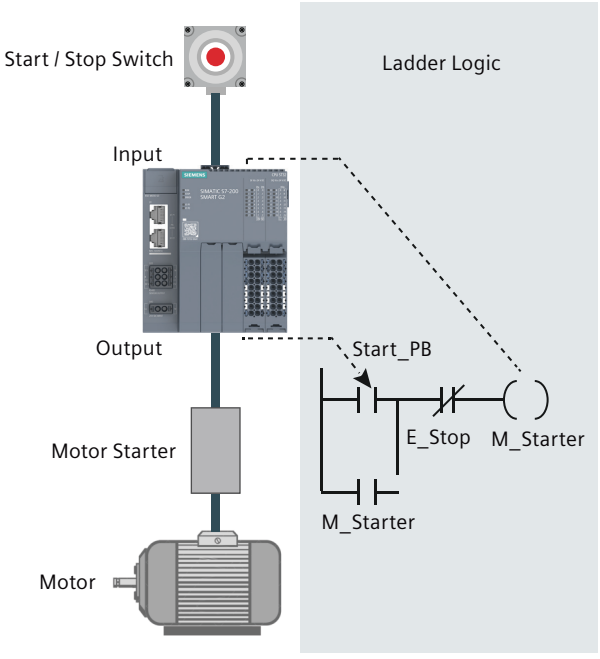
PLC concept

5

5.1 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.

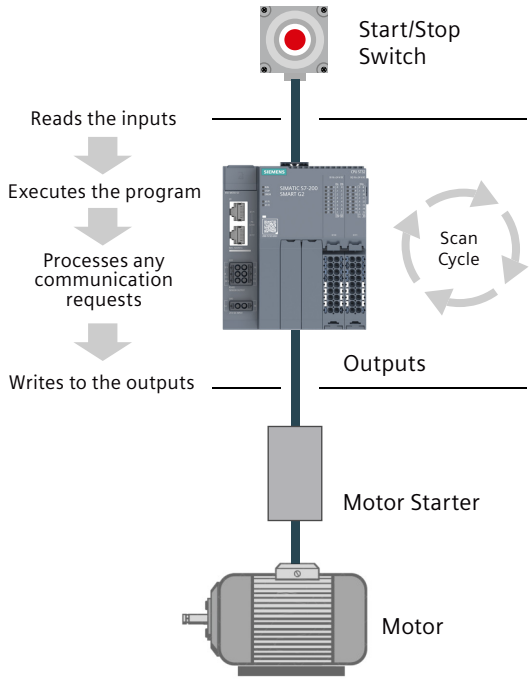


The figure shows a simple diagram of how an electrical relay diagram relates to the CPU. In this example, the state of the switch for starting the motor is combined with the states of other inputs. The calculations of these states then determine the state for the output that goes to the actuator which 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 5-1 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.

5.1.1 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.

5.1.2 Immediate reading or writing of the I/O

The CPU instruction set provides commands for immediately reading from or writing to physical I/O. These immediate I/O instructions allow direct access to actual output or input points, even though image registers are typically used as the source or destination addresses for I/O access. When using immediate instructions to access input points, the corresponding process image input registers remain unchanged. When using immediate instructions to access output points, the corresponding process-image output registers are updated simultaneously.

Note

When you read an analog input, the value is read immediately. When you write a value to an analog output, the output is updated immediately.

During program execution, using the process image register offers advantages over directly accessing input or output points. There are three reasons for using the image registers:

- The sampling of all inputs at the start of the scan synchronizes and freezes the values of the inputs for the program execution phase of the scan cycle. The outputs are updated from the image register after the execution of the program is complete. This provides a stabilizing effect on the system.
- Your program can access the image register much more quickly than it can access I/O points, allowing faster execution of the program.
- I/O points are bit entities and must be accessed as bits or bytes, but you can access the image register as bits, bytes, words, or double words. Thus, the image registers provide additional flexibility.

5.1.3 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.

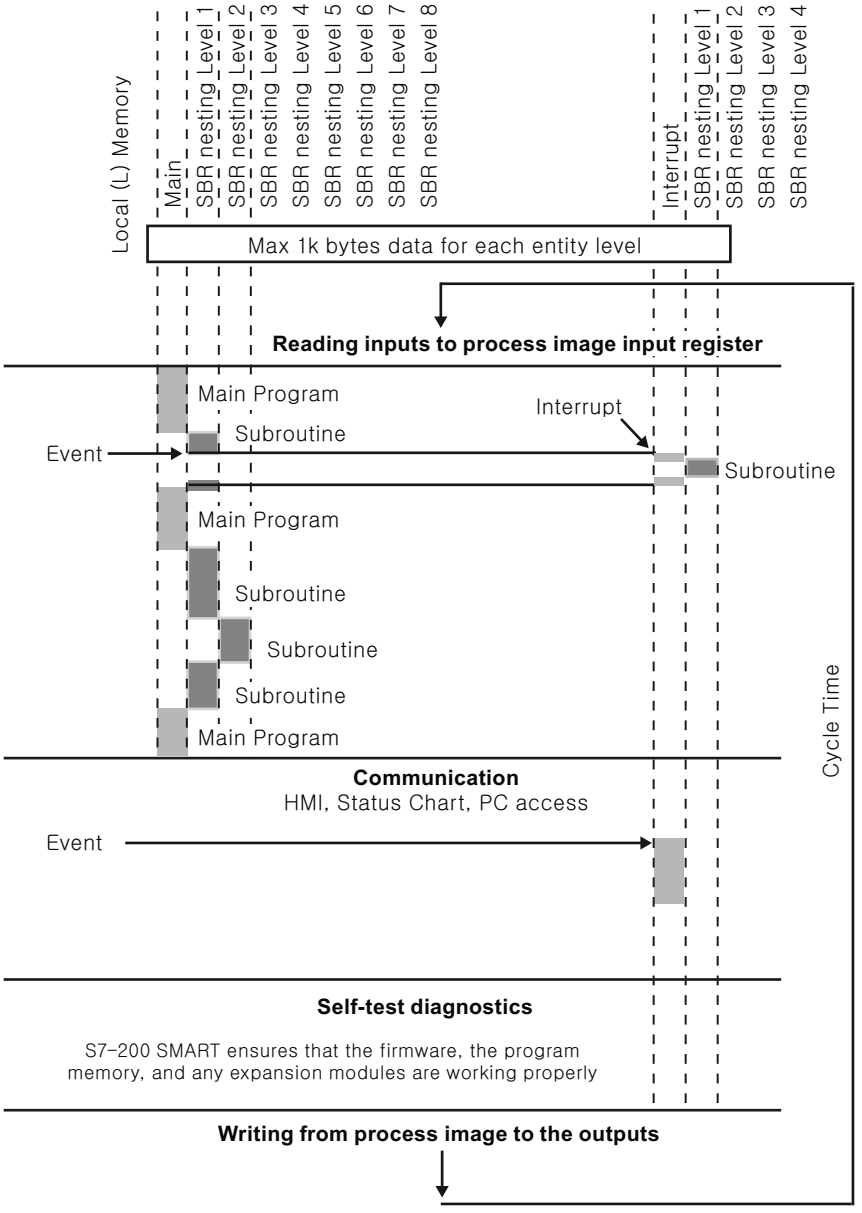


Figure 5-1 Typical_Scan_Flow

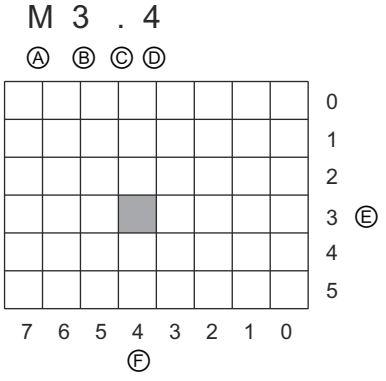
Note

The FB is called in the similar way as SBR.

5.2 accessing data

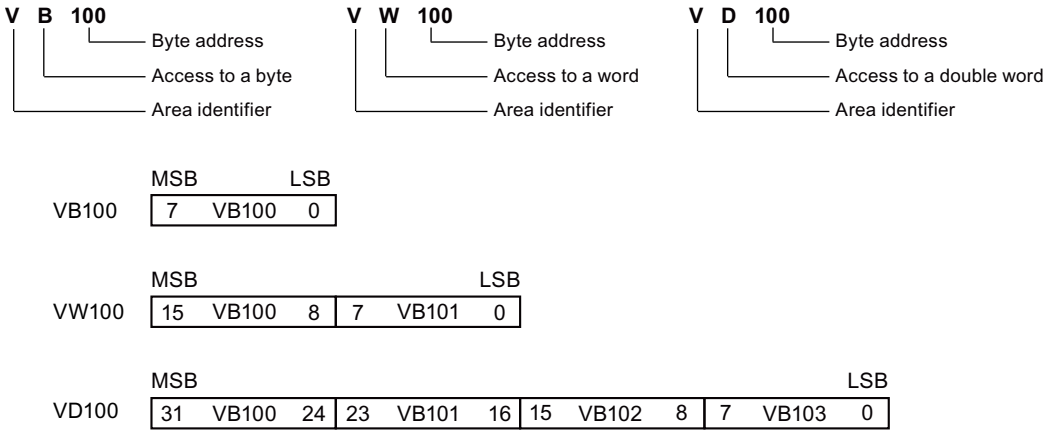
The CPU stores information in different memory locations that have unique addresses. You can explicitly identify the memory address that you want to access. This allows your program to have direct access to the information. To access a bit in a memory area, you specify the address, which includes the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

Table 5-2 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).

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. 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. 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 5-3 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)	*n.a.*	*n.a.*	+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.

5.2.1 Accessing memory areas

I (process-image input)

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

Table 5-4 Absolute addressing of 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 5-5 Absolute addressing of 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 5-6 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 5-7 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 5-8 Absolute addressing of 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. The timer has two variables:

- 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 5-9 Absolute addressing for T memory

Timer:	T*[timer number]*	T24

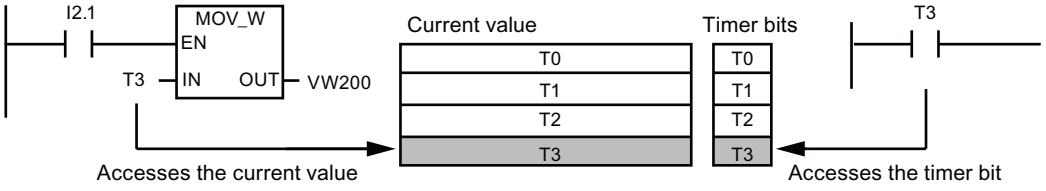


Figure 5-2 Accessing the timer bit or the current value of a timer

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 5-10 Absolute addressing of C memory

Counter	C*[counter number]*	C24

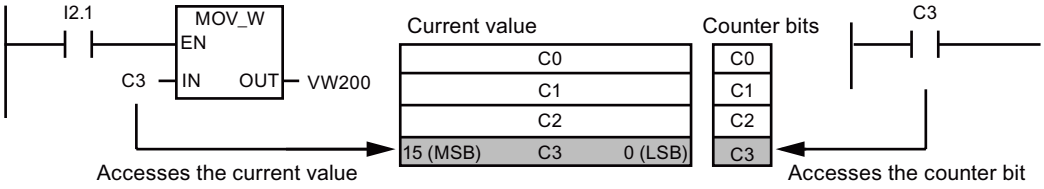


Figure 5-3 Accessing the counter bit or the current value of a counter

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 5-11 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 398).

Table 5-12 Absolute addressing of AC memory

Accumulators	AC*[accumulator number]*	AC0

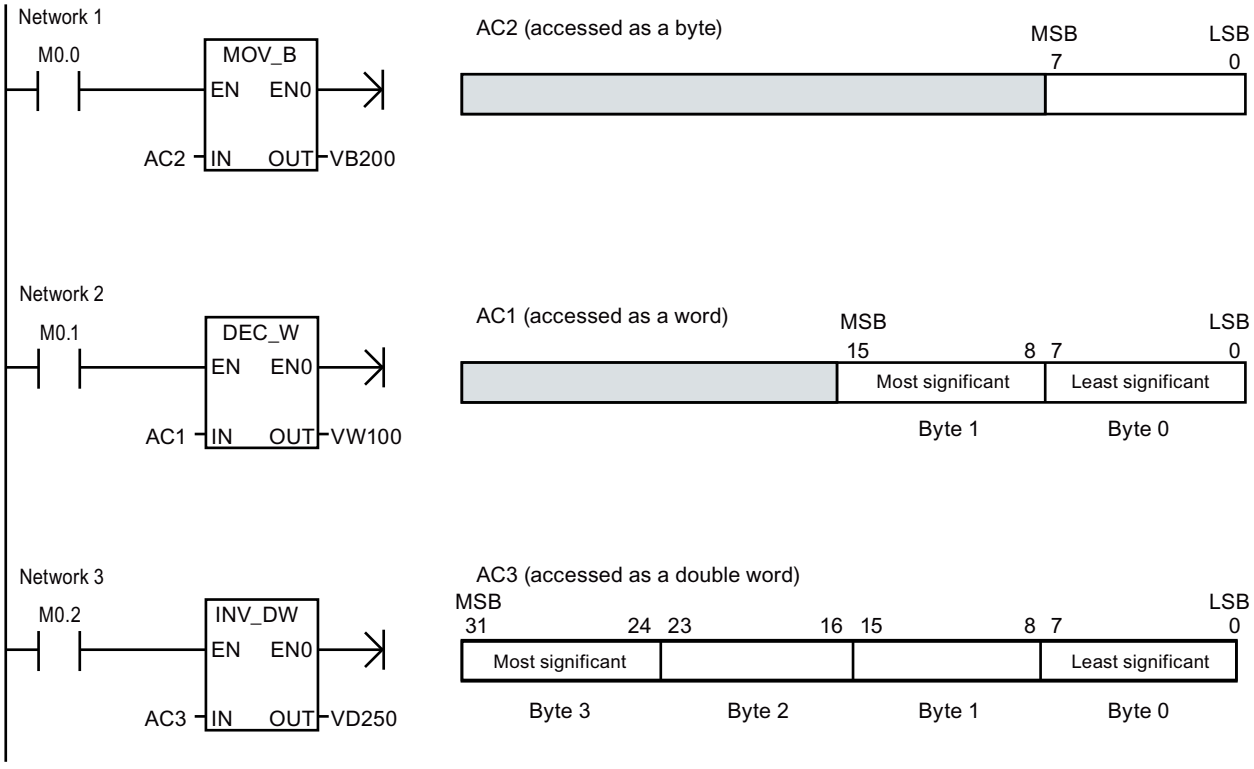


Figure 5-4 Accessing the accumulators

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 5-13 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 1191).

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 5-14 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 5-15 Absolute addressing of AI memory

Analog inputs	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 (AI), 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 5-16 Absolute addressing of AQ memory

Analog outputs	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 5-17 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

5.2.2 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.

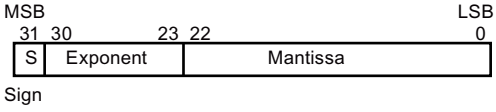


Figure 5-5 Format for Real numbers

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

5.2.3 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.

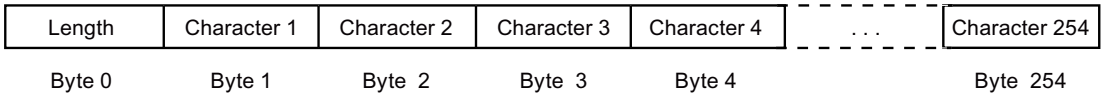


Figure 5-6 Format for Strings

5.2.4 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 5-18 Representation of Constant Values

Representation	Format	Example
dec	[decimal value]	20047
Hexadecimal	16#[hexadecimal value]	16#4E4F
bin	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.

5.2.5 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 5-19 CPU mapping convention

	CPU	Signal board 0	Signal board 1	Expansion module 0	Expansion module 1	Expansion module 2	Expansion module 3	Expansion module 4	Expansion module 5	Expansion module 6
Starting address	I0.0 Q0.0	I7.0 Q7.0 AIW12 AQW12	I72.0 Q72.0 AIW528 AQW528	I8.0 Q8.0 AIW16 AQW16	I12.0 Q12.0 AIW32 AQW32	I16.0 Q16.0 AIW48 AQW48	I20.0 Q20.0 AIW64 AQW64	I24.0 Q24.0 AIW80 AQW80	I28.0 Q28.0 AIW96 AQW96	I32.0 QI32.0 AIW112 AQW112

	Expansion module 7	Expansion module 8	Expansion module 9	Expansion module 10	Expansion module 11	Expansion module 12	Expansion module 13	Expansion module 14	Expansion module 15
Starting address	I36.0 Q36.0 AIW128 AQW128	I40.0 Q40.0 AIW144 AQW144	I44.0 Q44.0 AIW160 AQW160	I48.0 Q48.0 AIW176 AQW176	I52.0 Q52.0 AIW192 AQW192	I56.0 Q56.0 AIW208 AQW208	I60.0 Q60.0 AIW224 AQW224	I64.0 Q64.0 AIW240 AQW240	I68.0 Q68.0 AIW256 AQW256

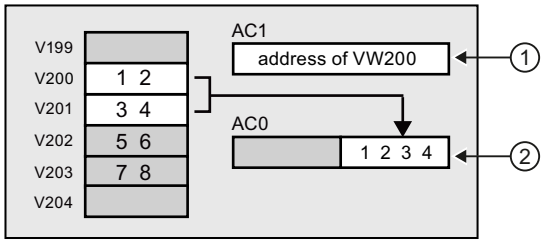
5.2.6 Using Pointers for indirect addressing

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

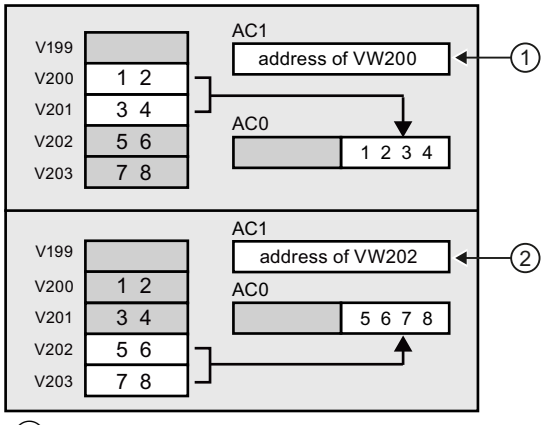
Creates the pointer by moving the address of VB200 (initial byte of VW200) to AC1

② MOVW *AC1, AC0

Moves the word value referenced by the pointer in AC1

Figure 5-7 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 5-8 Modifying a Pointer

Note

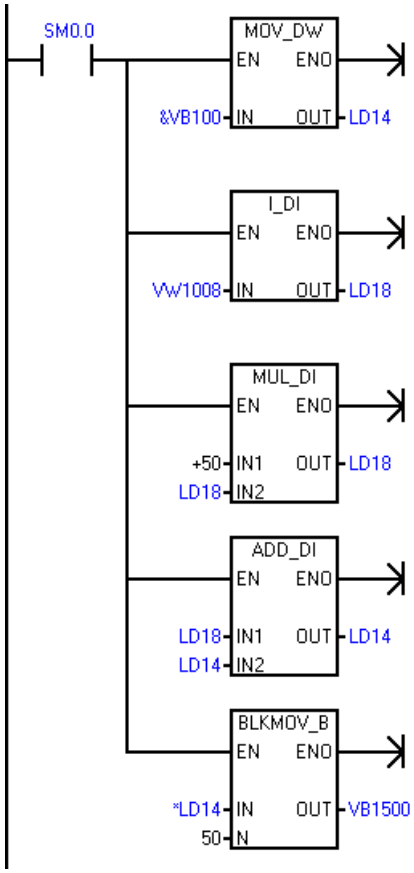
When modifying the value of a pointer, remember to adjust for the size of the data that you are accessing: to access a byte, increment the pointer value by 1; to access a word or a current value for a timer or counter, add or increment the pointer value by 2; and to access a double word, add or increment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

5.2.7 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 5-20 Example: Using a pointer to access data in a table

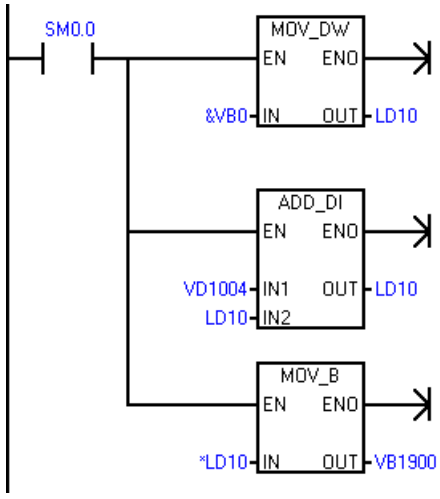
LAD		STL
	To transfer a recipe from a table of recipes: • Each recipe is 50 bytes long. • The index parameter (VW1008) identifies the recipe to be loaded. Create a pointer to the starting address of the recipe table. Convert the index of the recipe to a double-word value. Multiply the offset to accommodate the size of each recipe. Add the adjusted offset to the pointer. Transfer the selected recipe to VB1500 to 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 5-21 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`

5.3 Saving and restoring data

5.3.1 Downloading project components

Note

Compatibility between the project version and the CPU version during download

- Allows the ST32 (V3.1) project to be downloaded into the ST32 (V3.2) firmware, and then runs the behavior of V3.2.
- Downloading ST32 (V3.2) projects into ST32 (V3.1) firmware is not allowed.
- Projects from ST40/60 and SR40/60 must not be downloaded to the CPU of G2; nor can projects from G2 be downloaded to the CPU of ST40/60 and SR40/60.

Note

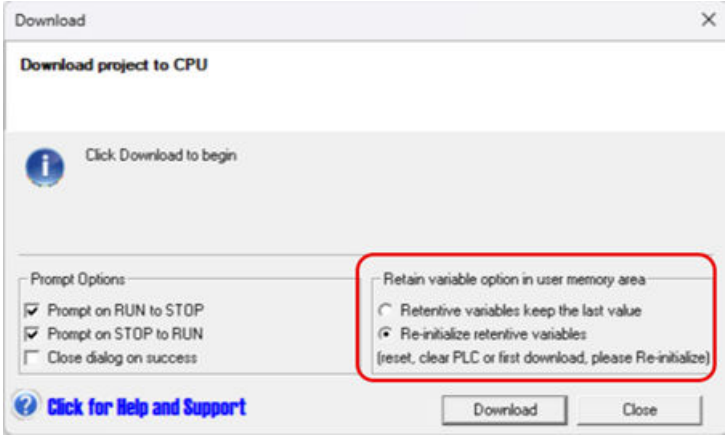
Downloading a program to the CPU completely overwrites any pre-existing program in the CPU. Be sure that you want to overwrite the program before performing a download.

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 73) or RS485 (Page 532) 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 81).
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 250)) 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 81) and STOP to RUN mode (Page 81).
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

Programs created with STEP 7 Micro/WIN V3 can only be downloaded to S7-200 SMART CPUs with version 3.0 or higher.

Programs created with STEP 7 Micro/WIN (Version 2.8) or earlier can only be downloaded to S7-200 SMART CPUs V2.8 or earlier.

You can download a maximum of 4 MB of project information to an S7-200 SMART V3.0 and G2 CPU, 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 (Page 181).

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	Operations	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 1187) and non-fatal run-time errors (Page 1184) 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 (Page 129)

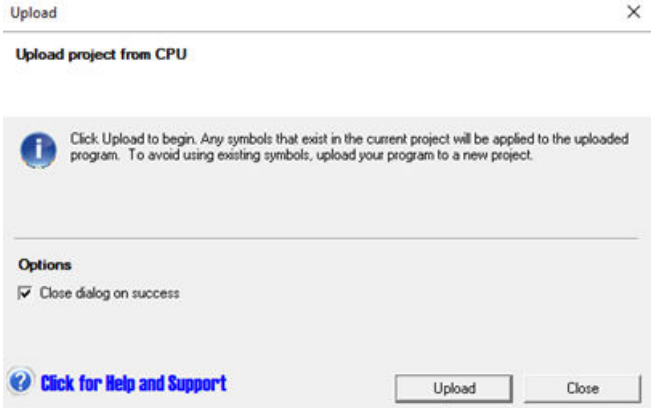
5.3.2 Uploading project components

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 742).
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 126) 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.

Note

If you download the V3.1 project to the V3.2 firmware, the motion control instructions and TO will be automatically updated to version V3.2 after uploading.

5.3.3 Type of energy 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 133)
  - To completely erase the PLC as a restore-to-factory-defaults card (Page 235)
  - To update the PLC and expansion module firmware as a firmware update card (Page 131)

5.3.4 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 133)
- Reset CPU to factory default condition (Page 235)
- 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 5-22 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 235) purposes, the memory card is organized as follows:

Table 5-23 Memory card used to reset to factory defaults

At the root level 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 5-24 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.

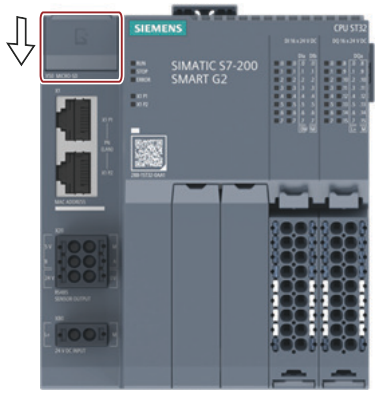
Note

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.

5.3.5 Inserting a memory card in a standard CPU

Table 5-25 Inserting and removing a memory card in a standard CPU

Task	Step
	Follow the steps below to insert the microSDHC memory card into the CPU. 1. Pull down the terminal block connector cover on the upper left side. 2. Insert the microSDHC memory card in the memory card slot (marked X50) located above the terminal block connectors. 3. Slide back 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. Pull down the terminal block connector cover on the upper left side. 2. Grasp the microSDHC memory card from the CPU and pull it out of the card slot (marked Micro-SD X50). 3. Slide back the terminal block connector cover

5.3.6 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 138) 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:

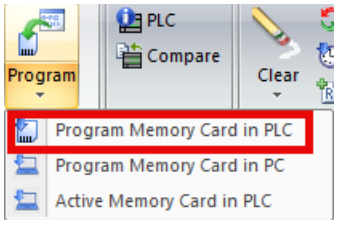
1. Ensure that your network hardware and PLC connector cable are working, the CPU is powered on and in STOP mode, and that PLC communication is operating properly (Page 73).
2. If not already inserted, insert a microSDHC memory card in the CPU. You can insert or remove the memory card while the CPU is powered on.
3. Download (Page 81) the program to the PLC, if not already downloaded.

   Note

   Compatibility between the project version and the CPU version during download

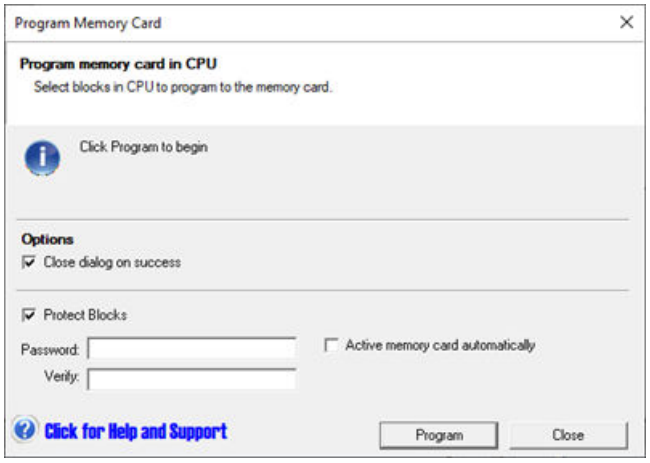
   - Allows the ST32 (V3.1) project to be downloaded into the ST32 (V3.2) firmware, and then runs the behavior of V3.2.
   - Downloading ST32 (V3.2) projects into ST32 (V3.1) firmware is not allowed.
   - Projects from ST40/60 and SR40/60 must not be downloaded to the CPU of G2; nor can projects from G2 be downloaded to the CPU of ST40/60 and SR40/60.

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.

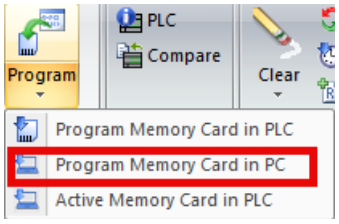
6. Click the Setting button.
7. Enter the password (Page 56) if one is required for programming the memory card.



Creating a program transfer memory card on the PC

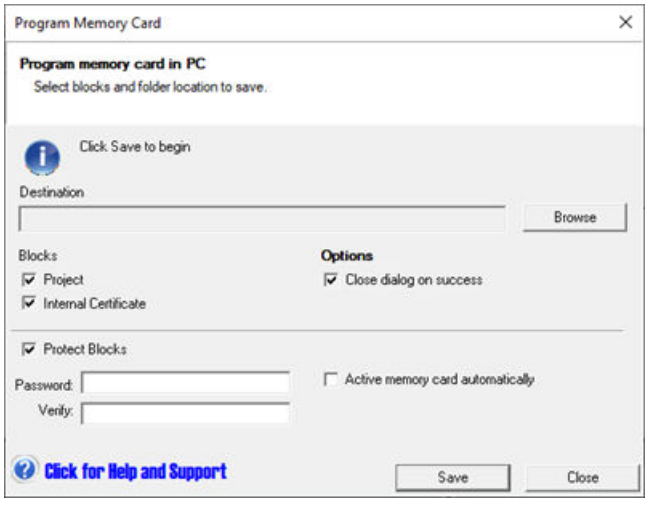
To save the program on the computer, follow the steps:

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.

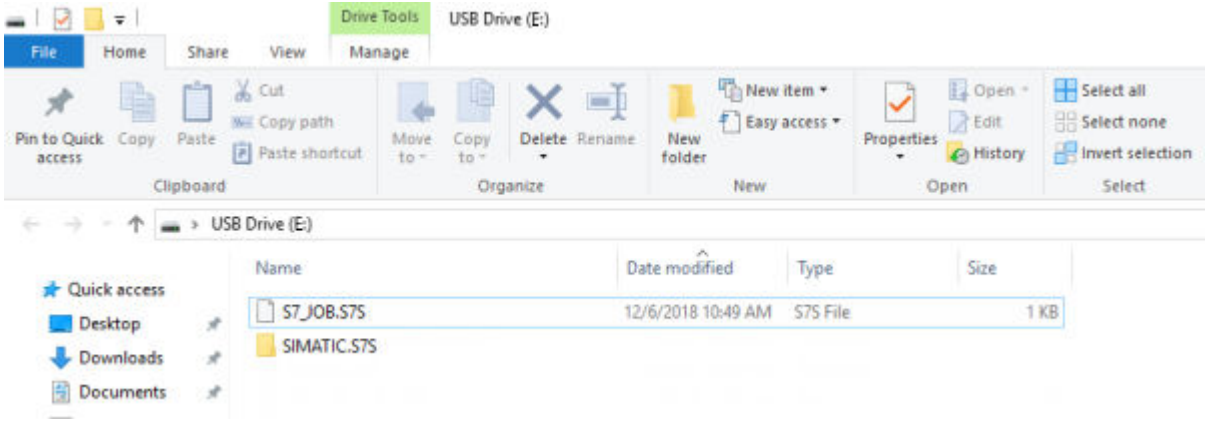


4. Select "Protect Data" if you want to set a password for activating the memory card.

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 205)
- 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 the Web server is disabled, the internal certificate will be grayed out and cannot be stored.

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 235).

5.3.7 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

5.4 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.

place 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 432) in your program logic to put the CPU in STOP mode.

5.5 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:

Status	LED status	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

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

Table 5-26 LEDs of the analog module (EM)

Description	DIAG (Red / Green)	I/O （Channel） (Green)
Field-side power is off	Flashing red	Off
No configuration, update in progress, or broken wire detection	Flashing green	Off
Module configured with no errors	On (green)	On (green)
Error condition	Flashing red	-
I/O error (with diagnostics enabled)	-	Off
I/O error (with diagnostics disabled)	-	On (green)

Table 5-27 LEDs of the digital module (EM)

Description	DIAG (Red / Green)	I/O （Channel） (Green)
Not configured, update in progress	Flashing green	• Logical 1: On • Logical 0: Off
Module configured with no errors	On (green)	
Error condition	Flashing red	

Table 5-28 LEDs for thermocouple and RTD modules (EM)

Description	DIAG (Red / Green)	I/O （Channel） (Green)
No configuration, update in progress, or broken wire detection	Flashing green	Off
Module configured with no errors	On (green)	On (green)
Error condition	Flashing red	-
I/O error (with diagnostics enabled)	-	Off
I/O error (with diagnostics disabled)	-	On (green)

Programming concepts

6

6.1 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. If control devices fail, they can create unsafe conditions, 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.

6.2 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 very useful when you want to repeatedly execute a certain function. 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 execution is completed. By designing a general control task FB, the FB can be reused for multiple devices by: Selecting different FB instances for different calls of the FB. An FB stores inputs, outputs, input/output parameters, 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.

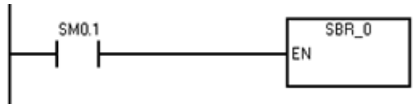
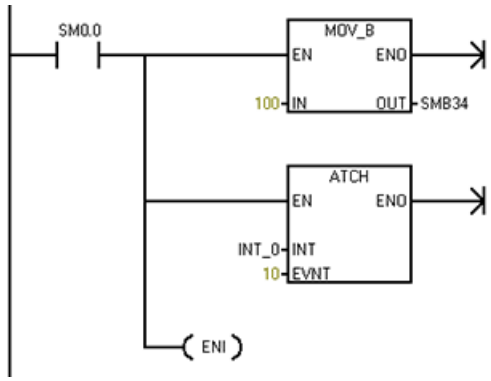
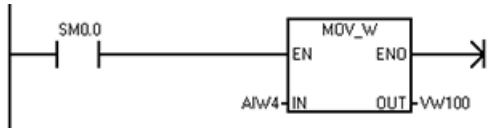
  Use the local variable table of the interrupt routine to ensure that your interrupt routine uses only the temporary memory and does not overwrite data used somewhere else in your program.

  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 398).

- Other blocks contain information for the CPU.
  - System blocks: The system block allows you to configure different hardware options for the CPU.
  - Data blocks: 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 6-1 Sample program with a subroutine and an interrupt routine

Main program		**Network 1** **LD SM0.1** **CALL SBR_0**	On the 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.

6.3 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.

6.3.1 Earlier versions of STEP 7-Micro/WIN projects

As a new generation product, S7-200 SMART G2 has been fully upgraded at the hardware level, including CPU, expansion modules and signal boards. Among them, the S7-200 SMART G2 CPU has significantly improved its expansion capability and can now support up to 16 expansion modules.

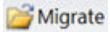
Because the S7-200 SMART G2 CPU model and the number of I/O points it supports are different from previous versions, when you migrate an existing project to the S7200 SMART G2 CPU, you must replace the expansion modules and signal boards configured in the system block to ensure compatibility with the S7200 SMART G2.

STEP 7-Micro/WIN SMART V3 (Version 3.2) allows you to migrate your STEP 7-Micro/WIN SMART (Version 2.8) or STEP 7-Micro/WIN SMART V3 (Version 3.0) projects and configurations to the S7-200 SMART G2 series. During the migration process, the CPU, signal board (SB), and expansion module (EM) used in the original version project will be automatically replaced with the CPU, signal board (SB), and expansion module (EM) corresponding to the S7-200 SMART G2. If the number of I/O points is found to be insufficient after migration, the system will automatically add additional S7-200 SMART G2 expansion modules to make up the required I/O capacity.

After migration, STEP 7-Micro/WIN SMART V3 can accurately read and display information such as PLC data, alarms, and errors of its modules. To ensure compatibility, the migrated project files can only be opened with STEP 7-Micro/WIN SMART V3 (version 3.1) and above.

To use a project created in STEP 7-Micro/WIN SMART (Version 2.8) or STEP 7-Micro/WIN SMART V3 (Version 3.0) in STEP 7-Micro/WIN SMART V3 (Version 3.2), 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.

After the migration is complete, the CPU version will be updated to S7-200 SMART G2.

Note

Only projects created with STEP 7-Micro/WIN SMART (Version 2.8) or STEP 7-Micro/WIN SMART V3 (Version 3.0) can be migrated to Version 3.2. For projects created in versions prior to STEP 7-Micro/WIN SMART (Version 2.8), they need to be opened and saved in Version 2.8 before they can be migrated to STEP 7-Micro/WIN SMART V3 (Version 3.2).

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 STEP 7-Micro/WIN SMART V3 (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.

POU

Name

- If the SBR/INT names in the migrated project conflict with the newly added system POU names in STEP 7-Micro/WIN SMART V3, 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.

Instructions

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 381)
- Multiply integers to double integer and divide integer with remainder (Page 385)
- AND, OR, XOR, NAND, NOR, and NXOR (Page 414)
- Shift and rotate (Page 435)

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 Version 2.8		STEP 7-Micro/WIN SMART V3	
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 setting • 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 (Version 2.8) to STEP 7-Micro/WIN SMART V3 (Version 3.2), 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 STEP 7-Micro/WIN SMART V3. 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 (Version 2.8)	Global variable table in STEP 7-Micro/WIN SMART V3	
symbol name	Variable names	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 projects from Version 2.8 to STEP 7-Micro/WIN SMART V3 and modify it either with motion wizard or the new motion control. For more information, refer to "Motion control(PLCopen) (Page 765)".
- **HSC**: The project with HSC created in STEP 7-Micro/WIN SMART (Version 2.8) can be opened and used directly in STEP 7-Micro/WIN SMART V3 as before.

- **PROFINET**
  - After the migration, the PROFINET network structure will be converted to a hardware network topology.
  - 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 S7-200 SMART G2.
- **Web server**

Note

Web server users are lost after migration

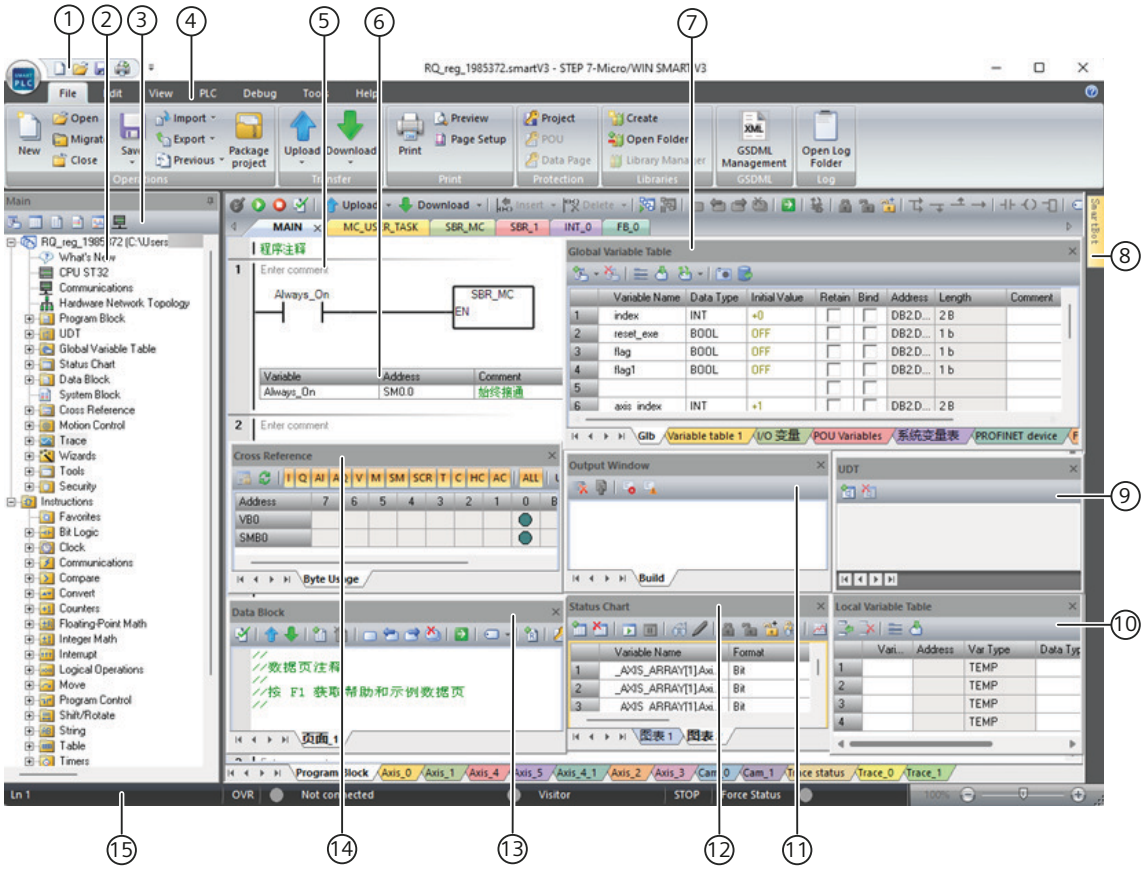
When a project in Micro/WIN SMART (Version 2.8) configured web server users, after migration, the users are lost.

6.3.2 Using STEP 7-Micro/WIN SMART user interface

STEP 7-Micro/WIN SMART V3 is the configuration, programming and operating software for S7-200 SMART CPUs with firmware version V3.0 (ST/SR40, ST/SR60) and V3.1/V3.2 (ST32). One instance of STEP 7-Micro/WIN SMART V3 can connect to one S7-200 SMART CPU at a time.

Screen areas and elements

The STEP 7-Micro/WIN SMART V3 user interface provides many windows for organizing, programming, and monitoring. Note that you can dock each window or float and arrange windows on the screen as you choose. You can display each window separately, or you can combine windows such that each window is accessible from a separate tab.

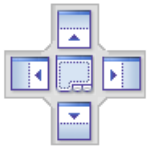


① Quick access toolbar (Page 152)
② Project tree (Page 152)
③ Navigation bar (Page 152)
④ Menus (Page 152)
⑤ Program editor (Page 152)
⑥ Variable information table
⑦ Global variable table (Page 159)
⑧ SmartBot
⑨ UDT (Page 169)
⑩ Local variable table (Page 163)
⑪ Output window (Page 152)
⑫ Status chart (Page 737)
⑬ Data block (Page 157)
⑭ Cross reference (Page 731)
⑮ Status bar (Page 152)

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".

Moving windows

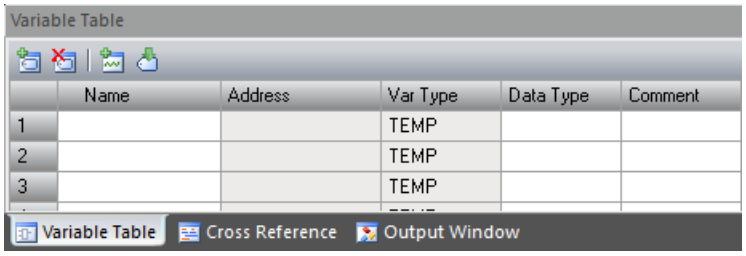


To move a window, click on the frame of a window and begin dragging it. As you move, you see a locator symbol that helps you dock a window to the left, right, top or bottom of the current window. You also have left, right, top, and bottom locator symbols within the STEP 7-Micro/WIN SMART V3 user interface. You can use these symbols to dock a window on one of the four sides of STEP 7-Micro/WIN SMART V3. As you move the window over a locator symbol, STEP 7-Micro/WIN SMART V3 shades the area where it will position the window if you drop it at that location. You can also drop the window alone at any floating location.

Displaying multiple windows on tabs

To display multiple windows as tabs of a single window, drag a window on top of another window. STEP 7-Micro/WIN SMART V3 then combines the two windows with a tab for each. The following example shows tabs for a combination of the following windows:

- Variable Table
- Cross reference table
- Output window



Note

STEP 7-Micro/WIN SMART V3 remembers your layout from your previous session and restores the windows you had opened in their last positions.

6.3.3 Using STEP 7-Micro/WIN SMART 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.

6.3.4 Using wizards to help you create your control program

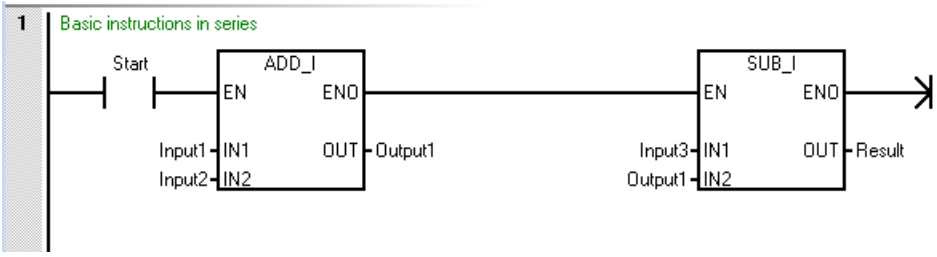
STEP 7-Micro/WIN SMART V3 provides the following wizards to make aspects of your programming easier and more automatic:

- High-speed counter
- Motion (V2.x)
- PID
- PWM (Pulse Width Modulation)
- Get/Put
- Data Log (standard CPUs only)
- 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.

6.3.5 Ladder logic 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.

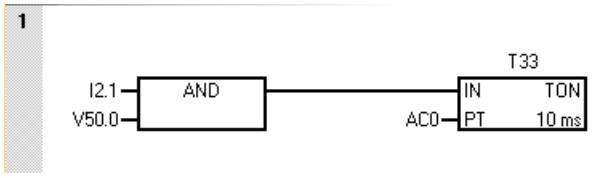
Points to consider

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 155) or STL editor (Page 155) to display a program created with the LAD editor.

6.3.6 Function block diagram 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 154) or STL editor (Page 155) to display a program created with the FBD editor.
- The expandable AND/OR boxes make it easier to draw complex input combinations.

6.3.7 Statement list 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

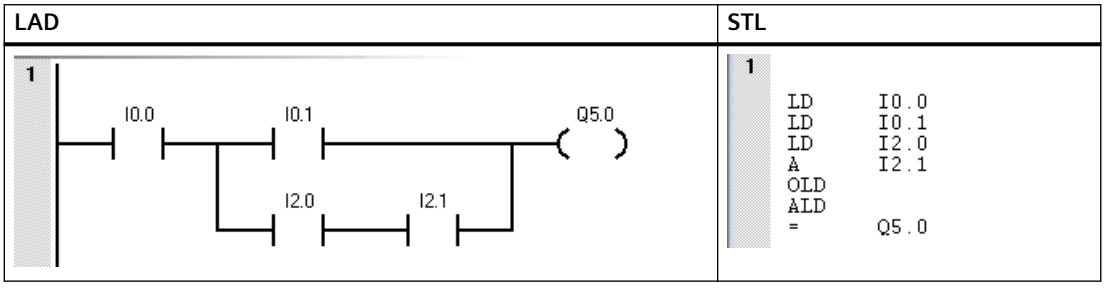
```
1

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 6-2 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 154) or FBD editor (Page 155), 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.

6.4 Data block (DB) editor

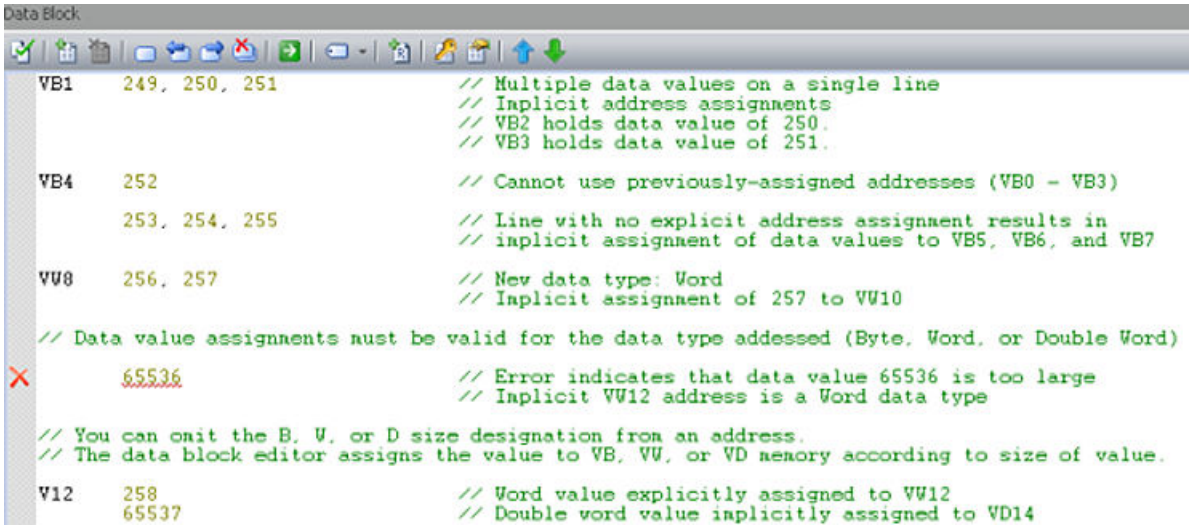
The data block allows you to assign constants (Page 122) (either numeric values or character strings) to specific locations of V memory. You can make value assignments to byte (V or VB), word (VW), or double word (VD) addresses. You can also enter optional comments, preceded by // double forward slashes.

- 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 159) 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

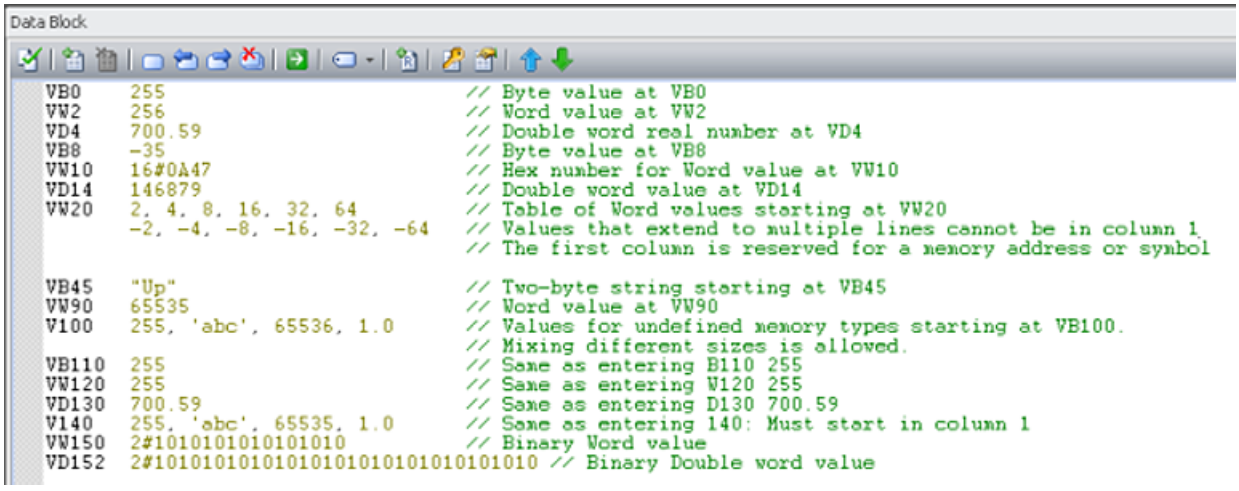
If variables with data types are used in the DB, it must be ensured that the values in the DB are consistent with the data types. The DB cannot use complex structure variables.

Example: Data block page



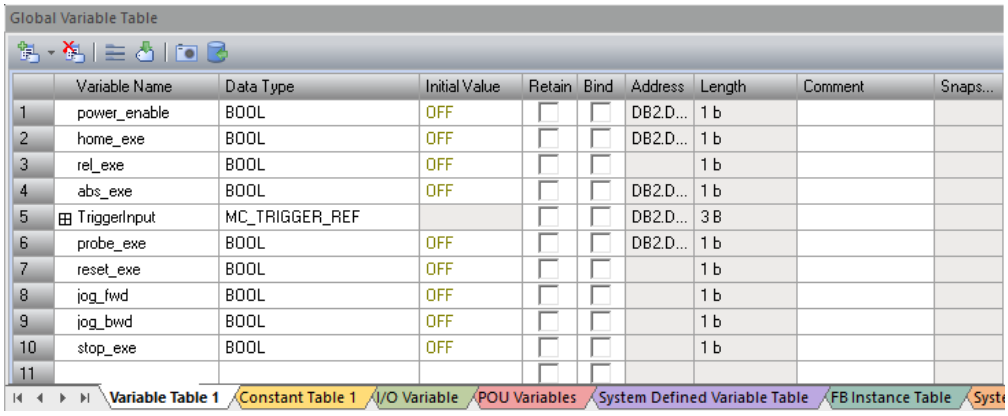
Note: Enter a space before the data values on the lines where you enter no explicit address.

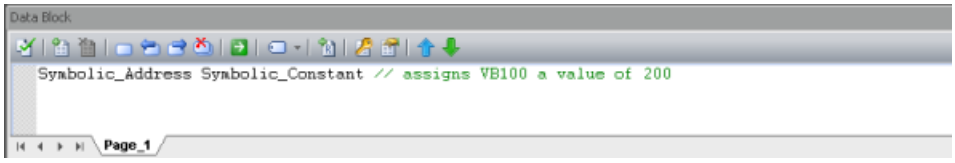
Example: Direct address and number values



Example: Global variable address and global variable assignment

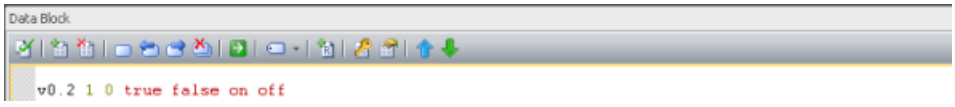
Complex structure variables cannot be used in DB.

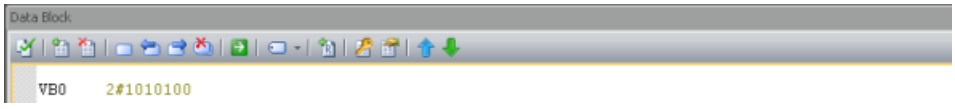




Example: Alternate binary entry methods and resultant binary assignment

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.





6.5 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 apply globally. You can use your defined global variables in all of the Program Organizational Units (Page 143) (POUs) of your program. If you make a variable name assignment in a local variable table (Page 163), 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.
Variables cannot start with a number.
"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:

- First character requirement: Variable names must begin with a letter or extended character (ASCII 128 to ASCII 255) and cannot start with a digit.
- Allowed characters: Variable names can contain letters, digits, underscores (_), and extended characters (ASCII 128–255), but special symbols such as spaces, colons, quotation marks, and other illegal characters are not allowed.
- Total length limit:
  - The maximum length of global and local variables is 128 characters.
  - Local variable names support a maximum of 23 alphanumeric characters and underscores.
- Use of keywords is prohibited: Reserved words or instruction names from the programming language cannot be used as variable names. For example, "Begin" is a reserved word and is an invalid variable name.
- Cannot use "Aa+Number", "aa+Number", "aA+Number", or "AA+ number format". Variable names in these formats may be incorrectly identified by the system as AQ addresses.
- Uniqueness requirement: In the same project, the same variable name cannot be reused for global assignment, as it will result in a compilation error. If duplicate names occur, the system will automatically add a "_number" suffix to differentiate them.

In addition, there are additional limitations in certain application scenarios:

- Axis names are allowed a maximum of 23 alphanumeric characters and underscores, and the first character can only be a letter or an underscore, and also cannot be the same as an existing instruction name or keyword.
- The name of the web server monitoring table should be less than 254 characters or 127 Chinese characters, and should not contain special 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 position.
- 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 position.
- 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.

6.6 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 159).

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 143) 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.

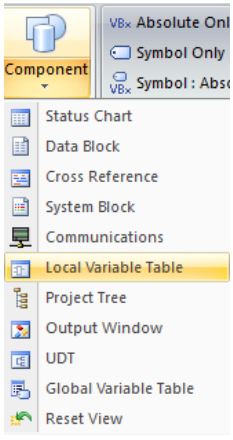
When main program calls SBR0, the value of INPUT1 is passed to FIRST.

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

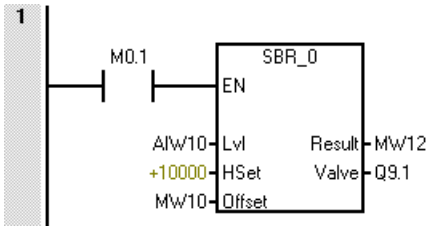
The collapse/expand button allows you to collapse or expand the Array/UDT type variable in the local variable table. By clicking , you can collapse or expand the Array/UDT variable to conveniently view its contents.

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		



6.7 User-defined data types

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:

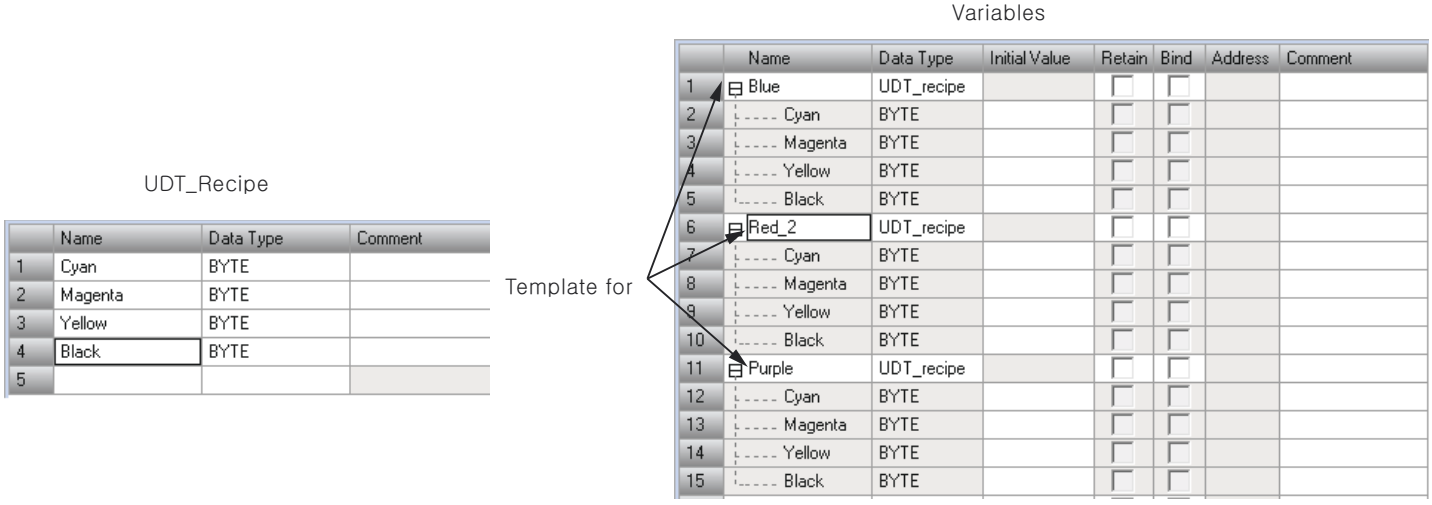
- Group data according to the process control
- Transferring a parameter as a data unit

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.

6.7.1 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 6-1 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			

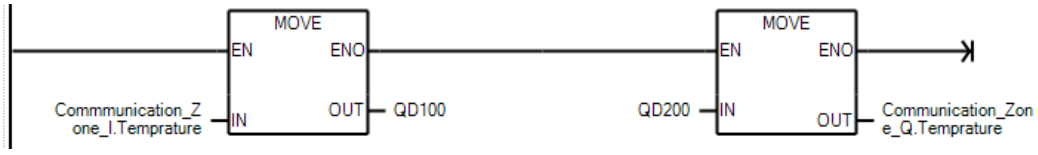
Figure 6-2 Q_Point

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.



6.8 PLC error reaction

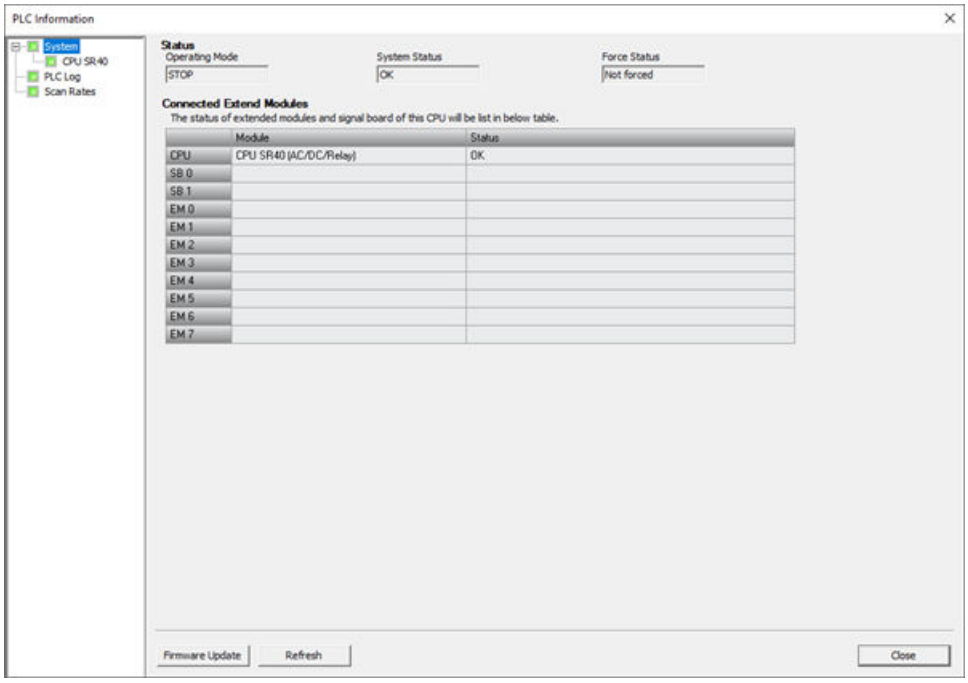
To view PLC information or perform a firmware update, click the PLC button in the Information area of the PLC menu ribbon strip.



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.

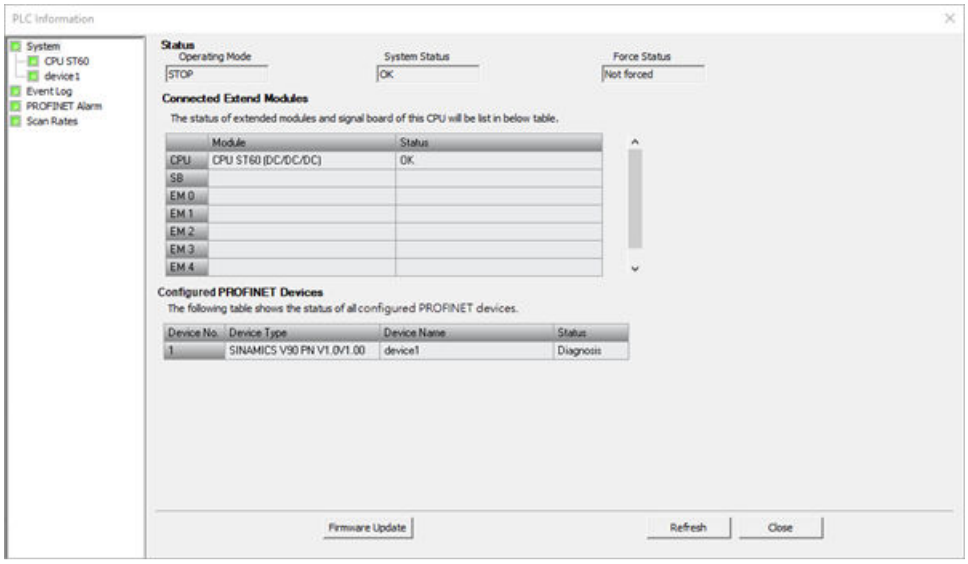
Note the following information:

- The PLC provides SM bits for programmed reactions to errors. Refer to the listing of the SM bits (Page 1191).
- 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 433) for details.

6.8.1 System information

6.8.1.1 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.



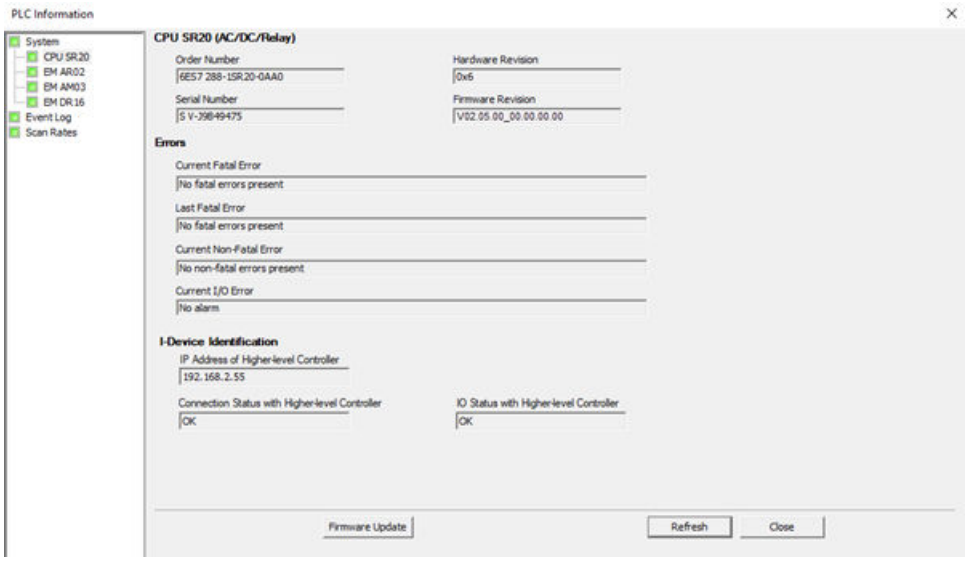
The "System" dialog displays the following information:

- 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.

6.8.1.2 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 1187).
  - 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

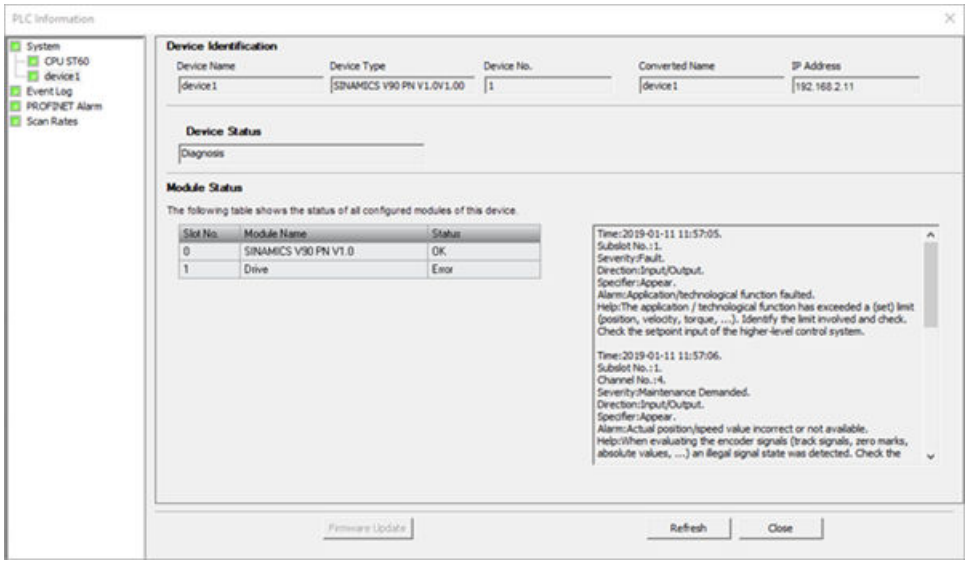
  If the CPU works as an I-Device, the CPU dialog displays the I-Device Identification information.

  - 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

6.8.1.3 PROFINET device

To view information for a specific device, click the appropriate device under "System".

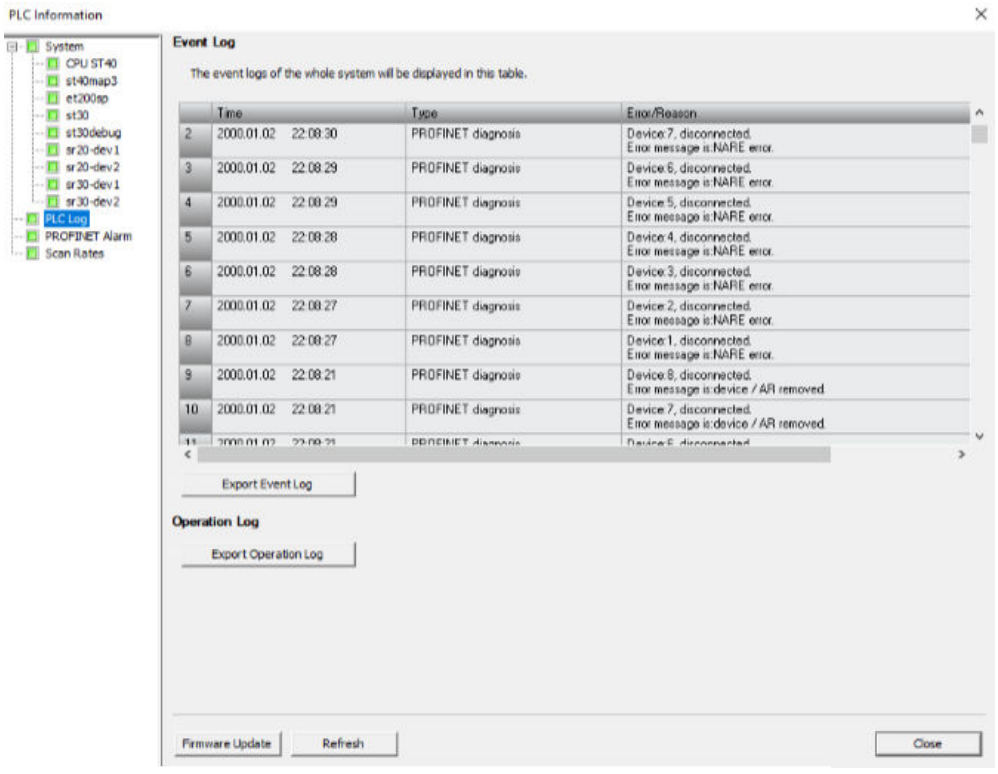
The PLC Information dialog shows you the device identification information, device status and module status as follows:



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.

6.8.2 Event 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.

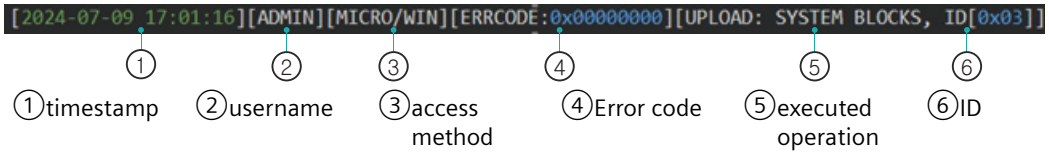


The maximum number of displayed event log is 128.

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.



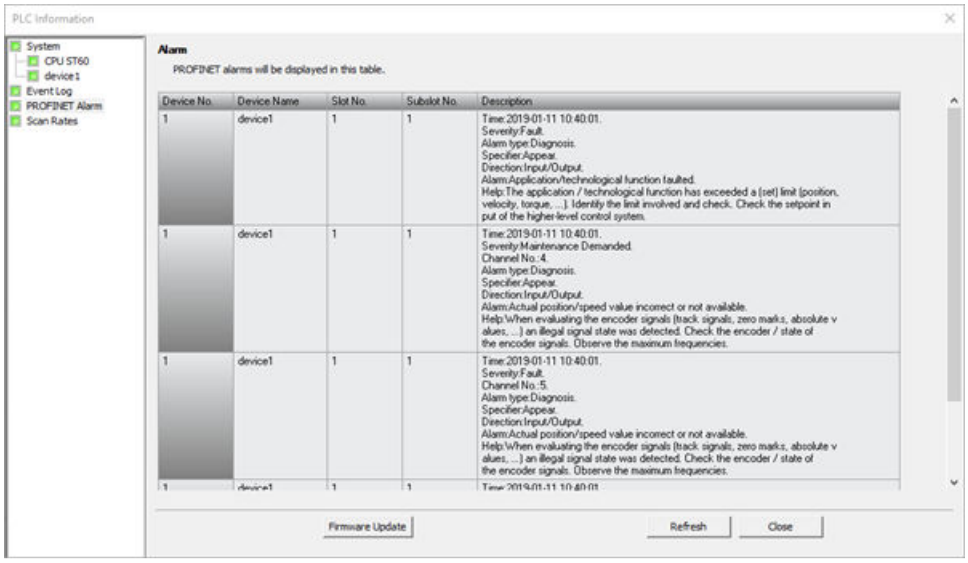
Note

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.

NOTICE
Export logs promptly To ensure effective troubleshooting and data integrity, be sure to export system logs promptly and regularly. This helps provide critical information when needed and supports your system maintenance efforts.

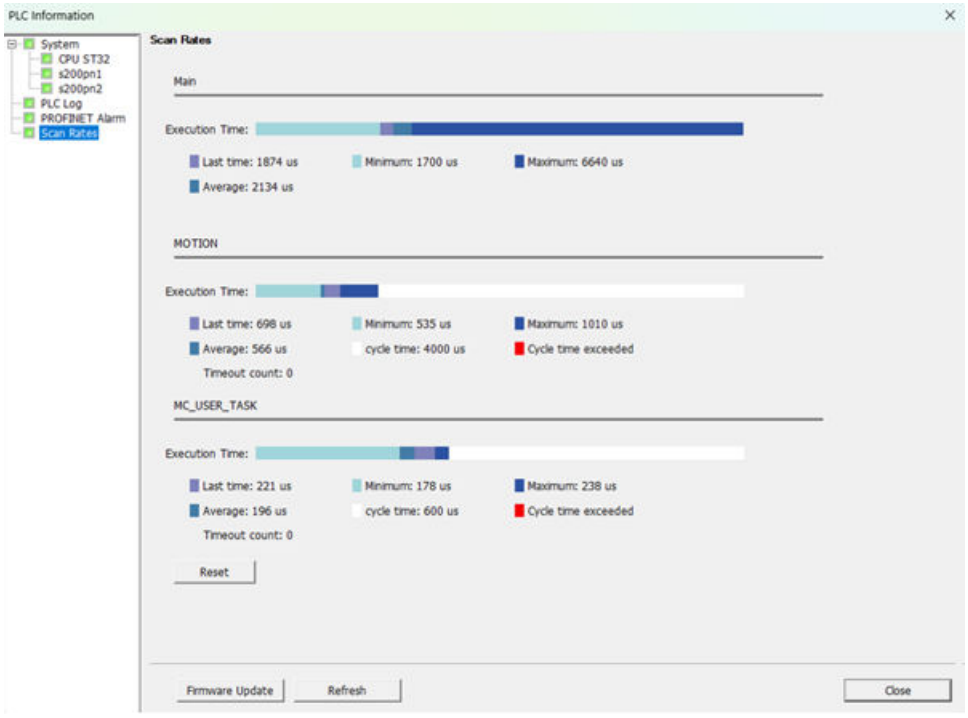
6.8.3 PROFINET Alarm

To view the PROFINET alarms for the PROFINET network, click "PROFINET Alarm". The PLC Information dialog displays a table of time-stamped alarms. The table lists the device number, device name, slot number, sub-slot number and alarm description.



6.8.4 Scan rates

To view scan rate statistics, click "Scan Rates".



"MAIN" displays the scan rate for the main program

"MOTION" displays the scan rate for the motion control

"MC_USER_TASK" displays the scan rate of the user task.

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.

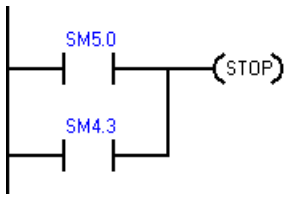
6.8.5 Non-fatal errors and I/O errors

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 6-3 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 6-4 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 433) to get the current non-fatal error code and reset SM4.3 to OFF.

Refer to the non-fatal error code list (Page 1184) for a description of compile rule violations and run-time programming problems.

Refer to the description of the SM bits (Page 1191) for more information about the SM bits used for reporting I/O and program execution errors.

6.8.6 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 1187) for details.

6.9 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.

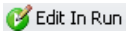
Handling positive or negative transitions

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 205), 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.

6.10 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 makes it easy to move back and forth between lines of a long program to specific lines
- Tracking references in your program with the cross reference table (Page 731)
- Using a status chart (Page 737) to display PLC data values and status
- Displaying status in the program editor (Page 733)

For more information about debugging your program, refer to the chapter on diagnostics and troubleshooting (Page 731).

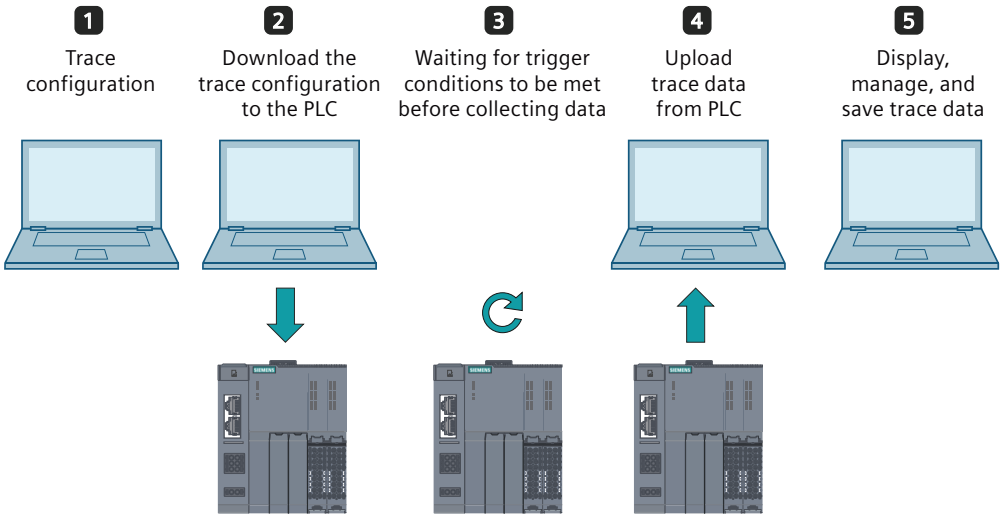
6.11 Trace

6.11.1 Introduction

Traces help you capture and track variable changes visually and precisely. By capturing, tracking, and comparatively analyzing this data, you can more effectively assess the program's operational status. Especially for transient data changes that are difficult to detect with the naked eye, the trace curve provides powerful monitoring and analysis capabilities. For example, in motion control systems, it can be used to monitor and analyze speed and position changes; during communication, it can be used to monitor and analyze data changes, completion bit or error bit status; and it can also be used to monitor and analyze rising or falling edge events in programs, among other application scenarios.

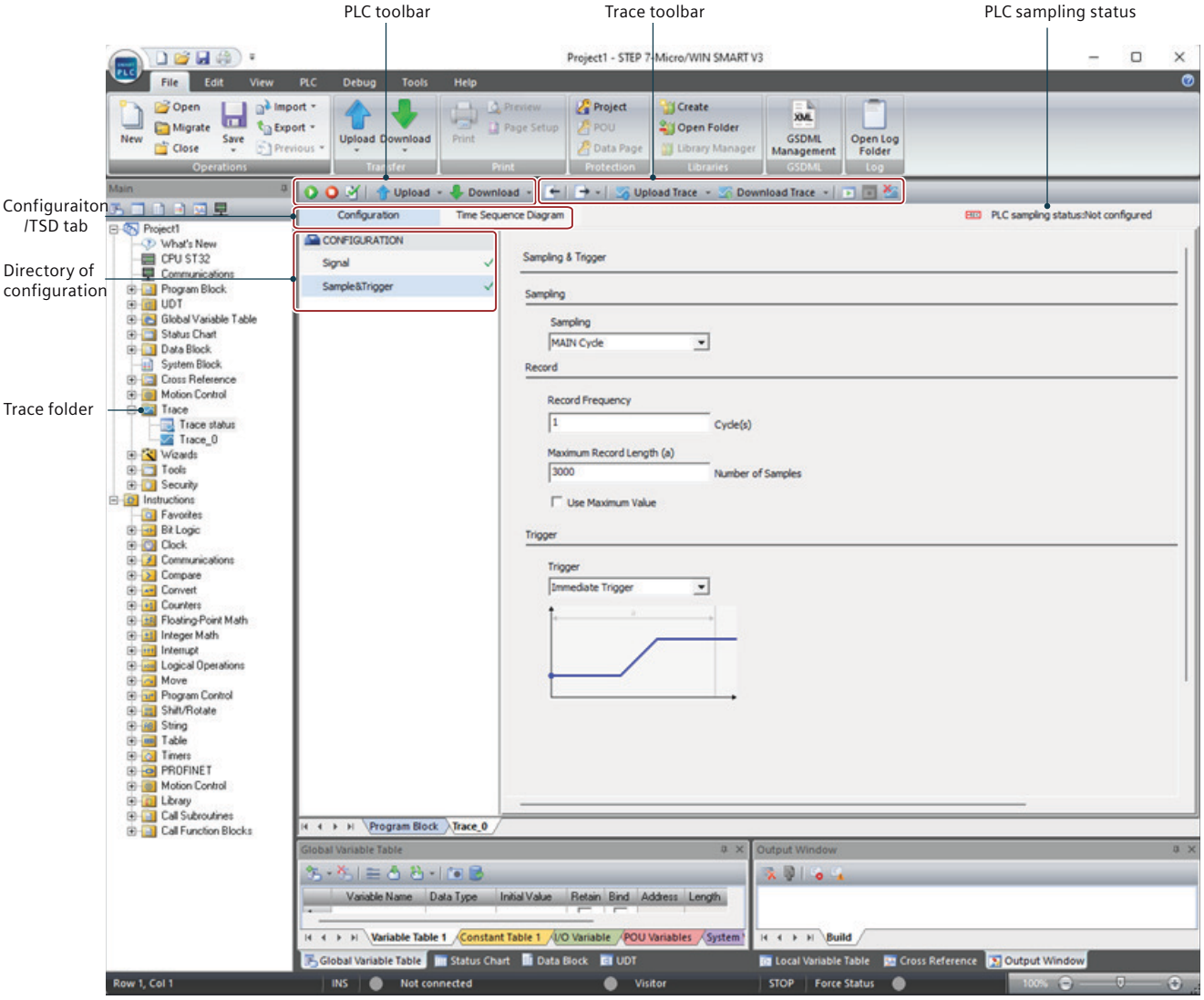
S7-200 SMART G2 supports loading 4 trace configurations, and each trace can be configured with a maximum of 32 signals.

Operating modes of the trace function



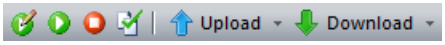
		Description
①	Configuring traces on the STEP7-Micro/WIN SMART V3 programming device	• Select the signal to acquire • Define sampling and trigger conditions
②	Downloading the trace configuration from the programming device to the PLC	When an online connection is established, the trace configuration can be downloaded to the PLC in the form of a task.
③	Wait for the trigger condition to be met before acquiring data	When trace configuration is activated, data acquisition takes place independently of the programming device. When the trigger condition is met, acquisition starts immediately. The acquired data in the PLC are not retentive, but are permanently saved in the project. A maximum of 128 traces can be saved in a STEP7-Micro/WIN SMART V3 project.
④	Reading trace data from PLC	The upload function can be used to upload the traces in the PLC to STEP7-Micro/WIN SMART V3. A trace task includes trace configuration and data acquisition. Each acquired trace task can be uploaded and saved in the project. Traces can be viewed offline.
⑤	Display, manage and save trace data	The acquired data in the curve diagram and signal table can be evaluated in various ways. The trace data can be exported as a CSV file.

6.11.2 User interface



PLC toolbar

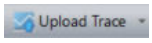
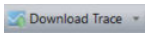
The PLC toolbar in the trace workspace provides a subset of the functions available from the PLC menu.



Trace toolbar

The trace toolbar in the trace workspace provides trace transfer and control functions.



	Importing a trace (Page 200)	Import trace configuration from a file. This operation will overwrite the current existing configuration.
	Exporting a trace (Page 200) Exporting data (Page 200)	Export the trace configuration with the file extension "*.xml"; Export trace data with the file extension ".csv"
	Upload trace (Page 201)	Uploads the trace of the specified task (i.e. configuration or configuration containing data) from the PLC and overwrites the current configuration.
	Download trace (Page 201)	Download the trace configuration to the PLC in the form of a task.
	Activating a trace (Page 202)	Activate the trace task on the PLC.
	Deactivating a trace (Page 202)	Stop the trace task on the PLC.
	Delete trace (Page 204)	Delete the trace on the PLC.

PLC sampling status

	The trace has not been configured.
	The trace task has been configured but not yet activated.
	The trace task has been configured and activated, but the triggering conditions have not yet been met.
	The trace task has been activated, the trigger conditions have been met, and the data acquisition has started.
	Trace data acquisition stopped: • The maximum number of acquisition entries has been reached. • Stopped due to CPU STOP. • Deactivating.

You can also view the PLC sampling status in the Trace special memory area (Page 1223).

6.11.3 Trace timing diagram interface

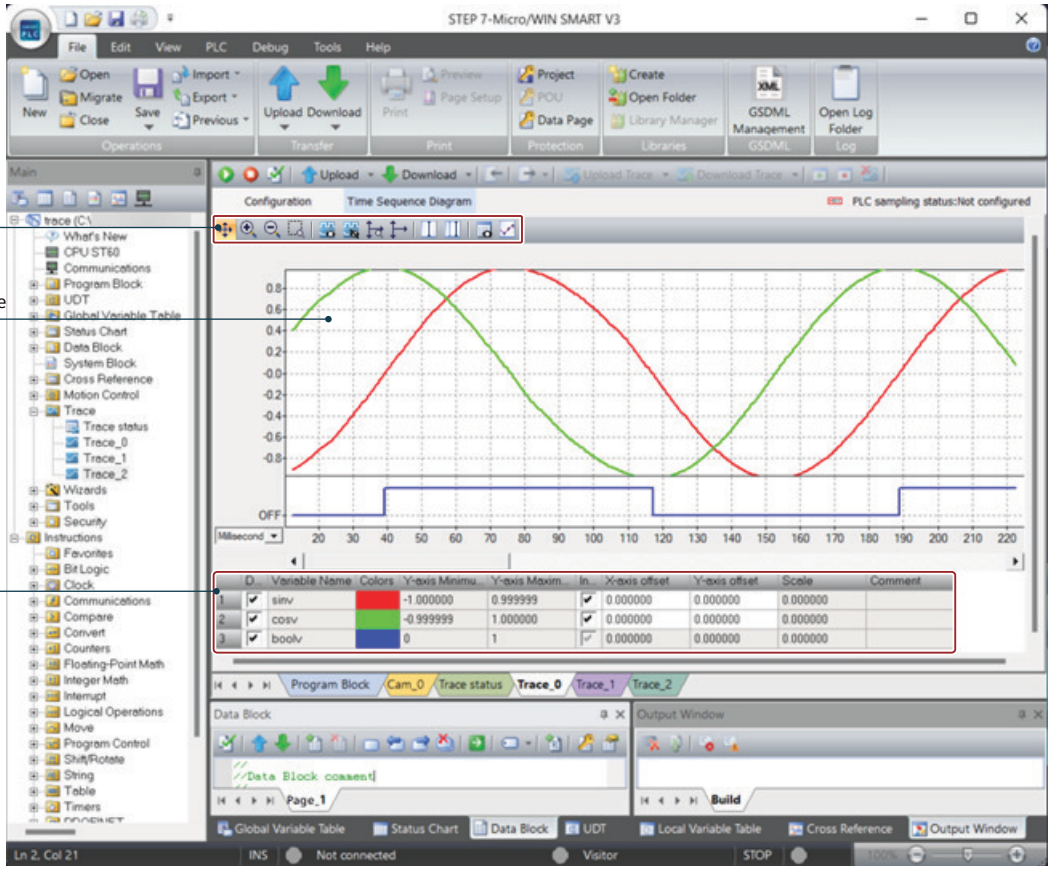


Chart toolbar

You can use the tools in the toolbar to view the trace curve.



	Move view	In this mode, you can move the timing diagram by dragging the mouse.
	Zoom in	Zoom in on the center of the canvas in 10% increments. Each time the button is pressed, the range of the time and value axes is reduced, making the curve display larger.
	Zoom Out	Zoom out from the center of the canvas in 10% increments. Each time the button is pressed, the range of the time and value axes is increased, making the curve display smaller.
	Zoom to selection	On the timing diagram, press and hold the left mouse button, then drag from the top-left to the bottom-right to draw a rectangular selection box. After releasing the left mouse button, the system will automatically zoom in on the selected area.
	Show all variables	Display the curves of all variables in the timing diagram.

Hide all variables	Hide the curves of all variables in the timing diagram.
Restore scaling	Click "Restore" after the trace acquisition is complete to reset all scaling levels and display according to the following conditions: • If dual rulers exist, they will be centered, displaying all data between them plus 100 data points before the first ruler and after the second ruler. • If a single ruler exists, it will be centered, displaying up to 1000 data points. • If single and dual rulers both exist, the dual rulers will be centered, displaying all data between them plus 100 data points before the first ruler and after the second ruler. • If no ruler exists, place the x-axis at the origin and display up to 1000 points.
Restore all	Restore the display to show the full data range of both the X-axis and Y-axis for all currently visible curves.
Display with single ruler	The single ruler tool helps you precisely locate coordinates on a curve. Display the Y-axis measurement values in the variable operation table.
Display with dual rulers	The ruler tool is used to calculate measurement values between any two points on a curve. The vertical position of the two measurement cursors can be moved with the mouse. Display the relevant measurement values and the positional difference between the two measurement cursors in the timing diagram.
Show/hide variable operation table	
Show the actual sampling points	

Timing diagrams

Note

In the timing diagram, analog and digital quantities are displayed separately.

If the acquired variables contain both analog and digital quantities, the system will generate two independent timing diagrams. The analog timing diagram will be displayed at the top, and the digital timing diagram will be displayed at the bottom.

If the acquired variables are only analog or digital quantities, the system will generate the corresponding timing diagram.

Coordinated axes

- Horizontal axis (X axis): Analog and digital signals share the same horizontal axis.
- Vertical axis (Y axis)
  - Analog: Displays floating-point values.
  - Digital: Display only discrete ON/OFF states.

Ruler display

- Single **ruler**



  - When the mouse pointer hovers over the ruler, the cursor changes to a horizontal drag pattern (usually a double-headed arrow). At this point, hold down the left mouse button and drag left or right to move the ruler on the timing diagram.
  - The ruler displays the precise horizontal and vertical coordinate values of the currently selected curve (the curve highlighted in bold) at the ruler position.
  - While dragging the ruler, the displayed coordinate values will be updated synchronously in real time.

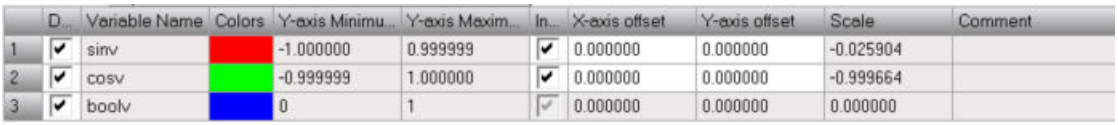
- Dual ruler



- The dual ruler consists of two independent vertical rulers on the left and right, which act on the analog and digital data in the timing diagram at the same time.
- When the mouse pointer hovers over either ruler, the cursor changes to a horizontal drag style (usually displayed as a double-headed arrow). At this time, hold down the left mouse button and drag to move the ruler on the timing diagram.
- The ruler displays the precise horizontal and vertical coordinate values of the currently selected curve (the curve highlighted in bold) at the ruler position.
- In the upper area between the two rulers, the system displays the horizontal increment (dx) and vertical increment (dy) of the currently highlighted curve between the two ruler points.
- The area between the two rulers is filled with a different background color to clearly indicate the selected data range.
- When you drag any ruler, the coordinate values and increment values displayed on the screen are refreshed synchronously.
- When the canvas is in pan mode, dragging within the range defined by two rulers will not pan the entire canvas, but will pan the dual rulers and their selected range as a whole.

Variable operation table

This table allows you to customize the display properties and behavior of data curves in the variable operation table.



	D...	Variable Name	Colors	Y-axis Minimu...	Y-axis Maxim...	In...	X-axis offset	Y-axis offset	Scale	Comment
1	✔	sinv		-1.000000	0.999999	✔	0.000000	0.000000	-0.025904	
2	✔	cosv		-0.999999	1.000000	✔	0.000000	0.000000	-0.999664	
3	✔	boolv		0	1		0.000000	0.000000	0.000000	

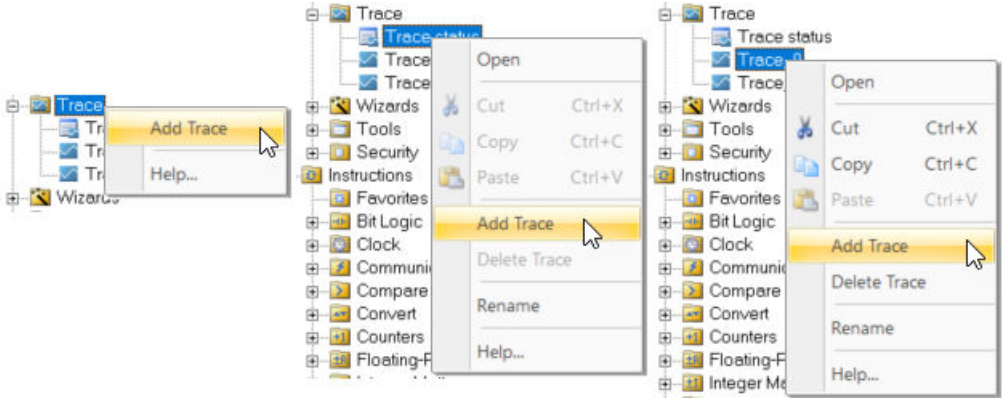
Show/hide	By checking the checkbox in this column, you can show or hide the corresponding data curve in the timing diagram.
Variable names	The variable name associated with the data curve cannot be edited.
Color	Displays the color of the data curve. This color cannot be edited.
Raw value	• By checking this option, the data curve will be displayed in its original Y-axis range without any scaling. • After unchecking this option, users can customize the scaling of the curve by entering values in the "Y-axis Max" and "Y-axis Min" fields below. **NOTICE**: Boolean variables do not support Y-axis scaling.
Y-axis Max	When the "Raw value" option is not checked, users can manually enter the maximum display value of the Y axis of the data curve here. The system will scale the curve proportionally according to the maximum and minimum values set by the user.
Y-axis Min	When the "Raw value" option is not checked, users can manually enter the minimum display value of the Y axis of the data curve here. The system will scale the curve proportionally according to the maximum and minimum values set by the user.
X offset	Used to adjust the display offset (translation) of the data curve in the X-axis direction.
Y offset	Used to adjust the display offset (translation) of the data curve in the Y-axis direction.
Ruler	This column displays the Y-axis value of the corresponding data curve at the position where the single ruler is located. This value cannot be edited. To ensure accurate data interpretation, the values displayed on the ruler always reflect the actual sampled values. This value is not affected by the currently applied X-axis or Y-axis offsets (panning) or the Y-axis scaling settings in the view.
Comment	Displays annotation information related to the data curve. This information is not editable.

6.11.4 Configuring the trace

"Trace" refers to the collection of trace configuration and acquisition data in STEP 7-Micro/WIN SMART V3. Trace configuration is used to specify how to collect data from PLC, including signal selection, sampling trigger conditions, etc. The trace sampling can be viewed in real time or offline in a time series diagram.

6.11.4.1 New trace

1. Add trace. The following options are available:
   - Right-click on the "Trace" folder in the project tree, and then select "Add trace".
   - Open the trace folder, right-click "Trace states" or an existing trace, and select "Add trace".



A new trace configuration will be created in the project tree, and the configuration window will open in the workspace.

Note

Each project supports the creation of up to 128 traces.

6.11.4.2 Configuring a signal table

Data type support and limitations

- A signal table allows you to add variables or addresses to be acquired as signals. A trace can be configured with up to 32 variable signals, and at least one signal must be configured.
- The variables used for trace signals must be consistent with the variables defined in the user's project. When the manually entered variable name does not exist or the address is invalid, the system will highlight the error in red.
- Supported address types: I/Q/V/M/SM/S/T/C/HC/AQ.

  Note

  The Trace function of STEP 7-Micro/WIN SMART V3 (3.2) does not support AI address type.

- Only pure data types (BOOL , BYTE , WORD , INT , DWORD , DINT , REAL) are supported. Complex data types such as UDT/Struct/Array/String are not supported. However, if a member (node) in a complex data type is a pure data type, the user can select a member of the pure data type.
- The length of a variable name (including the full path after expansion of complex data types) must not exceed 128 characters.

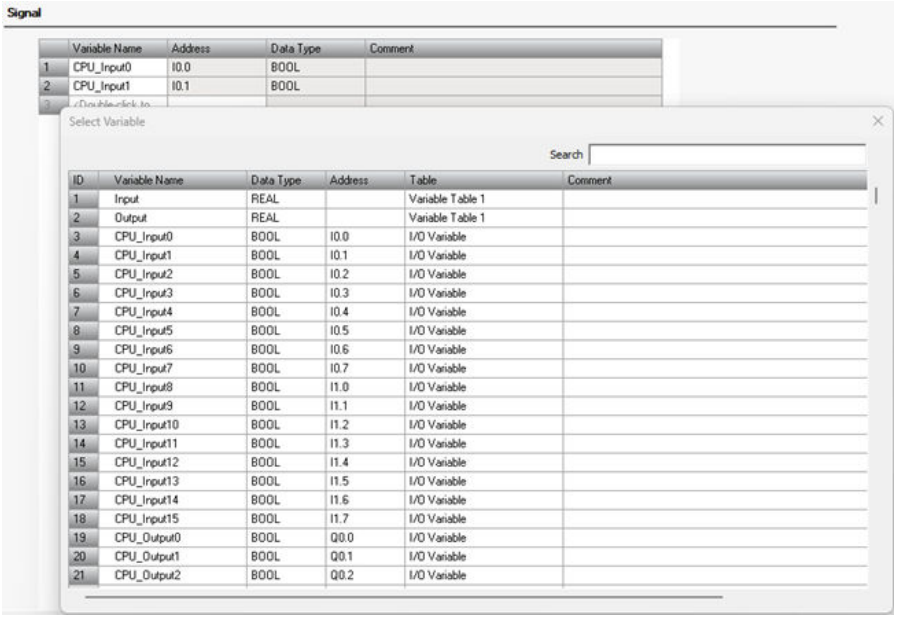
Preparation

- A trace is created and opened.
- The "Signals" area is open on the "Configuration" tab.

Add signal

To add the signals to be acquired, proceed as follows:

- Select from variable table:
  1. Double-click on a blank cell in the "Variable name" column, or right-click in the signal table area to open the "Select variable" window.
  2. Double-click a variable in the variable list to add it to the signal list.
  Variable search: You can also enter keywords in the search box to search. This search function supports searching structure member variables and highlights matching fields. After a variable is selected, its address, data type and comment will be displayed simultaneously.



- Manual input:
  - In the cell in the "Variable name" column, enter a name for the variable. After entering the variable name, its address, data type and comment will be displayed simultaneously.
  - Enter the address directly in the "Address" column. After entering the address, its data type will be displayed simultaneously.

Repeat the above steps until all signals to be acquired are entered into the table.

Editing a signal table

Delete a variable

To delete variables in a signal table, you can use the following methods:

- Delete a single row:
  - Select a row and press the Del key.
  - In the signal table area, you can delete variables through the right-click menu.
- Delete multiple rows: Select multiple rows and press the Del key.

Reselecting variables

In the signal table area, variables can be reselected via the right-click menu.

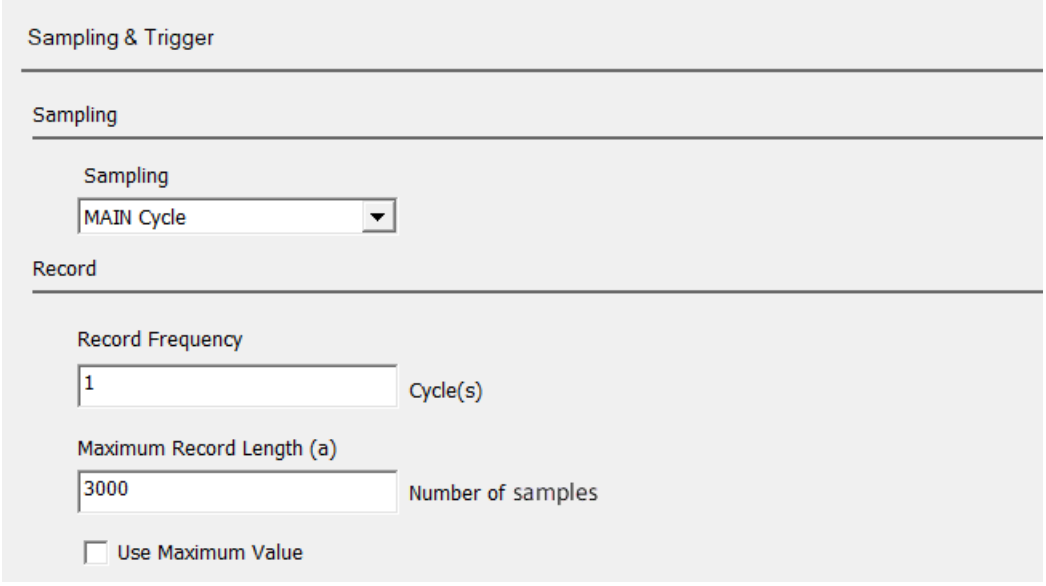
Download/activation verification

When a trace task is downloaded or activated, the system will perform a unified check on all variable names:

- Variable exists: If the variable exists in the project, the system will refresh its address, type and other information.
- Variable does not exist: If the variable does not exist in the project or is invalid, the system will report an error.

6.11.4.3 Configuring sampling conditions

The sampling and recording condition configuration is shown in the figure below



Sampling

- MAIN cycle: When configured as the MAIN cycle, the trace will record the values of the trace signals at the end of the scan cycle.

  Note

  Since the scan cycle duration fluctuates depending on the actual runtime conditions of the user's project and program, the trace variables are not recorded at fixed time intervals.

- IRT cycle: When configured for IRT cycle, the Trace will record the Trace signal value according to the IRT cycle.
  When configured for IRT cycle, the configuration interface will display the sampling time, which is the total time required to complete a specified number of sampling points, in seconds.
  Conversion formula: Sampling time = IRT cycle × Sampling frequency × Number of sampling points

  Note

  To use the IRT cycle, ensure that the IRT axis has been added and correctly configured in the project.

  Note

  The effect of recording variables on IRT

  Traces will be sampled when the IRT driving thread ends. When too many Trace variables are configured, this may affect the system's IRT function. Specifically, excessively long acquisition times may prevent the IRT task from being completed within one IRT cycle, thereby compromising the system's real-time performance.

Recording frequency

The recording frequency parameter is used to set the execution interval for trace recording, specifying how many cycles elapse between each recording.

Maximum record length

One signal sample counts as one measurement point. The configurable range is a minimum of 1 point and a maximum of 32768 points.

Requirements

- A trace is created and opened.
- The "Sampling & trigger" area is open in the "Configuration" tab.

Procedure

To configure the cycle and the duration of a recording, proceed as follows:

1. Select a sampling period from the "Sampling" drop-down list.
2. In the "Record frequency" input field, enter the cyclic interval at which the data should be recorded.
3. Specify the record length. The following options are available:
   - Enter the number of samples in the "Maximum record length" input field.
     The range of recording frequencies is: [1, 32768].
   - After checking the "Use Maximum Value" checkbox, the maximum record length will be displayed as 32768 and cannot be modified. After unchecking the box, the maximum record length will revert to its previous value.

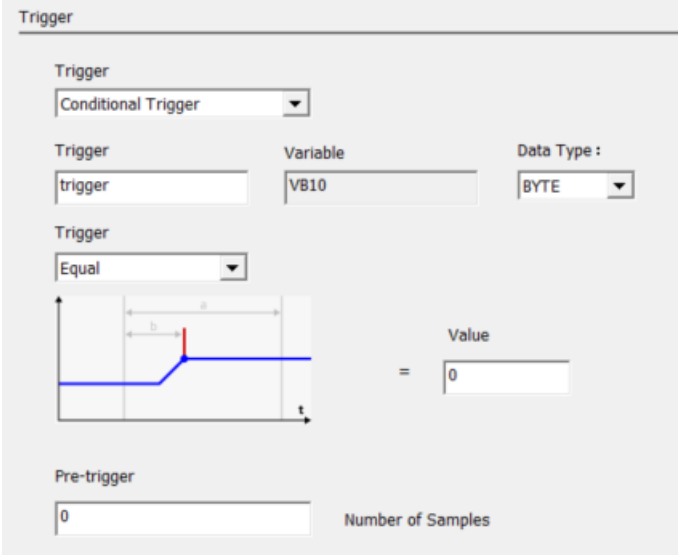
Configuring trigger conditions

In the trigger configuration interface, you can define the conditions for starting trace data recording. Trigger configuration is the starting condition for trace sampling.

Note

For the trigger configuration to take effect, at least one of the following conditions must be met:

- The user clicks "Activate trace" in the trace control panel
- Set the value of the Trace control special memory area (Page 1223) to 1



Trigger mode

Trigger mode	Description
Trigger immediately	After activating the trace, recording begins immediately without the need to configure additional parameters.
Conditional trigger	After activation, the trace enters a waiting state and will only begin recording once the trigger variable meets the set conditions.

Trigger variable

Please refer to the following table for variable trigger conditions:

Variable trigger conditions	Data type	Description
=TRUE	BOOL	Recording starts when the state of the trigger is TRUE
=FALSE	BOOL	Recording starts when the state of the trigger is FALSE
Rising edge	BOOL	Recording starts when the trigger condition changes from FALSE to TRUE
Falling edge	BOOL	Recording starts when the trigger condition changes from TRUE to FALSE
= value	BYTE、WORD、INT、DWORD、DINT、REAL	Recording starts when the trigger value equals the configured value for this event
<> value	BYTE、WORD、INT、DWORD、DINT、REAL	Recording starts when the trigger value is not equal to the configured value for this event
Value change	All types	Recording starts when the trigger value and the value at the time of recording activation are different

Pretrigger

Pretrigger refers to the measurement points recorded by the trace before the trigger conditions are met. For example, if the maximum record length is 1000 measurement points and the pre-trigger point count is set to 10 measurement points, then 990 measurement points will be recorded after the trigger condition is met.

Requirements

- The "Sampling & Trigger" area is already open in the "Configuration" tab.

"Trigger immediately" trigger condition

To start the acquisition immediately after activation, in the "Trigger mode" dropdown menu, select the entry "Immediate trigger".

"Conditional trigger" trigger mode

To start the acquisition depending on a condition, proceed as follows:

1. Select the "Conditional trigger" entry in the "Trigger mode" drop-down list.
2. Select a variable. The following options are available:
   - In the "Trigger variable" field, double-click a blank cell to open the variable table and select a variable.
   - Enter the address or the symbolic name of the variable directly in the input field for the trigger variable.
3. Select a trigger condition in the "Trigger conditions" drop-down list.
   - For non-BOOL variables, when you select the "Equal" or "Not equal" option, a value input box will be displayed. Its value range depends on the current data type.
   - For the BOOL type, the options include "High", "Low", "Rising edge", "Falling edge", "Value change".
4. Enter the number of samples for pretrigger in the "Pretrigger" input box.
   In order to acquire a period before the trigger event, enter a value greater than 0 in the input field for the pretrigger.

6.11.5 Managing traces in the project

6.11.5.1 Opening a trace

When opening a project, the system will first check whether the path to the trace file is saved in the project. Based on the results, the system will perform the following actions:

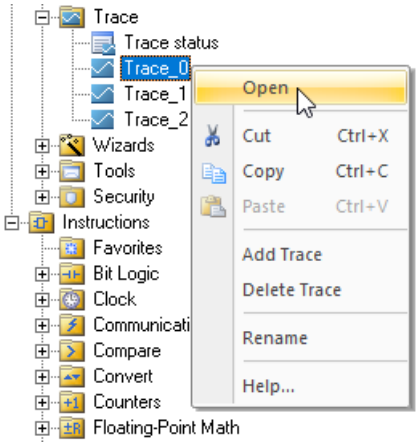
- If the trace file is successfully found, the system will read its configuration information and generate the corresponding trace data configuration accordingly.
- If the specified trace file is not found, the project opening process will not be affected, but the system will prompt that the trace file is missing.

A maximum of 10 lost trace names are displayed. The excess will be indicated by "...".

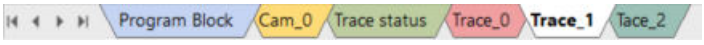
When the project is saved and reopened later, the relevant errors will no longer be reported. At this point, the saved project will only contain the trace configurations that were successfully loaded for the first time.

Opening a trace

1. Open a trace. The following options are available:
   – Open the "Trace" folder in the project tree, then double-click the trace name or use the right-click menu to open the trace you wish to access.



   – Click the tab below the program editor



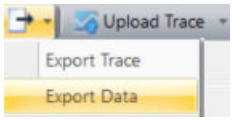
The trace configuration you want to edit opens in the workspace.

6.11.5.2 Importing/export inga trace

Exporting configuration & data

You can export by clicking on the "Export" button to export the current system configuration and related data into a file.

Exporting a trace



- Export the data and configuration to XML format files for subsequent importing the traces. Neither configurations nor data will be lost.

Exporting data

- Export the currently collected data (excluding configuration) to CSV.

Importing a trace

You can import configurations and data from a specified XML file by clicking the "Import" button in the toolbar.

6.11.6 Managing traces in PLC

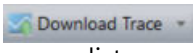
6.11.6.1 Downloading/uploading traces

A "task" is an executable instance of a configured trace after it has been downloaded to the PLC. It represents a specific data acquisition to be executed on the PLC. Each PLC can carry up to 4 tasks.

Note

- Trace download/upload operations can be performed regardless of whether the PLC is in Run or Stop state.
- During the trace activation process, trace download/upload operations are not supported.

Download the configuration to the PLC

- Click the inverted triangle icon next to the "Download trace" 

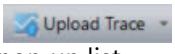
button in the toolbar, and select the task you want to download from the pop-up list.

Note

The download of trace configurations is independent of the STEP7-Micro/WIN SMART V3 project download. The specific behavior is as follows:

- When downloading a trace configuration to a specified task in the PLC, the system performs a configuration check. Common reasons for download failure include:
  - The PLC program is inconsistent with the STEP7-Micro/WIN SMART V3 project
  - No variables are configured
  - The variable does not exist or is invalid
  - The current task is acquiring data
- If the trace has already been downloaded to another task, the system will prompt you to delete the original trace
- After the download is successful, the trace data of the current task in the PLC will be overwritten. Meanwhile, the value in the trace status special register area will be set to 1

Uploading the trace to the PLC

- Click the inverted triangle icon next to the "Upload trace" 

button in the toolbar, and select the task you want to upload from the pop-up list.
During uploading, a dialog box will pop up, prompting you to overwrite the current trace. After confirming the upload, the system will perform the trace upload.

6.11.6.2 Activating/deactivating a trace

Preparation

- The trace configuration has been downloaded to the PLC.

Activating a trace

There are two ways to activate a trace.

- In the configuration workspace, click the "Activate" button in the toolbar to complete the activation.
  After activation, the view automatically switches to the timing diagram interface, which displays the collected data. The data displayed on the timing diagram is not in real time. Data will be loaded in batches. The next batch will be loaded only after the current batch is fully displayed.
- Activate trace in the trace special memory area (Page 1223): After data collection is complete, you can upload the data to STEP 7-Micro/WIN SMART V3 and view it in the timing diagram interface.

Note

It can only be activated when the PLC is in RUN state. If the PLC is in STOP state, the system will prompt you to start the PLC. After confirmation, the PLC starts and automatically activates the trace.

Note

- The timing diagram only displays non-real-time curve data; for acquisition status, check the status bar. The status bar updates every 5 seconds.
- During data acquisition, the trace configuration cannot be modified.
- During data acquisition, upload, download, or delete operations cannot be performed on the trace.

Pausing the operation

Pause

During data acquisition, the interface will display data in real-time scrolling. You can pause data acquisition by performing the following operations.

- Drag the canvas: Move the display area by dragging the mouse.
- Scaling operation Including using the mouse wheel to zoom in/out, or adjusting the view through other zoom tools (Page 188).
- Switch single/dual ruler: Enable or disable single and dual ruler display.

Exiting paused state

You can use the Restore or Restore all function to exit the paused state and resume interface refresh.

Deactivating

There are two ways to deactivate a trace.

- During data acquisition, click the Stop button to halt the acquisition process
- Stop the trace in the trace special memory area.

Note

If the PLC goes into STOP mode during the trace acquisition, the trace stops automatically.

6.11.6.3 Trace status table

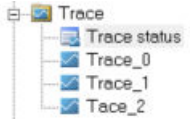
"Trace status" is used to observe, upload and delete the PLC trace task in STEP 7-Micro/WIN SMART V3.

Requirements

- A connection has been established with the STEP 7-Micro/WIN SMART V3, and the PLC has been logged into with the corresponding user credentials

Opening the trace status table

1. The trace status is located under the "Trace" folder in the project tree. Double-click to open the trace status interface.



Result: The trace status opens in the workspace.

Trace task status

The trace status table only reflects the actual runtime status of traces in the PLC and does not perform consistency checks with the current project or local files. Therefore, the status displayed in the table may not directly correspond to the configurations in your local project.

Task number	Trace status	Trace name	Variable number		
0	Configured	Trace_0	3		
1					
2	Not configured		0		
3	Not configured		0		

	Description
Task	The sequence number of the current task in the concurrent task list. The PLC can run a maximum of 4 trace tasks simultaneously.
Trace status	The status of the trace currently running in the current task.

	Description
Trace name	The name of the trace currently running in the current task.
Number of variables	The number of variables included in the trace currently running in the current task.
	Upload the trace currently running in the current task to STEP 7-Micro/WIN SMART V3. When performing an upload operation, you need to explicitly choose whether to create a new trace or overwrite an existing trace. • Create new: Upload the trace under the PLC task and generate a new trace configuration • Overwrite: Upload the trace under this task to the PLC and overwrite the trace configuration with the same name in the current STEP 7-Micro/WIN SMART V3.
	Delete a trace in the PLC

6.11.6.4 Deleting the trace configuration and data in the PLC

You can delete the trace configuration and its acquired data in the PLC by any of the following methods:

- Reset to factory
- Perform a CPU Clear operation
- Click the "Delete"  button in the toolbar. This method can be used to delete a trace regardless of whether the PLC is in the Run or Stop state

PLC device configuration

7

7.1 Configuring the operation of the PLC system

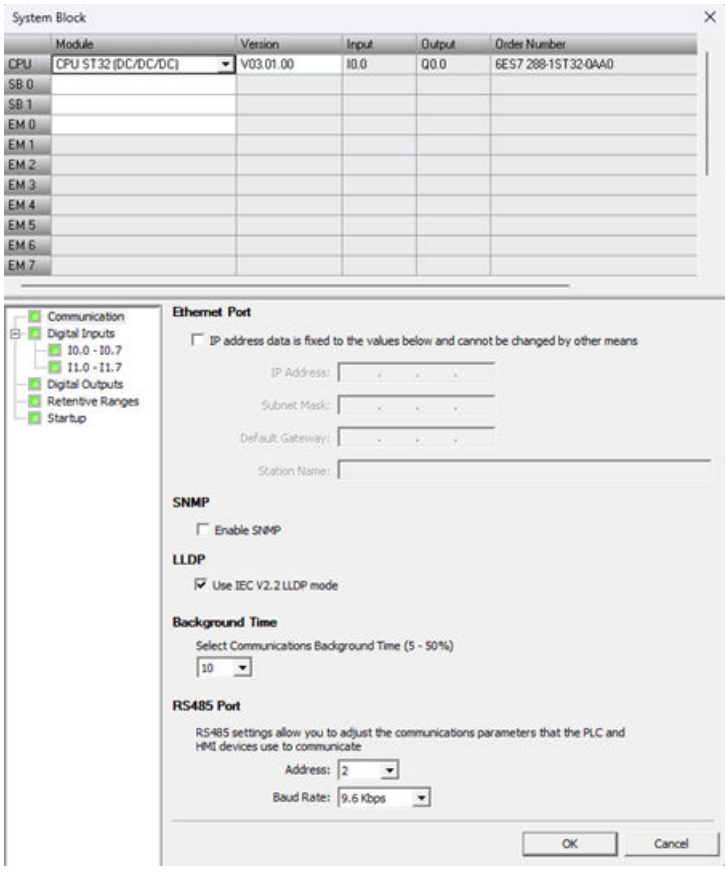
7.1.1 System block

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.

Changing system settings

Switching CPU module

- Switching from an S7-200 SMART 3.0 CPU to an S7-200 SMART G2 CPU:
  - If no SB or EM modules are configured, you can switch directly.
  - If SB or EM modules are configured, the system will prompt that incompatible SB/EM configurations will be cleared. Click "No" to cancel the switch, or click "Yes" to switch the CPU and clear all SB/EM configurations.
- Switching from an S7-200 SMART G2 CPU to an S7-200 SMART 3.0 CPU:
  - If no SB or EM modules are configured, you can switch directly.
  - If the SB or EM module is configured, the system will prompt you to manually clear the relevant configuration before switching.

Note

Switch the CPU type or version in the system block:

- Switch the CPU type from SR40/60 or ST40/60 to ST32, or switch the ST32 version from V3.1 to V3.2:
  The system will prompt: "After switching CPU type or version, the motion control instructions and structure also need to be updated. Switch?" "
  - Select [No]: Switch failed, motion control instructions and TO (Page 897) remain at version V3.1.
  - Select [Yes]: Switching successful. Motion control instructions and TO have been updated to version V3.2.
- Switch the CPU type from ST32 (V3.2) to SR40/60 or ST40/60, or switch the ST32 version from V3.2 to V3.1:
  - If the PN IRT axis is configured: Switching failed, system message: "The current CPU version does not support the PN IRT axis. Please delete and try again." "
  - If the PN IRT axis is not configured: The system will prompt: "After switching CPU type or version, the motion control instructions and structure also need to be updated. Switch?" "
    - Select [No]: Switching failed.
    - Select [Yes]: Switching successful. Motion control instructions and TO have been updated to version V3.1.

EM module modification

After the EM is modified, the default address of the current module is reset based on the current slot. If the address does not overlap with an address of another module, the address of the another module is maintained. If the address of the current module overlaps with an address of another module, the address of the overlapping module is set to the default address corresponding to the slot.

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 208)
- Digital inputs and pulse catch bits (Page 211)
- Digital outputs (Page 214)
- Retentive Ranges (Page 215)
- Startup (Page 217)

Configuration options specific to other devices such as analog inputs (Page 220), analog outputs (Page 223), RTD analog inputs (Page 225), Thermocouple (TC) analog inputs (Page 229), RS485/RS232 CM01 communications signal board (Page 232), Battery BA01 signal board, and additional digital inputs and outputs are accessible from the system block when you add those modules.

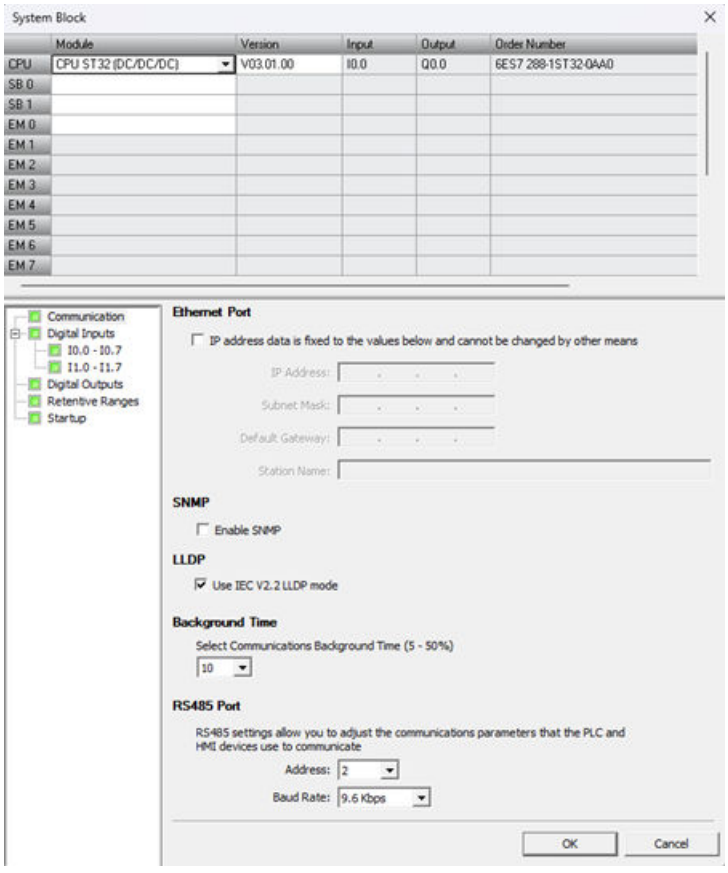
You must establish communications between STEP 7-Micro/WIN SMART and your CPU before you can download or upload a system bock.

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 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.

7.1.2 Configuring communication

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



Ethernet ports

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 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.
- 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:

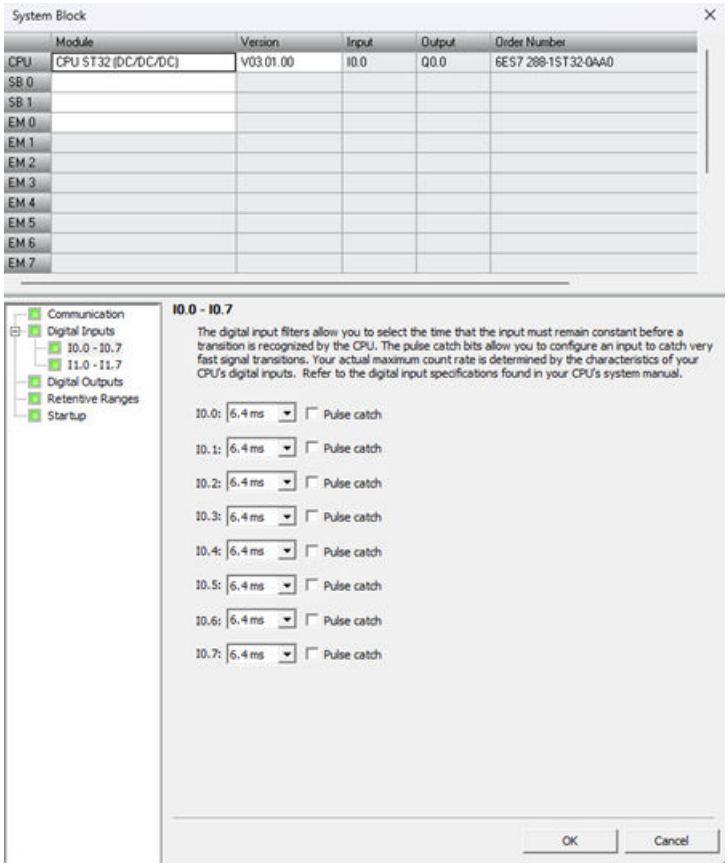
- Use a USB-PPI cable to program all CPU models through any serial port, including the RS485 port, the signal board port, and the DP01 PROFIBUS port.
- Use the RS485 and RS232 ports for HMI access (Data read/write) and Freeport communications.

Note

A direct connection to the USB-PPI cable is not supported by the S7-200 SMART G2 CPU. To connect an S7-200 SMART G2 CPU with this cable, you will need to make your own RS485 connection wire.

7.1.3 Configuring the digital inputs

Click the Digital Inputs node of the System Block dialog to configure digital input filters and pulse catch bits.



Digital input filter

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 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

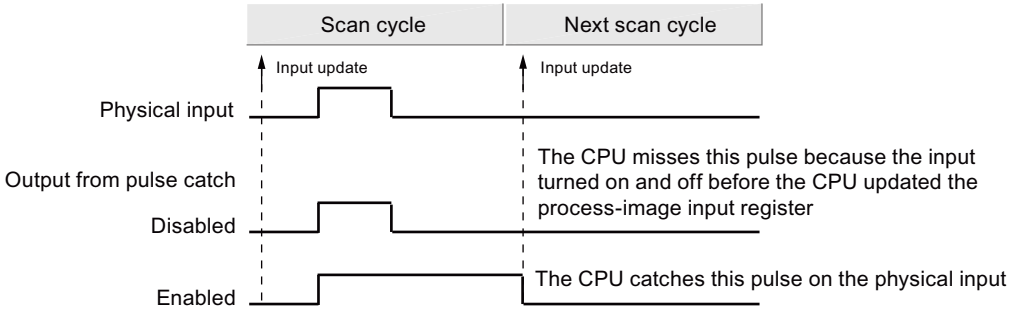
The S7-200 SMART CPU provides a pulse catch feature for digital input points. The pulse catch feature allows you to capture high-going pulses or low-going pulses that are of such a short duration that they would not always be seen when the CPU reads the digital inputs at the beginning of the scan cycle.

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.

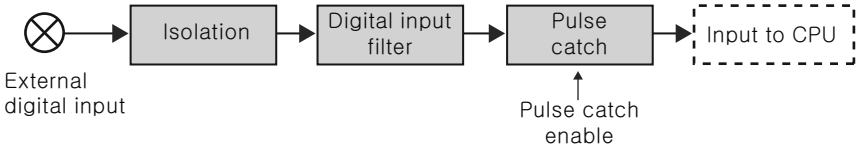
You can enable individual pulse catch operation for the first sixteen digital input points (I0.0 through I0.7 and I1.0 through I1.7), dependent upon the CPU model.

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.

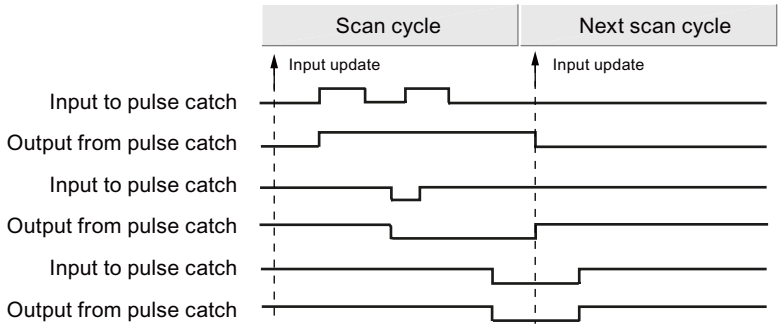
The figure below shows the basic operation of the S7-200 SMART CPU with and without pulse catch enabled:



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:

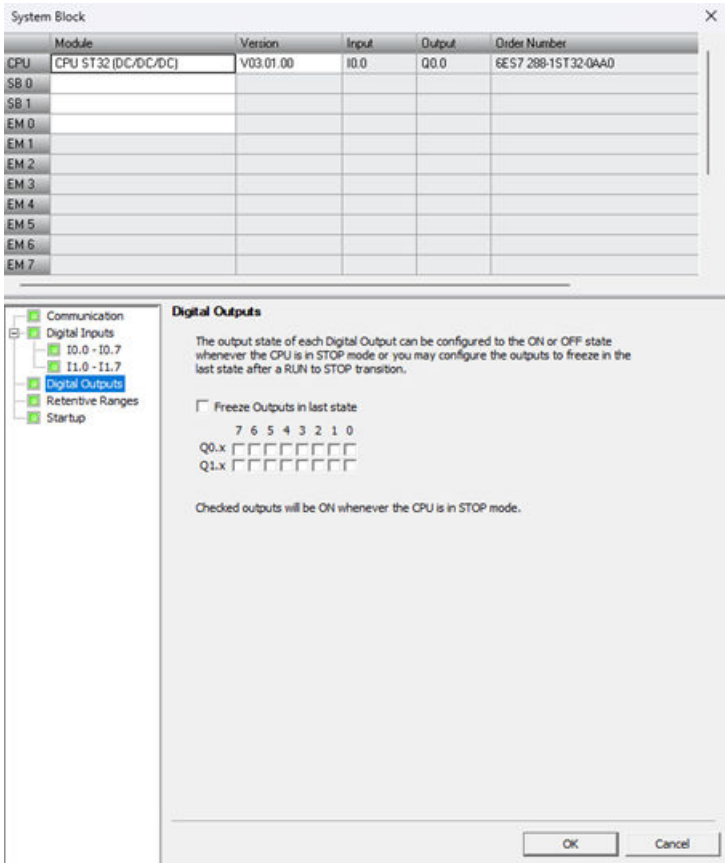


The figure below shows the response of an enabled pulse catch function to various input conditions. If you have more than one pulse in a given scan, only the first pulse is read. If you have multiple pulses in a given scan, you should use the rising/falling edge interrupt events:



7.1.4 Configuring the digital outputs

Click the Digital Outputs node of the system block to configure options for the digital outputs of the selected module.



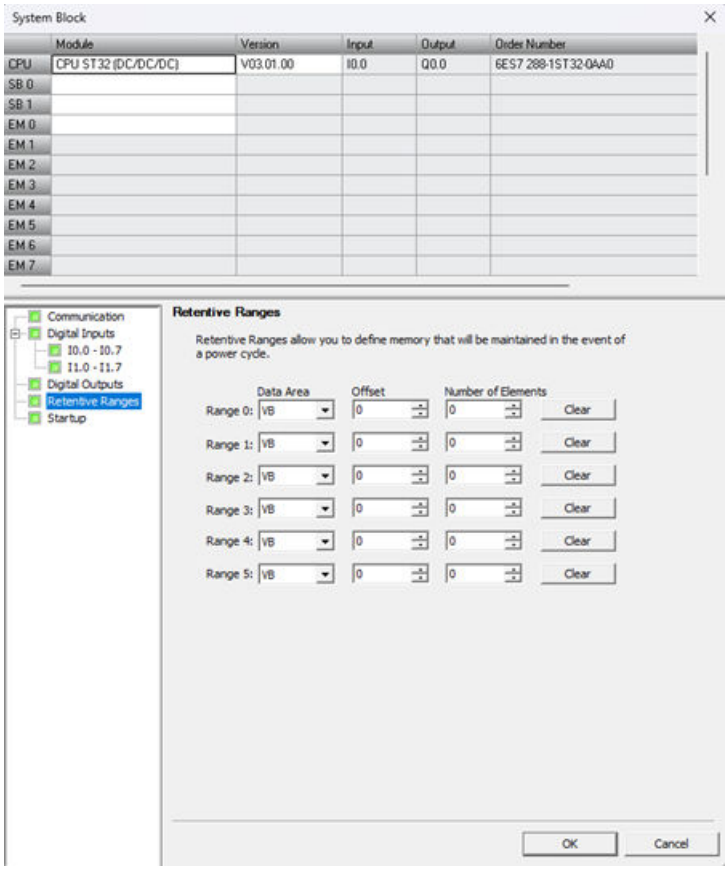
You can set digital output points to a specific value when the CPU is in STOP mode, or preserve the output states that existed before the transition to STOP mode.

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).

7.1.5 Configuring the retentive ranges

Click the "Retentive ranges" node of the "System block" 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, CPU ST60 and ST32 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:

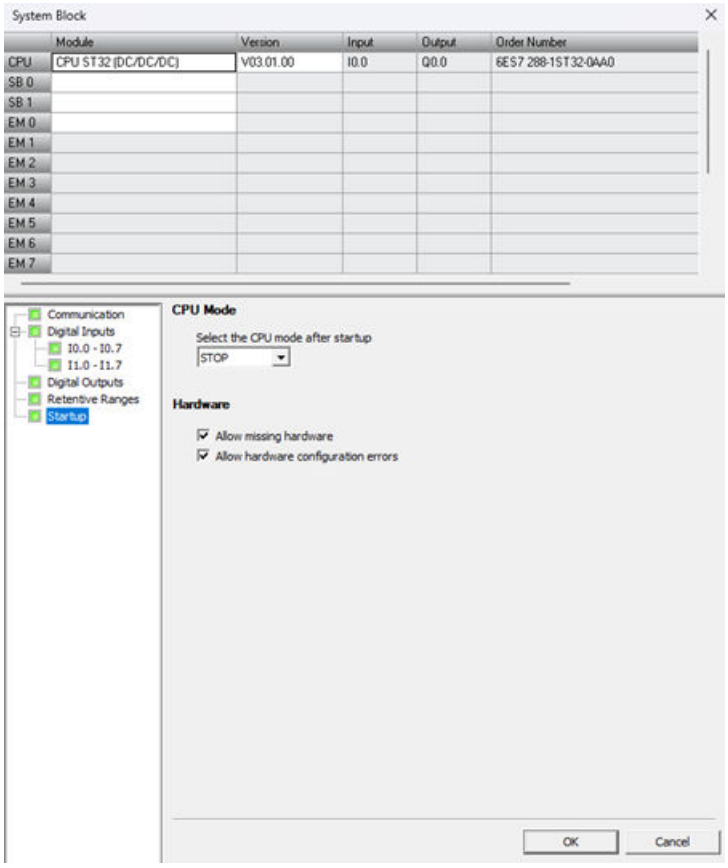
- **At power down:**
  The CPU saves the memory ranges designated as retentive to permanent memory.
- **At power up:**
  The CPU first clears V, M, C, T or DB2 memory, copies any initial values from the data block to V memory, and then copies the saved retentive values from permanent memory to RAM.

S7-200 SMART CPU memory addresses for retentive ranges

Data type	Description	CPU SR40 CPU ST40	CPU ST32 CPU SR60 CPU ST60
V	Data memory	VB0-VB81919	VB0-VB102399
T	Timer	T0-T31, T64-T95, T256 - T261, T268 - T291, T316 - T413	
C	Counter	C0-C511	
M	Flag bits	MB0-MB63	

7.1.6 Configuring the startup options

Click the Startup node of the system block 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 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).

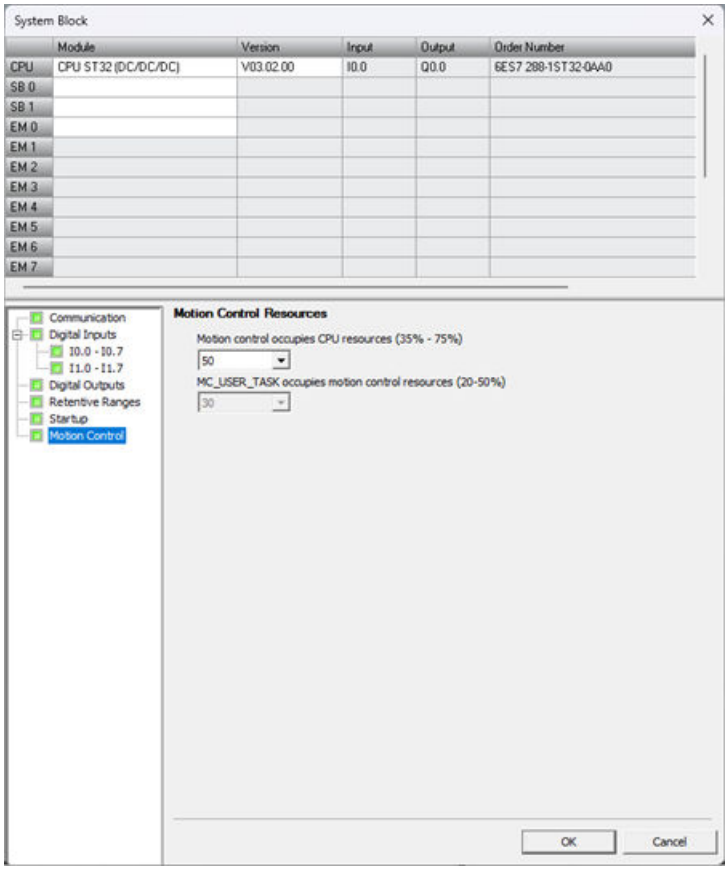
If you deselect one or both of the selections, the CPU is prohibited from entering RUN mode if any of the disallowed conditions are true.

7.1.7 Motion control

Click the "Motion control" node of the "System blocks" dialog box to configure the motion control resources for your PLC.

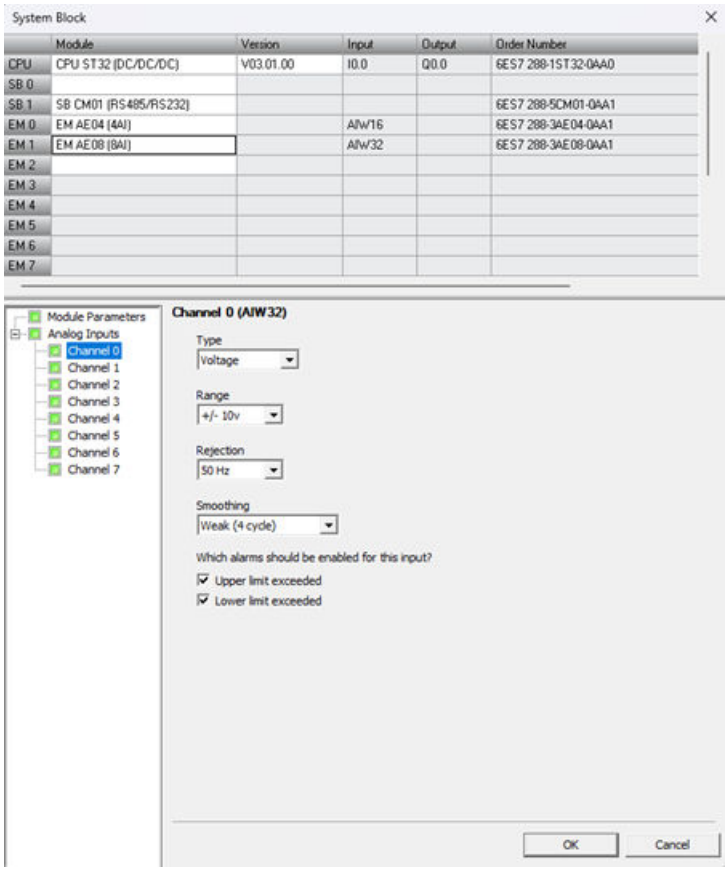
You can adjust the allocation of CPU resources among different tasks based on your actual needs.

- Proportion of CPU resources used for the motion control task
- Proportion of motion control resources used for ME_USER_TASK


System Block
Module
Version
Input
Output
Order Number
CPU
CPU ST32 (DC/DC/DC)
V03.02.00
I0.0
Q0.0
6ES7 288-1ST32-0AA0
SB 0
SB 1
EM 0
EM 1
EM 2
EM 3
EM 4
EM 5
EM 6
EM 7
Communication
Digital Inputs
I0.0 - I0.7
I1.0 - I1.7
Digital Outputs
Retentive Ranges
Startup
Motion Control
Motion Control Resources
Motion control occupies CPU resources (35% - 75%)
50
MC_USER_TASK occupies motion control resources (20-50%)
30
OK
Cancel

7.1.8 Configuring the analog inputs

Click the "Analog inputs" node of the "System block" 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, configure the type as voltage or current. The type selected for the evennumbered channels also applies to the oddnumbered 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

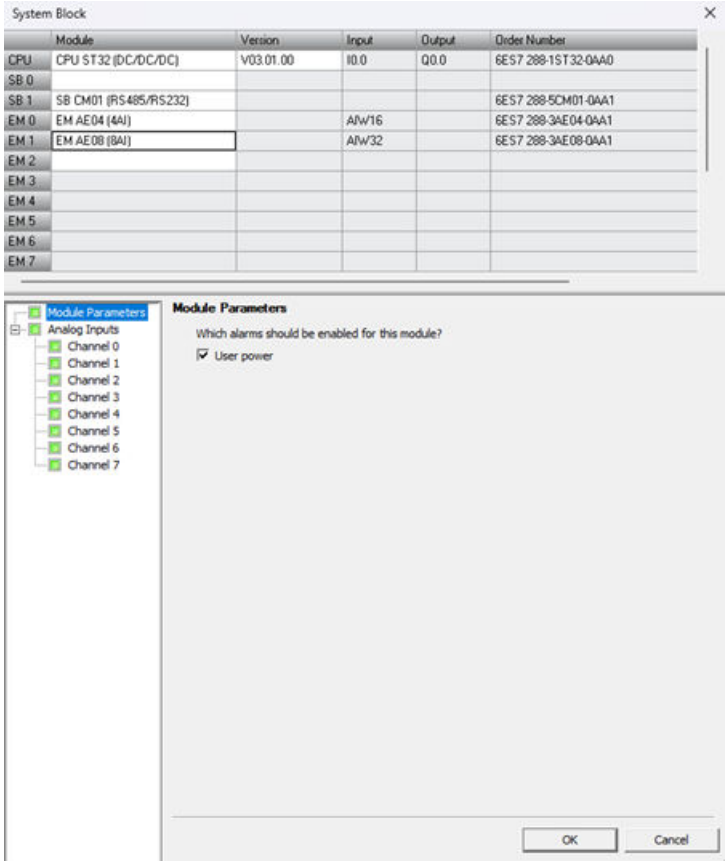
You can also configure the module to smooth the analog input signal over a configured number of cycles, thus presenting an averaged value to the program logic. You have four choices for the smoothing algorithm:

- 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).)



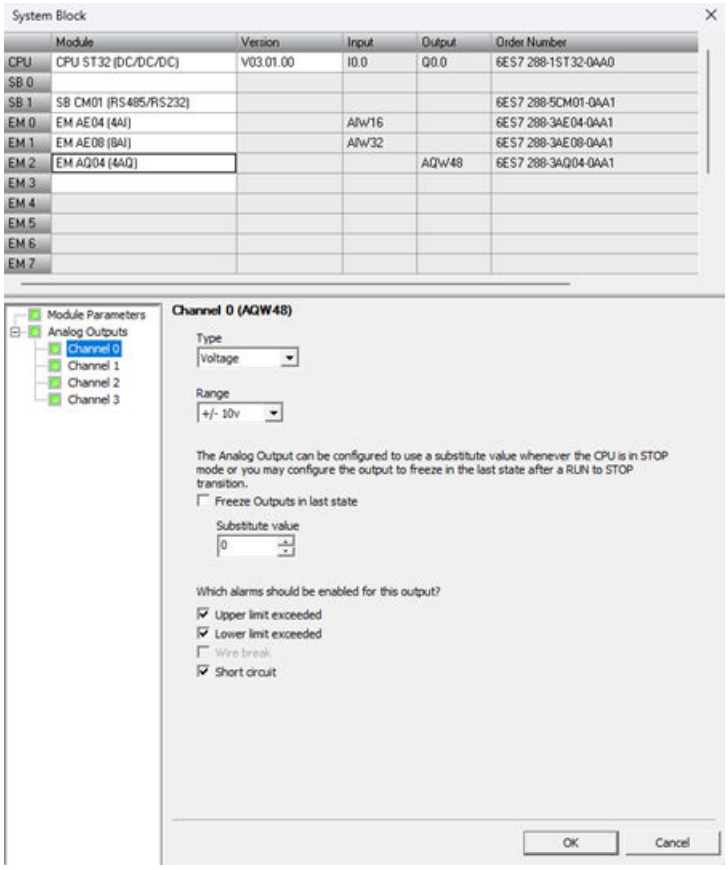
Reference to the analog inputs technical specifications

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 1165)
- Rejection: "Sample time and update times for the analog inputs" (Page 1165)
- Smoothing: "Step response of the analog inputs" (Page 1165)

7.1.9 Configuring the analog outputs

Click the "Analog outputs" node of the "System block" 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, configure the type as 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

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.

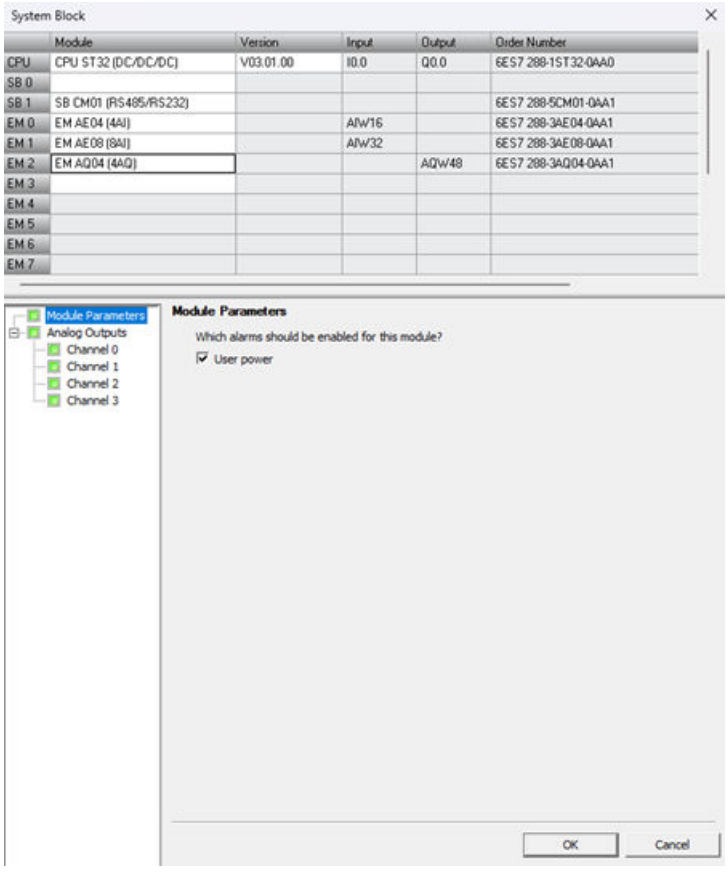
You have two ways to set the analog output behavior in STOP mode:

- 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.))

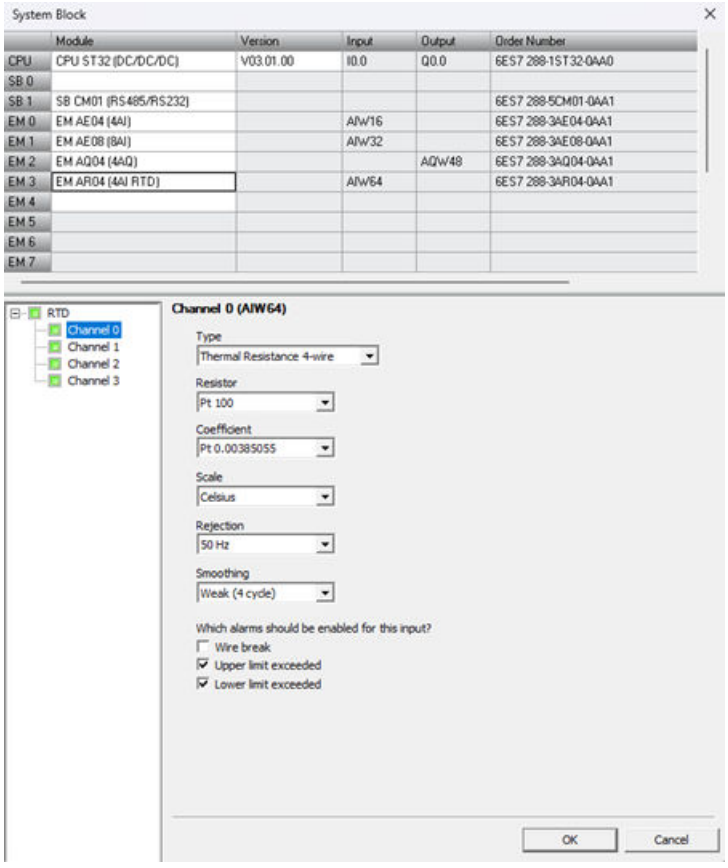


Reference to the analog outputs technical specifications

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 1166) technical specification

7.1.10 Configuring the RTD analog inputs

Click the RTD analog Input node of the system block 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 7-1 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.	• 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
• 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
• 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:

- degree 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

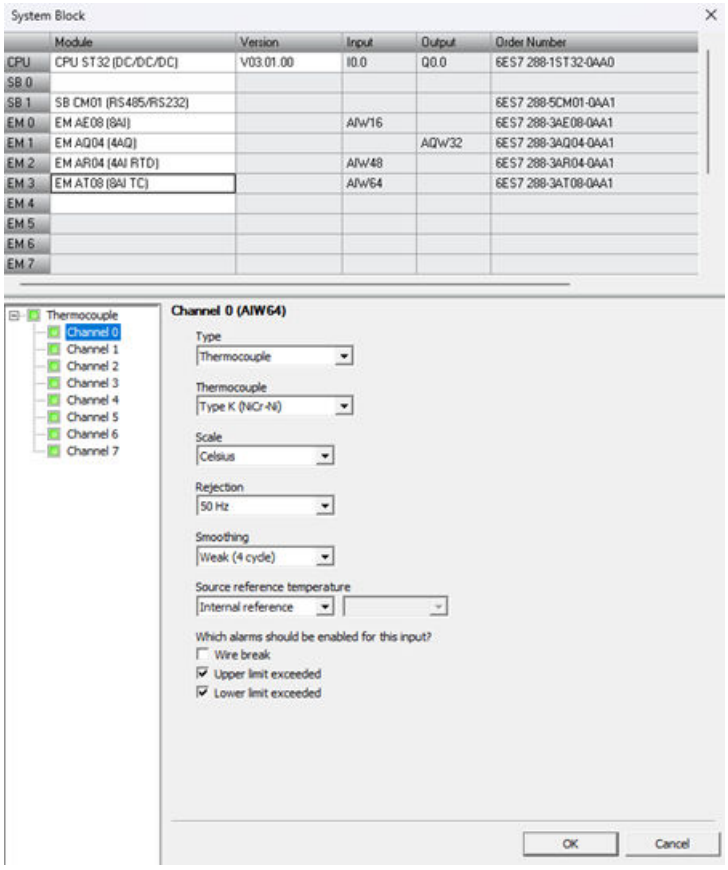
Alarm configuration

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

7.1.11 Configuring the TC analog inputs

Click the TC (Thermocouple) analog input node of the system block 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:

- degree 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

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.

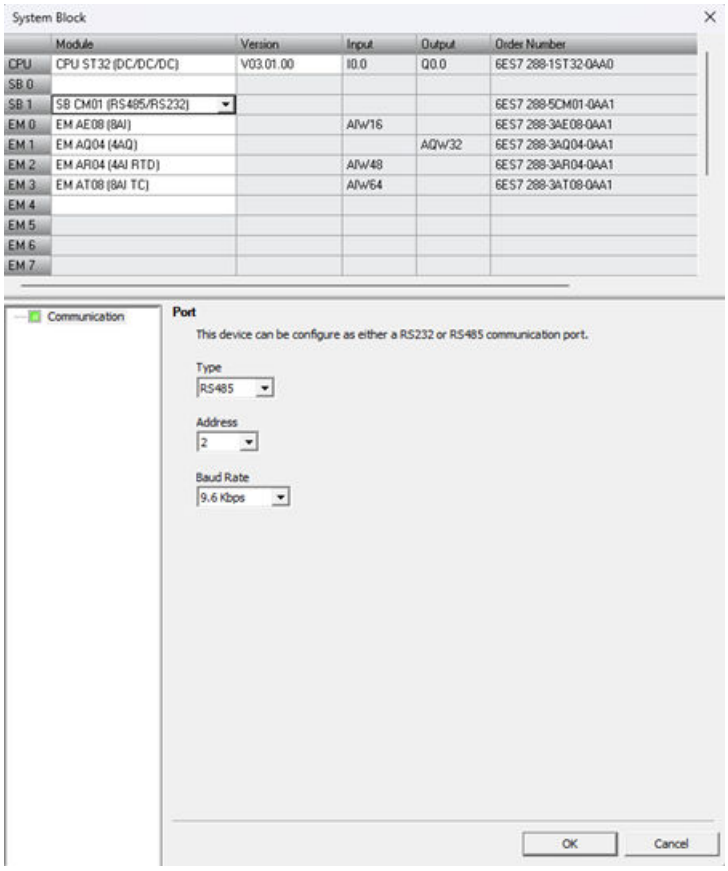
Cold junction compensation is used to compensate for the connector thermocouple. Thermocouple tables are based on a reference junction temperature, usually zero degrees Celsius. The cold junction compensation compensates the connector to zero degrees Celsius. The cold junction compensation restores the 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.

7.1.12 Configuring the RS485/RS232 CM01 communications signal board

Click the CM01 communications signal board node of the system block 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

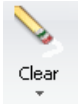
Choose the desired data baud rate from the dropdown list:

- 9.6 Kbps
- 19.2 Kbps
- 187.5 Kbps

7.1.13 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

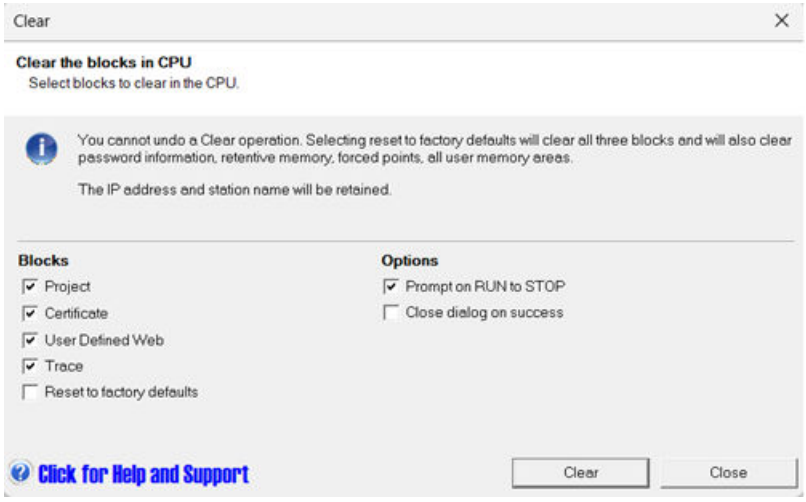
Effect of clearing PLC memory on outputs

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 - projects, certificates, traces, 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 1191) to initial values.

What to do if you forget the PLC password

If you forget the PLC password (Page 59), you can clear the PLC memory using a "Reset-to-factory-defaults memory card (Page 235)" 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 138) 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 133) 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.

7.1.14 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.

7.2 High-speed I/O

High-speed counter

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 344).

High-Speed Pulse Output

Standard CPU models support high-speed pulse output, capable of generating a high-speed pulse train output (PTO) or pulse width modulation (PWM) signal 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 with a specified number of pulses (ranging from 1 to 2,147,483,647 pulses) and a specified frequency (Hz). The PTO function can be programmed to generate a single pulse train or a pulse envelope containing multiple pulse trains. For example, a pulse envelope can be used to control a stepper motor through a simple ramp-up, run, and ramp-down sequence, or a more complex sequence.

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 duration equals the cycle time, the duty cycle is 100%, and the output remains continuously on. When the pulse duration is 0, the duty cycle is 0%, and the output is off.

To learn more about it, refer to Pulse output instruction (Page 370). The chapter on open-loop motion control provides additional information using PWM (Page 1039).

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 1039) for a full description of the motion capabilities in your CPU.

Web server

8

8.1 Web server Overview

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
- Configuring the HTTPS certificates for the Web server with certificate management

For detailed steps, refer to "Tools" > "Wizards" > "Web Server Wizard" in the online help of STEP 7-Micro/WIN SMART V3.

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 233) 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 250) 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.

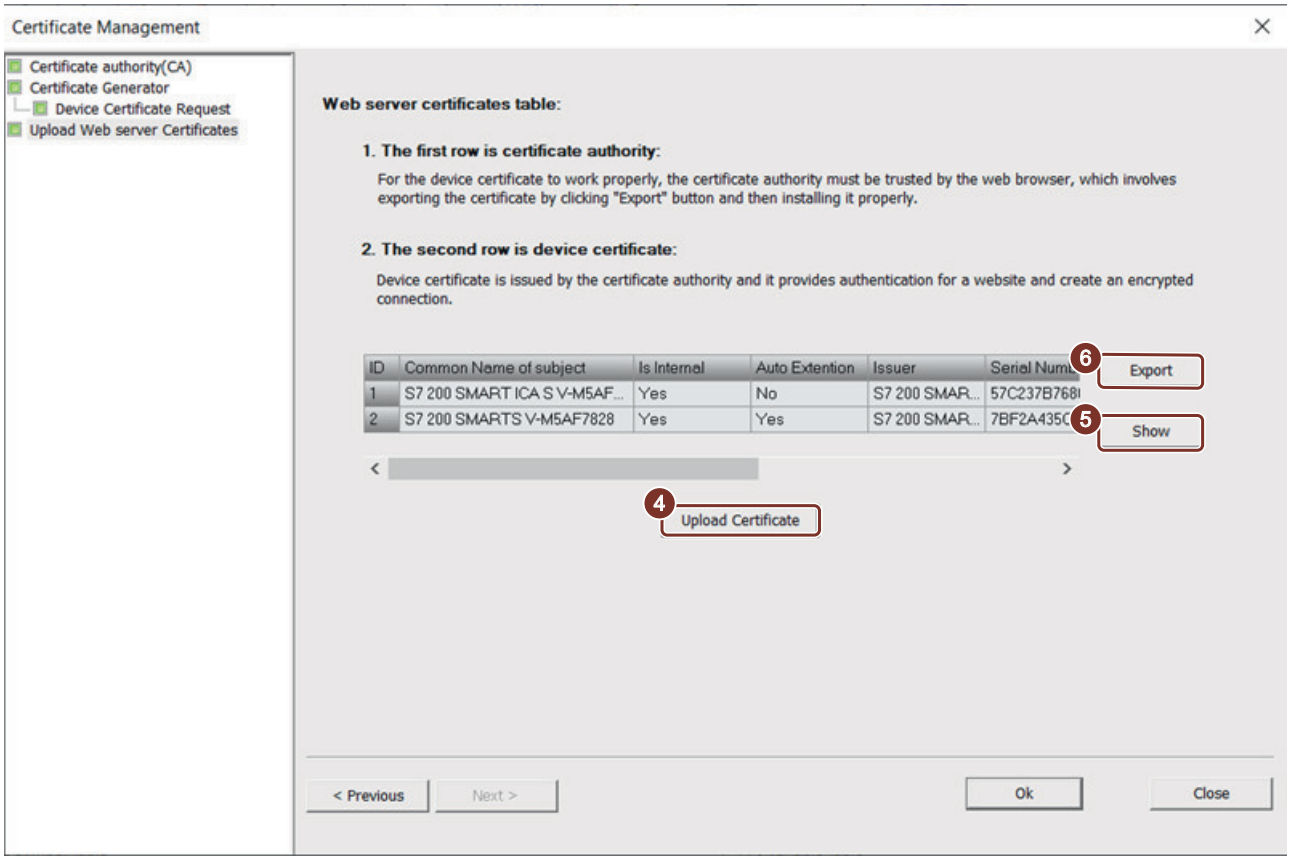
8.2 Downloading and installing the certificate of CA

For visiting the Web server from a browser, you need to install the certificate of certificate authority (CA) in your Internet options.

8.2.1 Get the certificate of CA

You can get the certificate of certificate authority as follows:

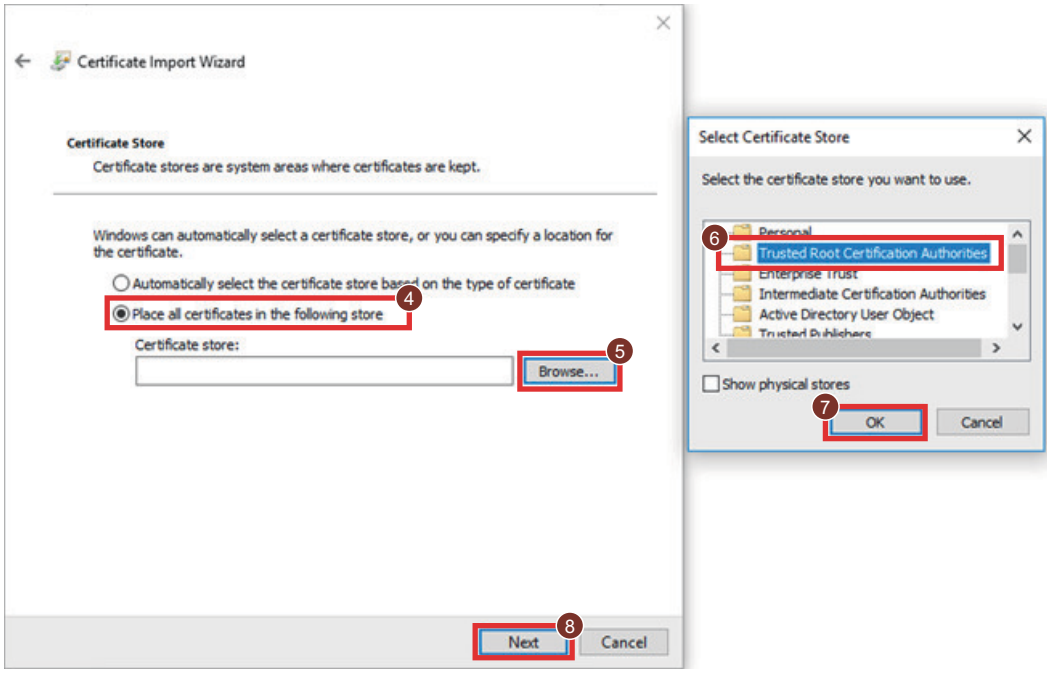
1. Set up the Ethernet communication with the CPU.
2. Open the Certificate management wizard.
3. Select "Upload Web server certificates" in the navigation pane.
4. Click "Upload Certificate".
   The certificates on the CPU are uploaded in the Web server certificates table: The certificate in the first 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".



8.2.2 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.

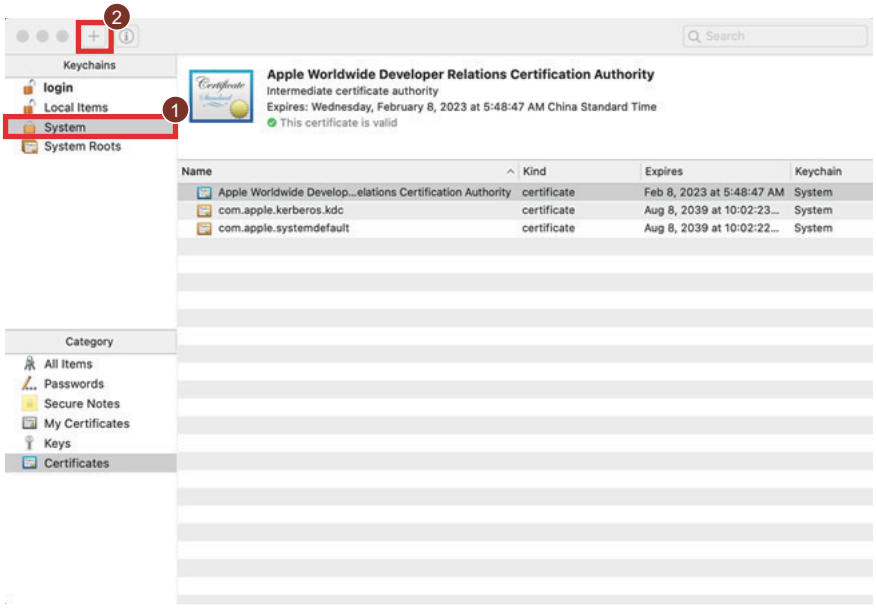
Import the certificate to Firefox

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.

8.2.3 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.

8.2.4 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.

8.3 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 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

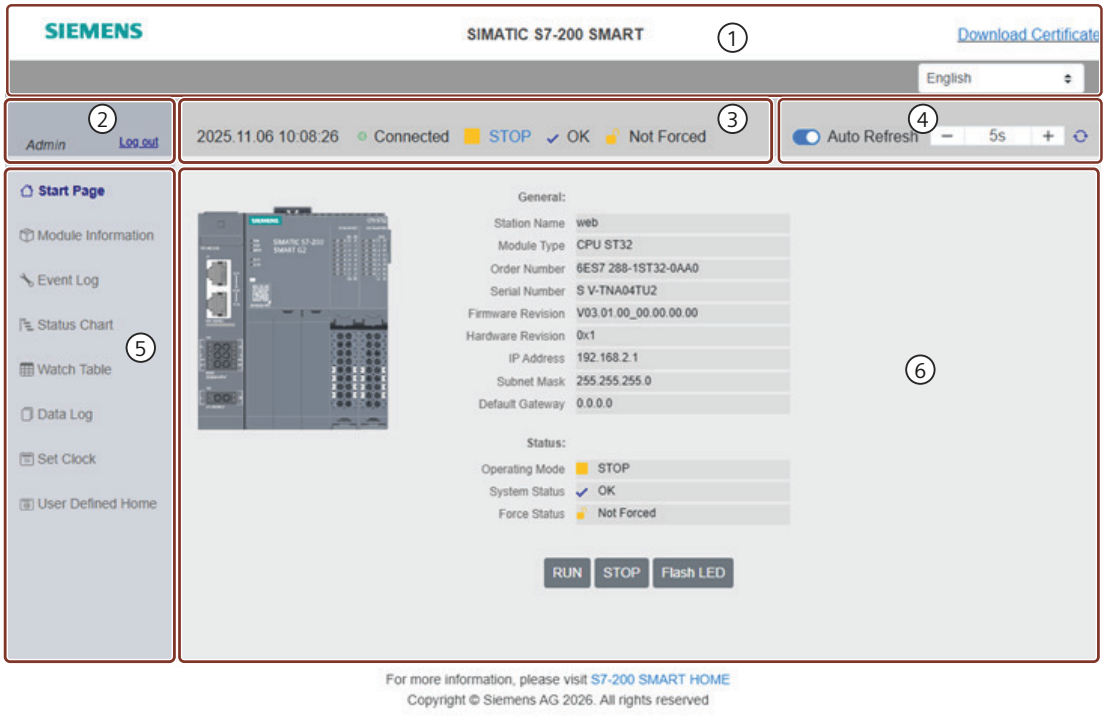
You can access web pages by entering the page URLs. The following table lists the corresponding URL of each page:

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	https: //PLCIP/user_define/

8.4 Standard Web pages

8.4.1 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.

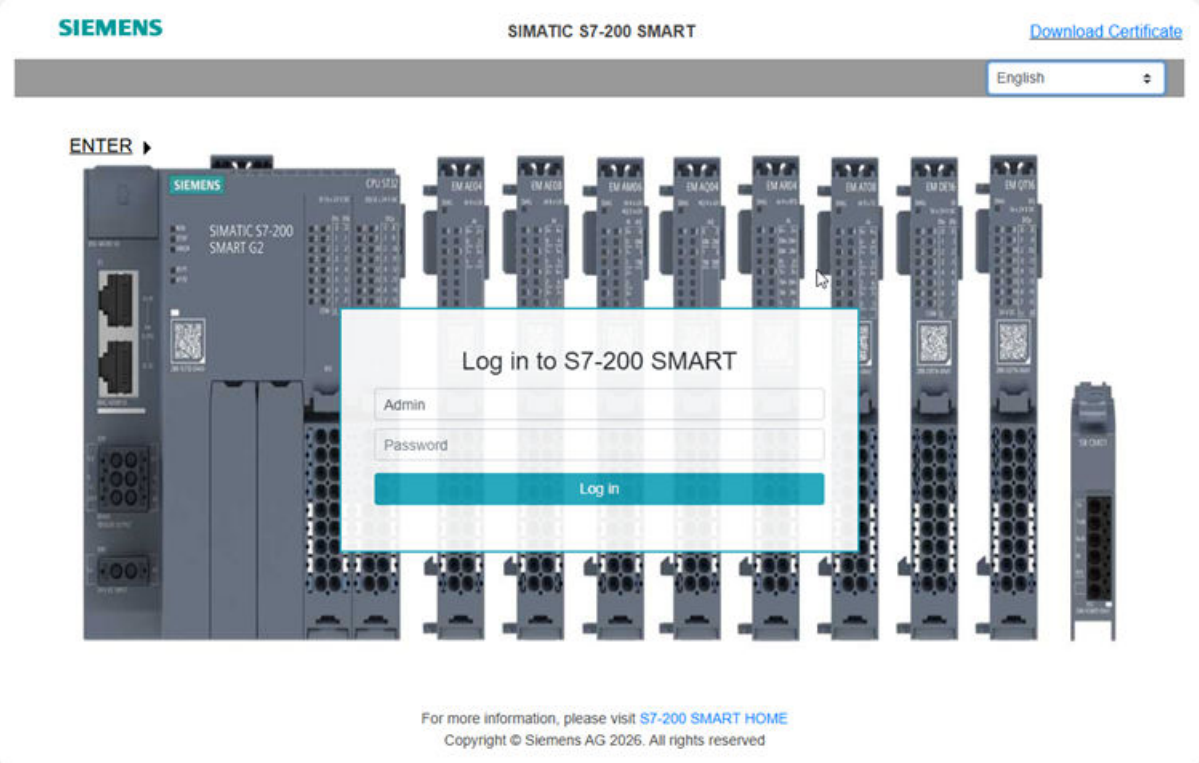
Note

User defined home tab

The user defined home is the entry point to the home page of user defined Web pages.

8.4.2 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.

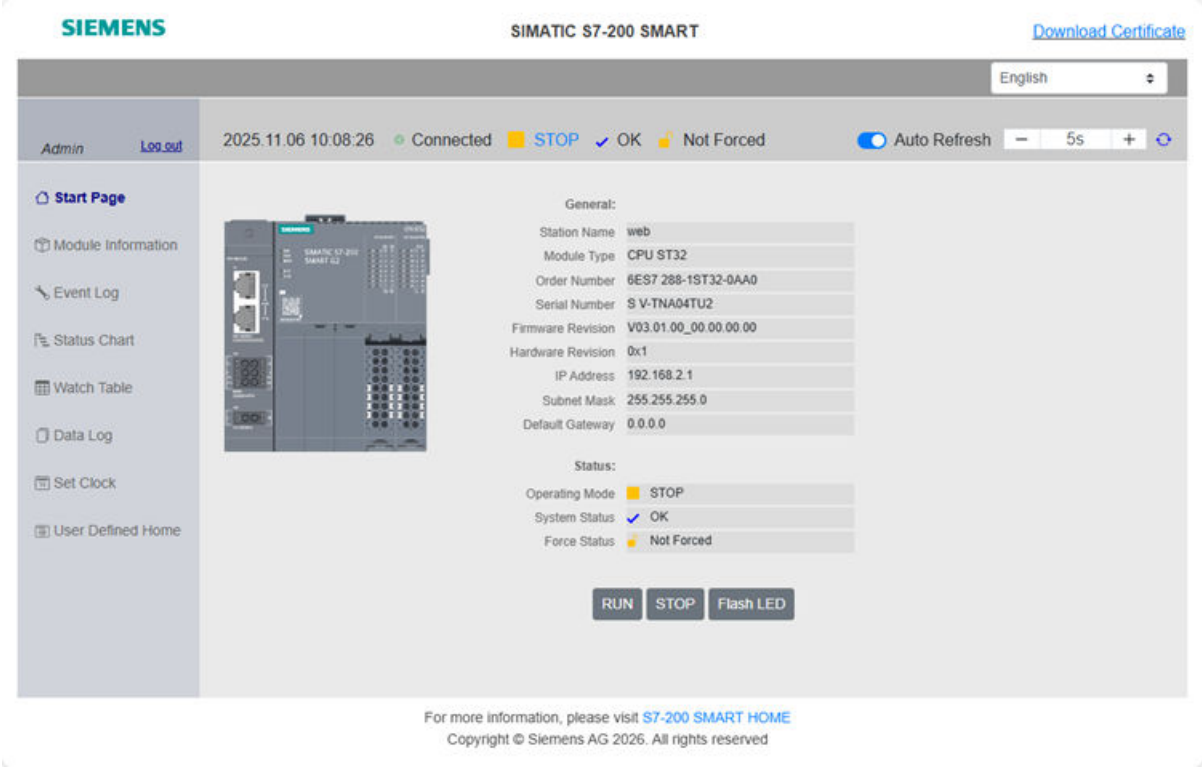
Your login session expires after thirty minutes of inactivity. If you switch on the Auto-refresh, the login session timeout resets, preventing the session from expiring.

8.4.3 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



8.4.4 Event Log

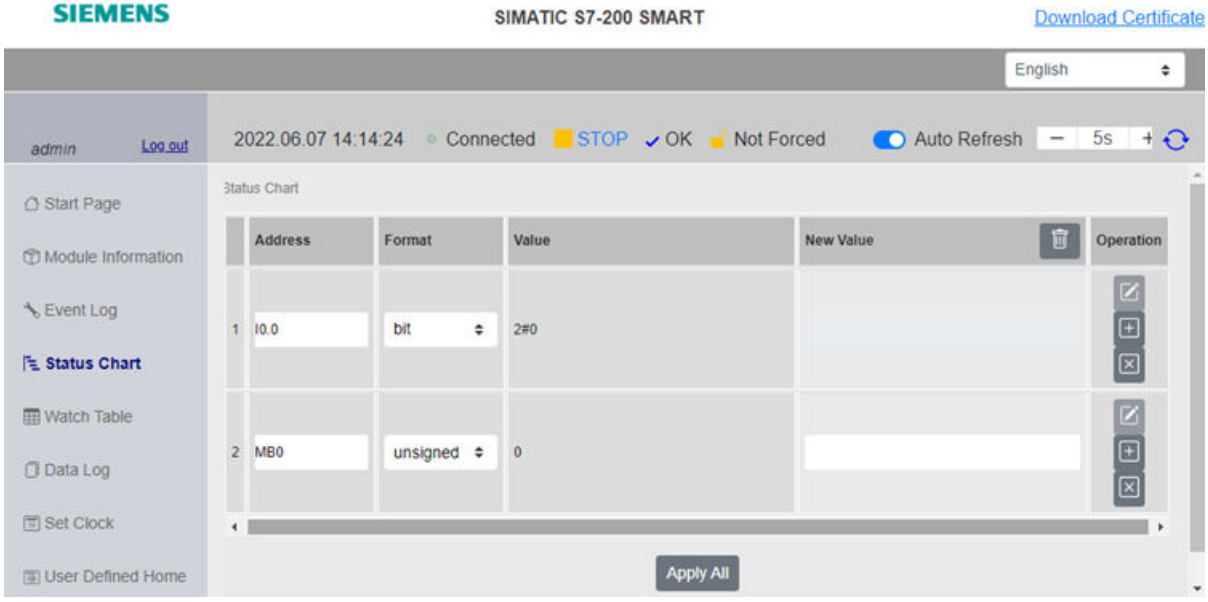
The Event Log page displays a table of time-stamped events. The table lists the time, event type and error or reason information.

The maximum number of the possible event log entries is 128.

You can click "Event log" or the icon to download the event log in a Microsoft Excel spreadsheet.

8.4.5 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

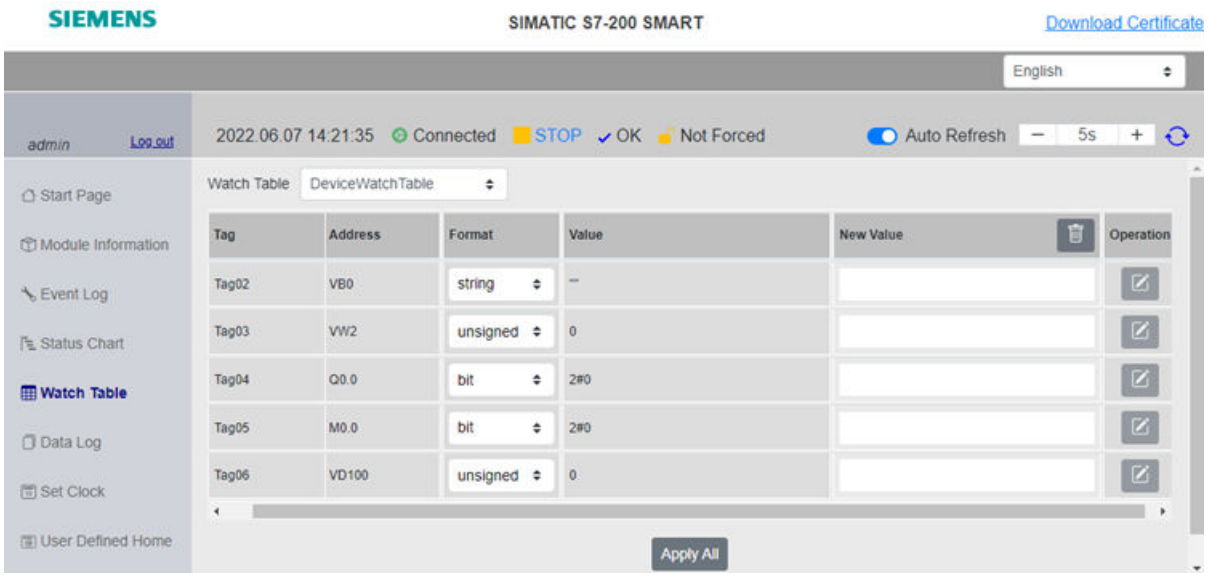
- The following memory areas cannot be written in: I, AI, SM, HC, T bit and C bit.
- If the CPU is in stop mode, applying changes is not supported in the Status Chart page.
- If more than one item in the Status Chart is changed, and there is an error code for one item, applying all changes cannot be successful.

8.4.6 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.

8.4.7 Data log

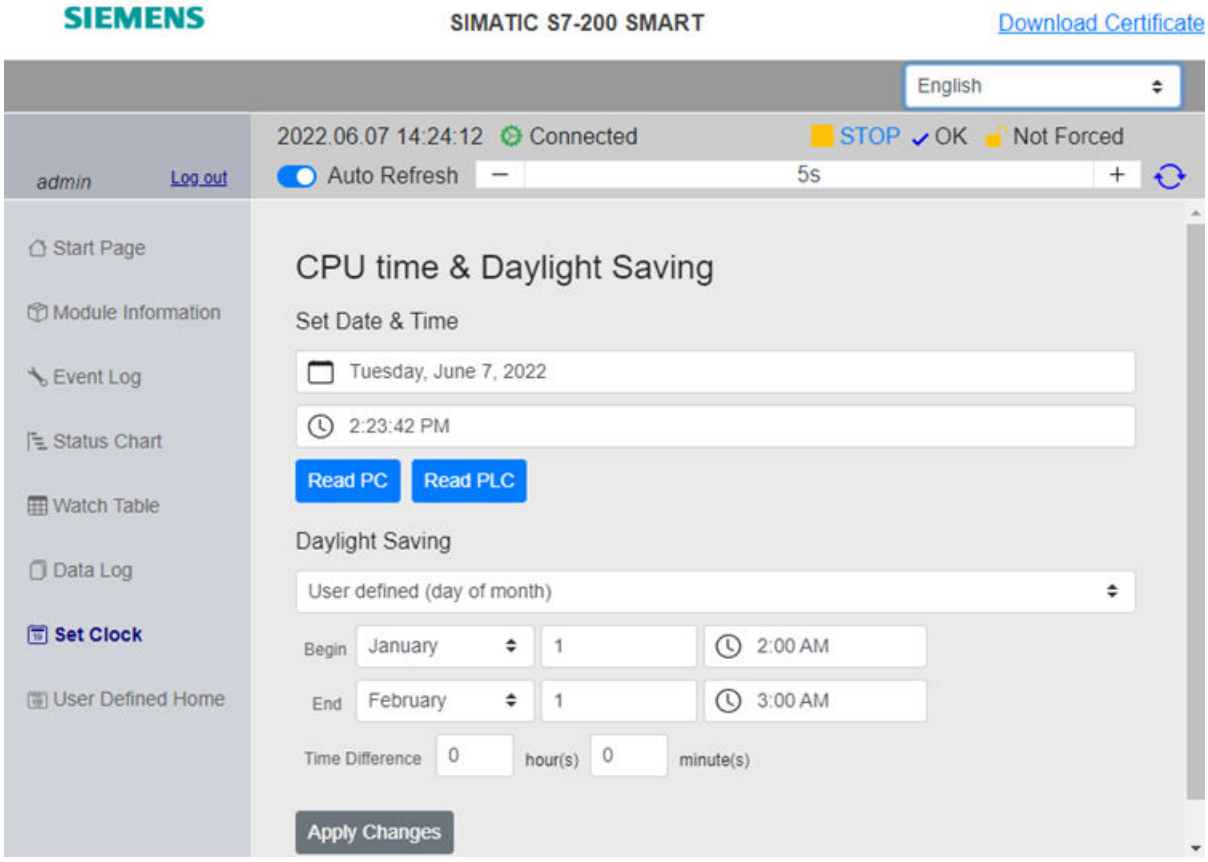
The Data Log page allows you to get access to data log files of the connected CPU.

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.

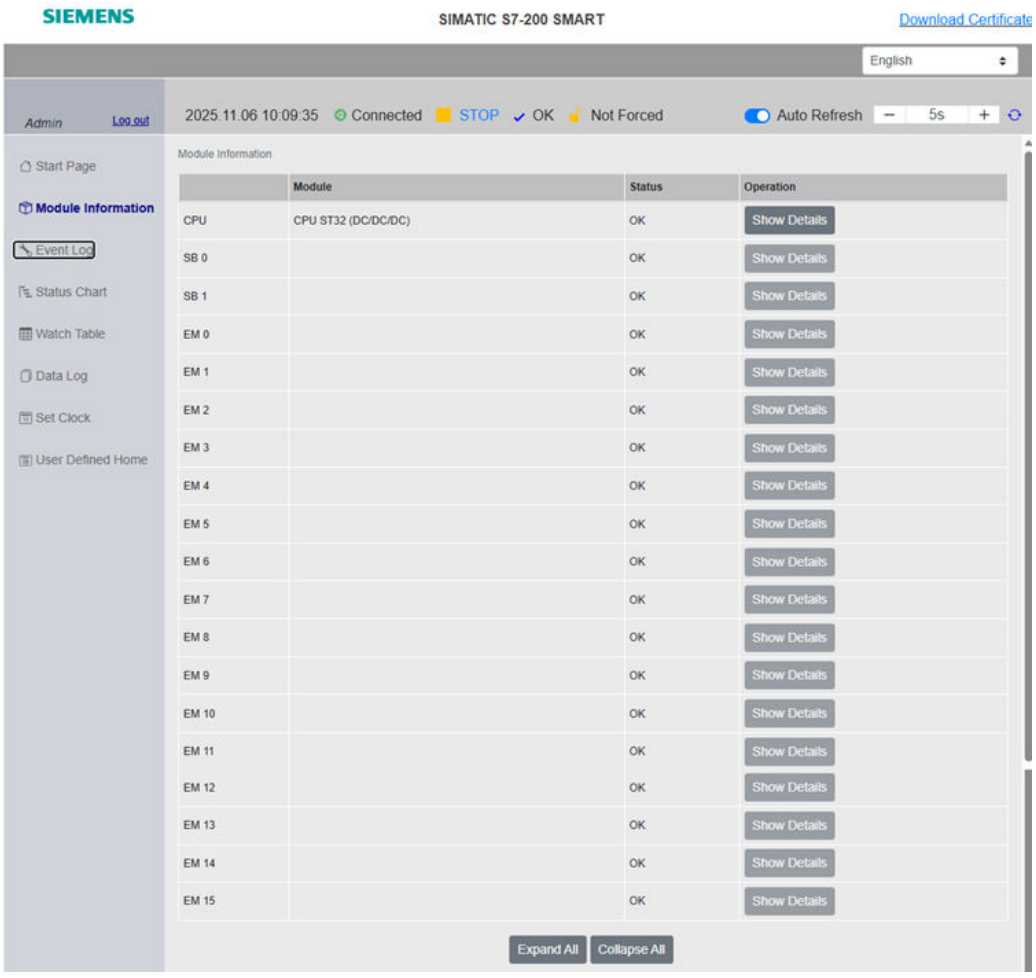
8.4.8 Set clock

The Set Clock page allows you to set the time and date and configure daylight saving time.



8.4.9 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.

8.5 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 250).
3. Configure Web server with Web server wizard.
4. Configure the HTTPS certificates for the Web server with certificate management.
5. Visit user-defined Web pages.
   You can access user defined web pages through URL, 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).

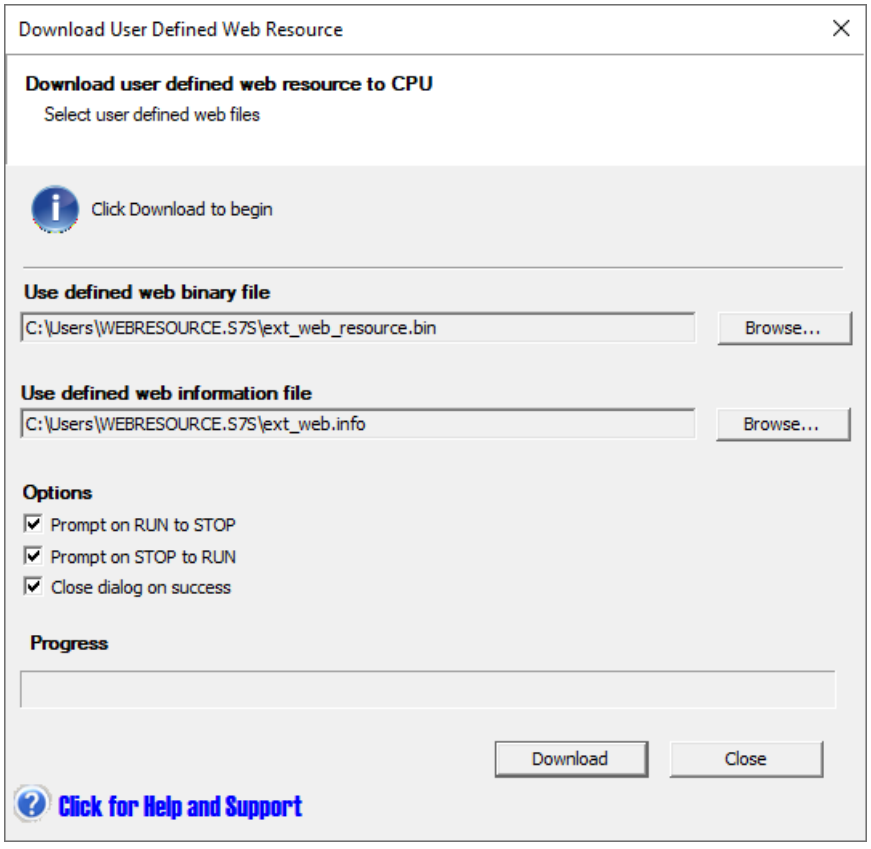
8.5.1 Downloading user-defined Web resource files

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

8.6 Web API (Application Programming Interface)

The CPU offers you a Web-based API (Web API) as an interface for reading and writing CPU data. Through the Web API, you can:

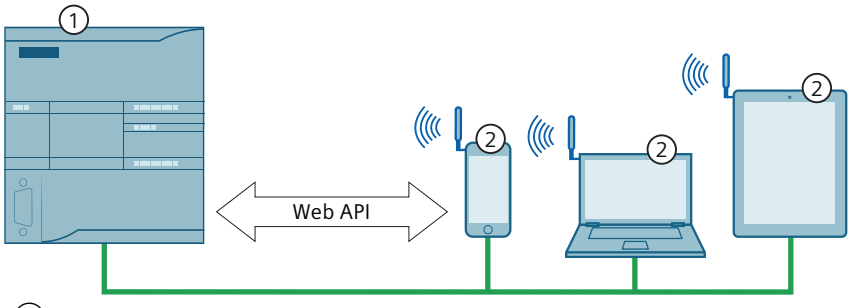
- implement web applications at the latest state-of-the-art technology.
- communicate with the Web server of the CPU via script and programming languages.
- create web applications that connect to multiple CPUs at the same time, for example, to create dashboards that visualize the status of multiple 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.

Instructions	Description
login	log in CPU
logout	log out CPU
get_permission	get a list of authorised actions
read	read the process data of CPU

Instructions	Description
write	write the process data to CPU
browse	browse Web API list supported by CPU

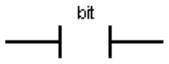
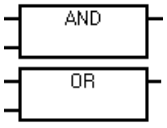
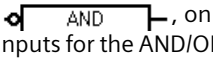
Program instructions

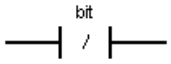
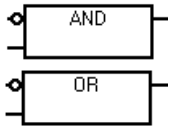
9

9.1 Bit logic

9.1.1 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		-

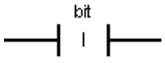
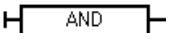
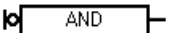
See also

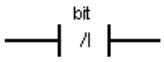
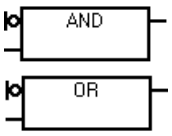
Bit logic input examples (Page 269)

Logic stack overview (Page 257)

9.1.2 Immediate inputs

LAD	FBD	STL	Description

		`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  in front of an input connection.
			The Normally closed immediate instruction is represented by the immediate indicator and negation circle  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.

		`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 269)

Logic stack overview (Page 257)

9.1.3 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.

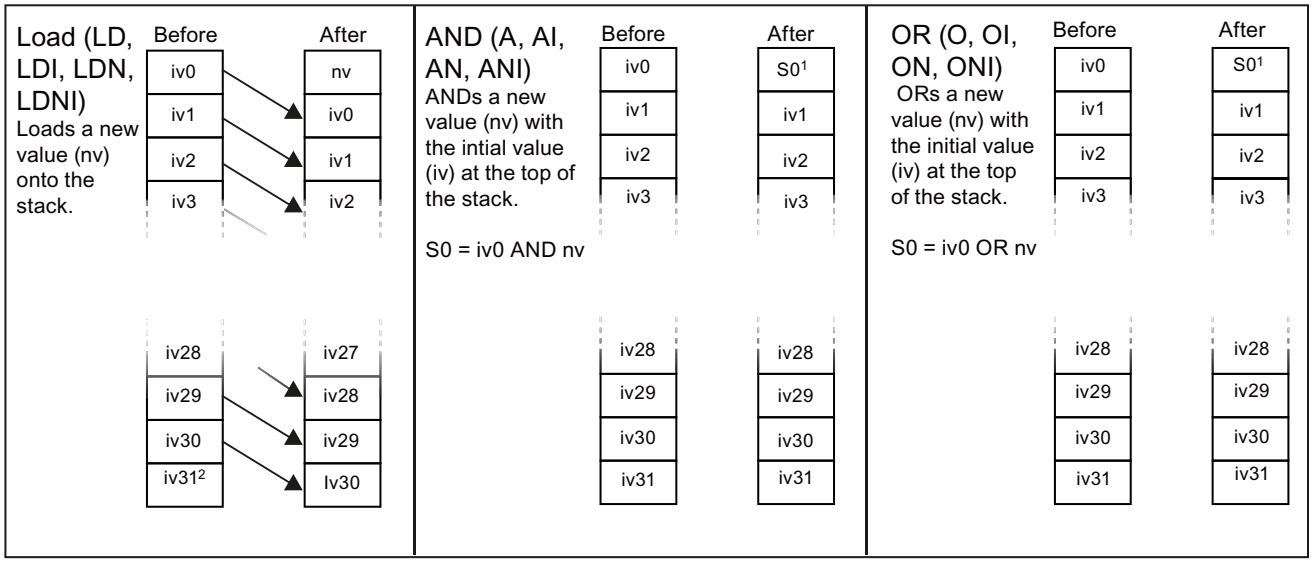
For LAD and FBD editing, the STL logic stack instructions are automatically generated and the programmer does not need to use the logic stack instructions.

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.

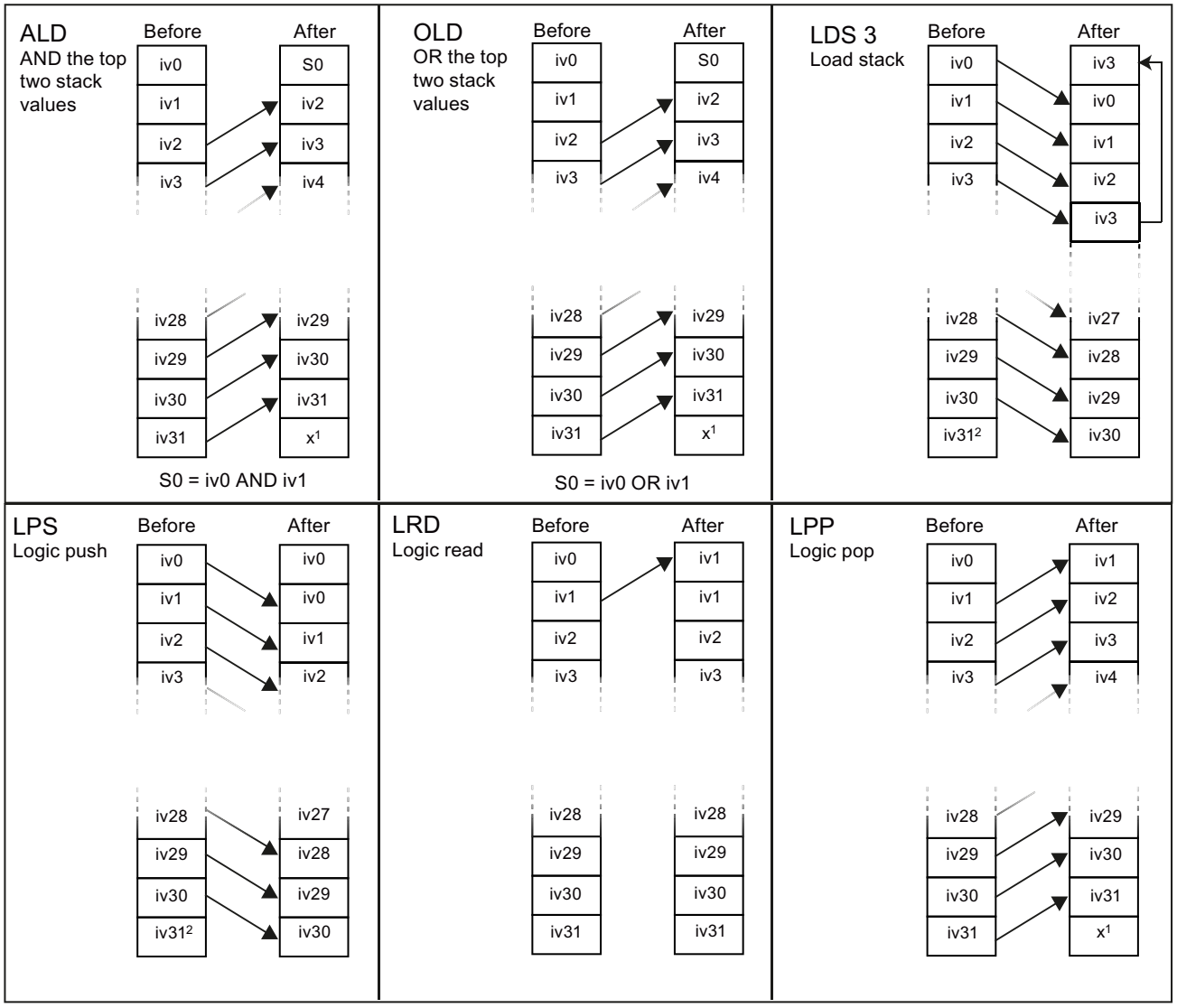
9.1.4 STL logic stack instructions

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

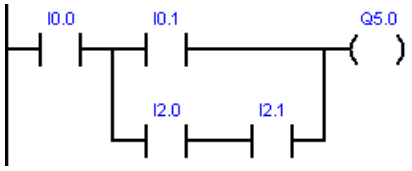
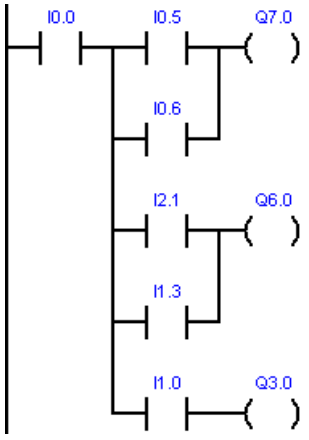
LDS (Load Stack) Input	Data type	Operands
N	BYTE	Constant (0 to 31)

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.

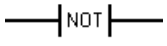
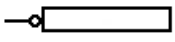


[1] The value is unknown (it could be either a 0 or a 1).
[2] After the execution of a Logic push or a Load stack instruction, value iv31 is lost.

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**

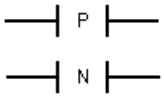
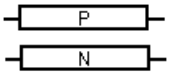
9.1.5 NOT

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 269)

9.1.6 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

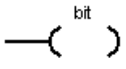
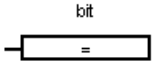
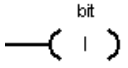
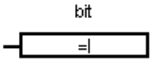
Note

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 269)

9.1.7 Coils: output and output immediate instructions

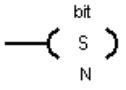
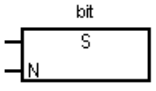
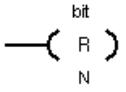
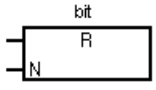
LAD	FBD	STL	Description
		`= bit`	The Output instruction writes the new value for the output bit to the process image register. **LAD** and **FBD**: When the output instruction is executed, the S7-200 SMART turns the output bit in the process image register ON or OFF. The assigned bit is set equal to power flow state. **STL**: The value on the top of the stack is copied to the assigned bit.
		`=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.

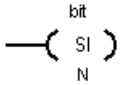
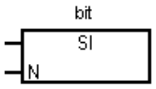
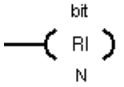
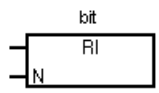
Input / output	Data type	Operand
Bit	BOOL	I, Q, V, M, SM, S, T, C, L
Bit (immediate)	BOOL	Q
Input (LAD)	BOOL	Power flow
Input (FBD)	BOOL	I, Q, V, M, SM, S, T, C, L, Logic flow

See also

Bit logic output examples (Page 270)

9.1.8 Set, reset, set immediate, and reset immediate functions

LAD	FBD	STL	Description
		`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.
		`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

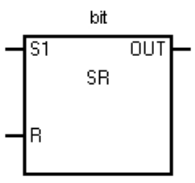
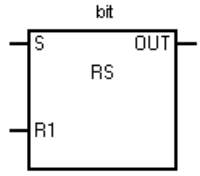
Input / output	Data type	Operand
Bit	BOOL	I, Q, V, M, SM, S, T, C, L
Bit (immediate)	BOOL	Q
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, Constant, *VD, *AC, *LD
Input (LAD)	BOOL	Power flow
Input (FBD)	BOOL	I, Q, V, M, SM, S, T, C, L, Logic flow

See also

Bit logic input examples (Page 269)

Bit logic output examples (Page 270)

9.1.9 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

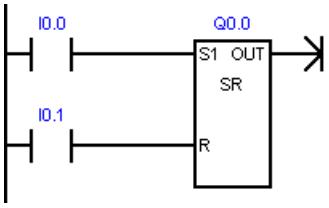
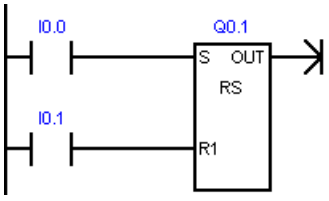
SR truth table

S1	R	Out (bit)
0	0	Previous state
0	1	0
1	0	1
1	1	1

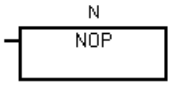
RS truth table

S	R1	Out (bit)
0	0	Previous state
0	1	0
1	0	1
1	1	0

Example SR and RS

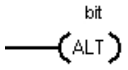
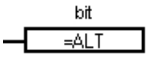
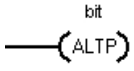
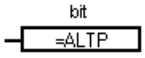
LAD	STL
	**NETWORK 1** **LD I0.0** **LD I0.1** **NOT** **A Q0.0** **OLD** **= Q0.0**
	**NETWORK 2** **LD I0.0** **LD I0.1** **NOT** **LPS** **A Q0.1** **= Q0.1** **LPP** **ALD** **O Q0.1** **= Q0.1**

9.1.10 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)

9.1.11 Alternating output and alternating output on pulse

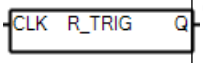
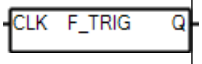
LAD	FBD	STL	Description
		**=ALT bit**	The Alternating output instruction (ALT) continues to invert the state of the output bit when the power flow is valid. If the condition is not met, the output bit remains as it is. **LAD/FBD**: The ALT contact instruction changes the state of output bit when power 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.
		**=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 270)

9.1.12 Edge detector

LAD	STL	Description
	TRIGR	With the "Detect rising edge" instruction, you can detect a state change in the power flow input CLK from "0" to "1". This instruction compares the current value of the input CLK with the last query state (recorded via edge storage bits) stored in the specified instance. When the instruction detects that the state of the input CLK has changed from "0" to "1", it will generate a rising edge signal in the output Q. The output will be "ON" or "1" within one cycle and change the power flow state. In all other cases, the signal state at the output of the instruction is "0", with no power flow output.
	TRIGF	With the "Detect falling edge" instruction, you can detect a state change in the power-flow input CLK from "1" to "0". This instruction compares the current value of the input CLK with the last query state (recorded via edge storage bits) stored in the specified instance. When the instruction detects that the state of the input CLK has changed from "1" to "0", it will generate a rising edge signal in the output Q. The output will be "ON" or "1" within one cycle and change the power flow state. In all other cases, the signal state at the output of the instruction is "0", with no power flow output.

Input/output	Data type	Operands
CLK(FBD)	BOOL	I, Q, V, M, SM, S, T, C, L, Logic Flow
Q(FBD)	BOOL	I, Q, V, M, SM, S, T, C, L, Logic Flow

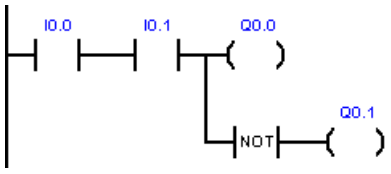
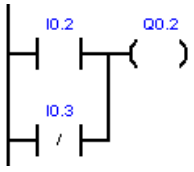
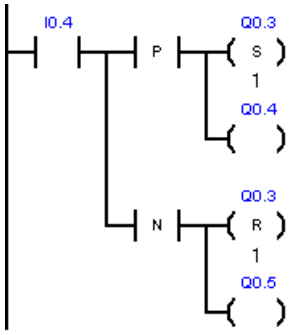
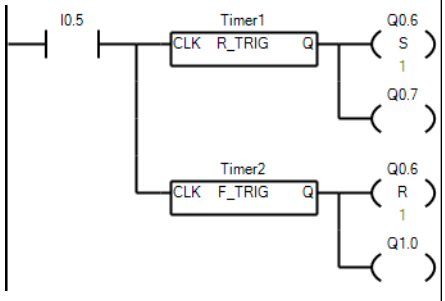
Note

Because the Positive Transition and Negative Transition instructions require an on-to-off or an off-to-on transition, you cannot detect an rising or falling edge 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.

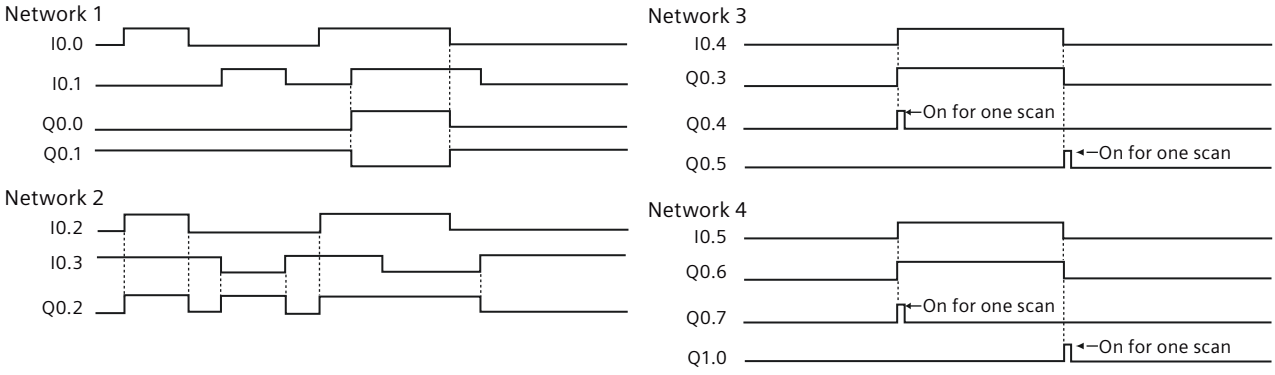
See also

Bit logic input examples (Page 269)

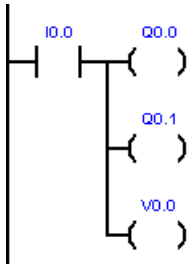
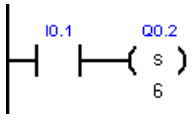
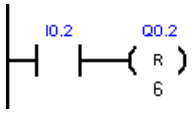
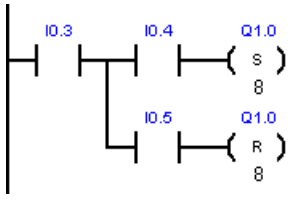
9.1.13 Bit logic input examples

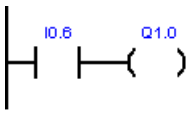
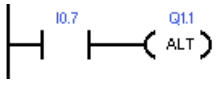
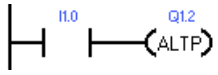
LAD		STL
	Normally-open contacts I0.0 AND I0.1 must be ON (closed) to activate Q0.0. The NOT instruction is used as an inverter. In RUN mode, Q0.0 and Q0.1 have opposite logic states.	**Network 1** **LD I0.0** **A I0.1** **= Q0.0** **NOT** **= Q0.1**
	Normally open contact I0.2 must be closed or normally closed contact I0.3 must be open for Q0.2 to be activated. One or more parallel LAD branches (OR logic) must be true for the output to be activated.	**Network 2** **LD I0.2** **ON I0.3** **= Q0.2**
	A rising edge input on a P contact or a falling edge 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**
	When the CLK of R_TRIG has a rising edge, it will output a pulse that lasts for one scan cycle. When the CLK of F_TRIG has a falling edge, it will output a pulse that lasts for one scan cycle. In RUN mode, the pulsed state changes of Q0.7 and Q1.0 are too fast to be visible in program status view. The Set and Reset outputs latch the pulse state into Q0.6 and make the state change visible in program status view.	**LD I0.5** **LPS** **TRIGR Timer1** **S Q0.6, 1** **= Q0.7** **LPP** **TRIGF Timer2** **R Q0.6, 1** **= Q1.0**

Run-mode timing for input example

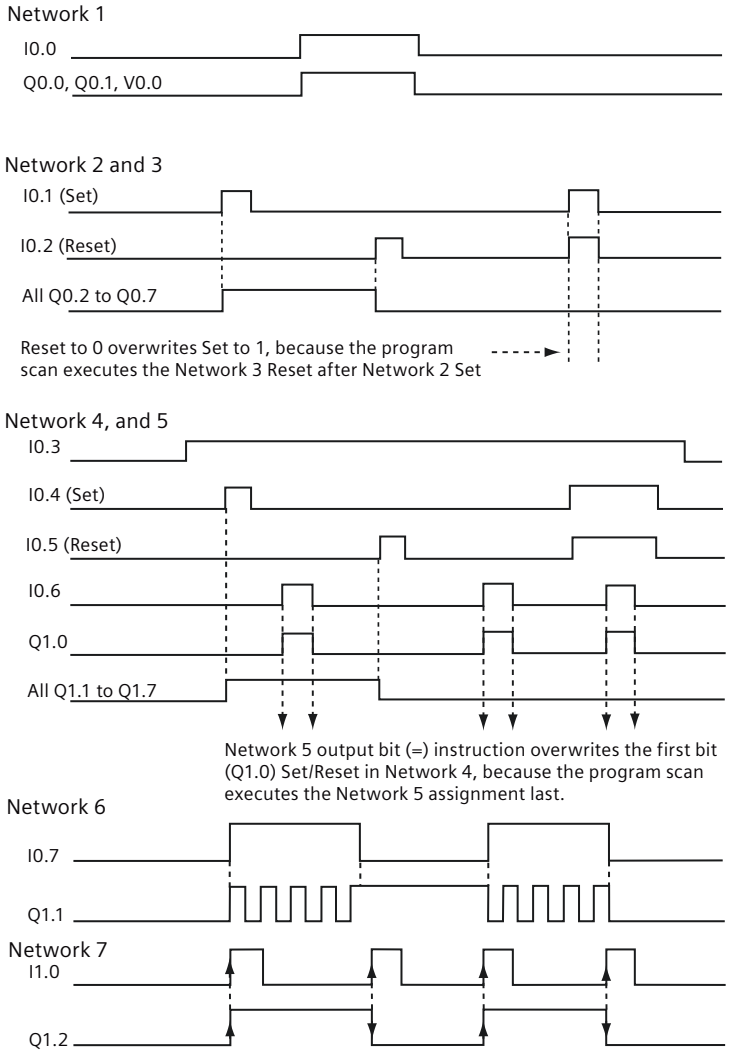


9.1.14 Bit logic output examples

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**

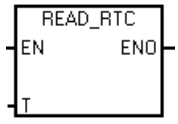
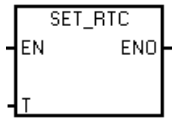
	The Set and Reset instructions perform the function of a latched relay. To isolate the Set/Reset bits, make sure they are not overwritten by another assignment instruction. In this example, Network 4 sets and resets eight output bits (Q1.0 to Q1.7) as a group. In RUN mode, Network 5 can overwrite the 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



9.2 Clock

9.2.1 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.

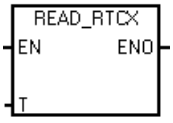
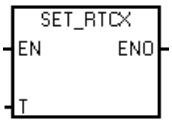
Date	Time	Day of week
01-Jan-2000	00:00:00	Saturday

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.

9.2.2 Read and set real-time clock extended

LAD / FBD	STL	Description
	**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.
	**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.

9.3 Communications

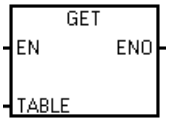
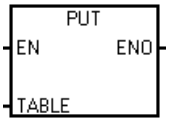
9.3.1 GET and PUT (Ethernet)

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".

Table 9-1 GET and PUT instructions

LAD/FBD	STL	Description
	**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 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") 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 9-2 Valid operands for the GET and PUT instructions

Input/output	Data type	Operand
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 9-3 Definition of GET and PUT instructions TABLE parameter

Byte offset	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
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.

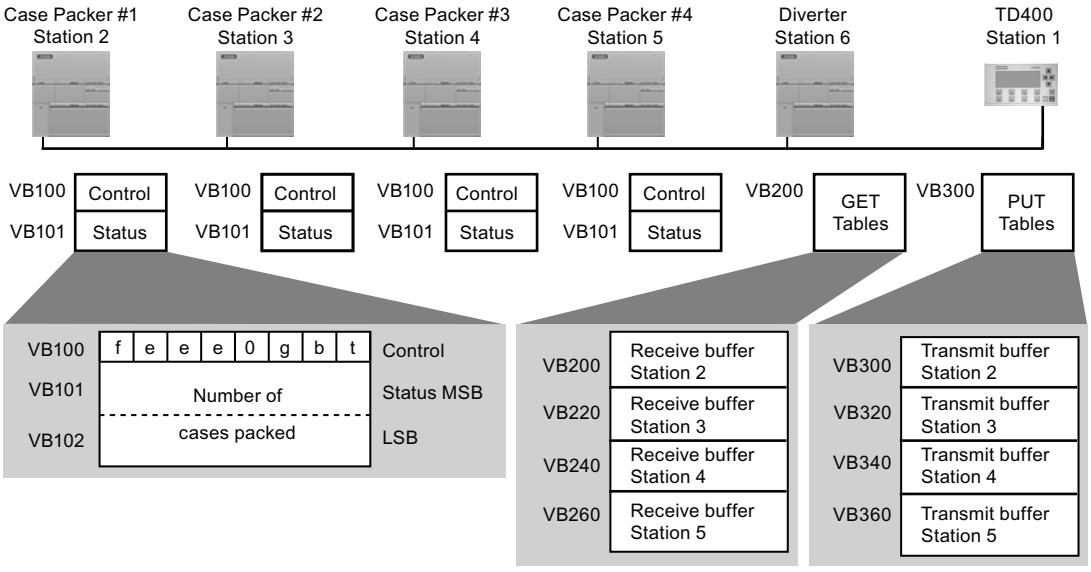
6 Data length: The number of bytes of data that are to be accessed in the remote station (1 to 212 bytes for PUT and 1 to 222 bytes for GET).

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 9-4 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 8 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.

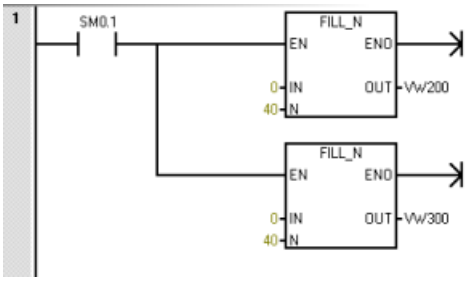
Table 9-5 GET and PUT instructions buffer for reading from and clearing the count of Case Packer 1

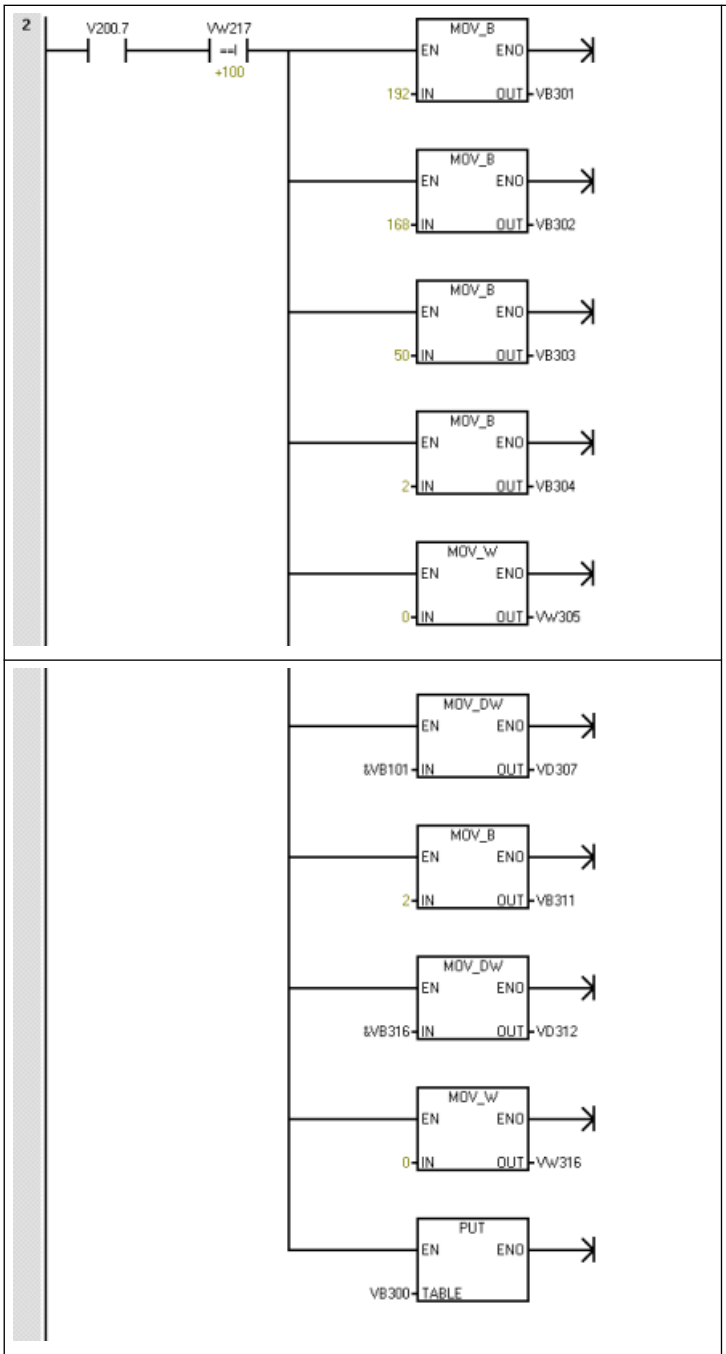
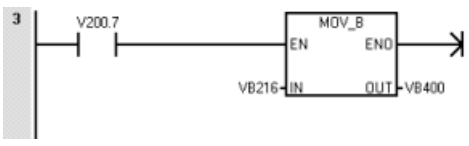
GET_ TABLE buffer	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**PUT_ TABLE buffer**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
VB200	D	A	E	0	Error code				VB300	D	A	E	0	Error code			
VB201	Remote station IP address = 192.								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							

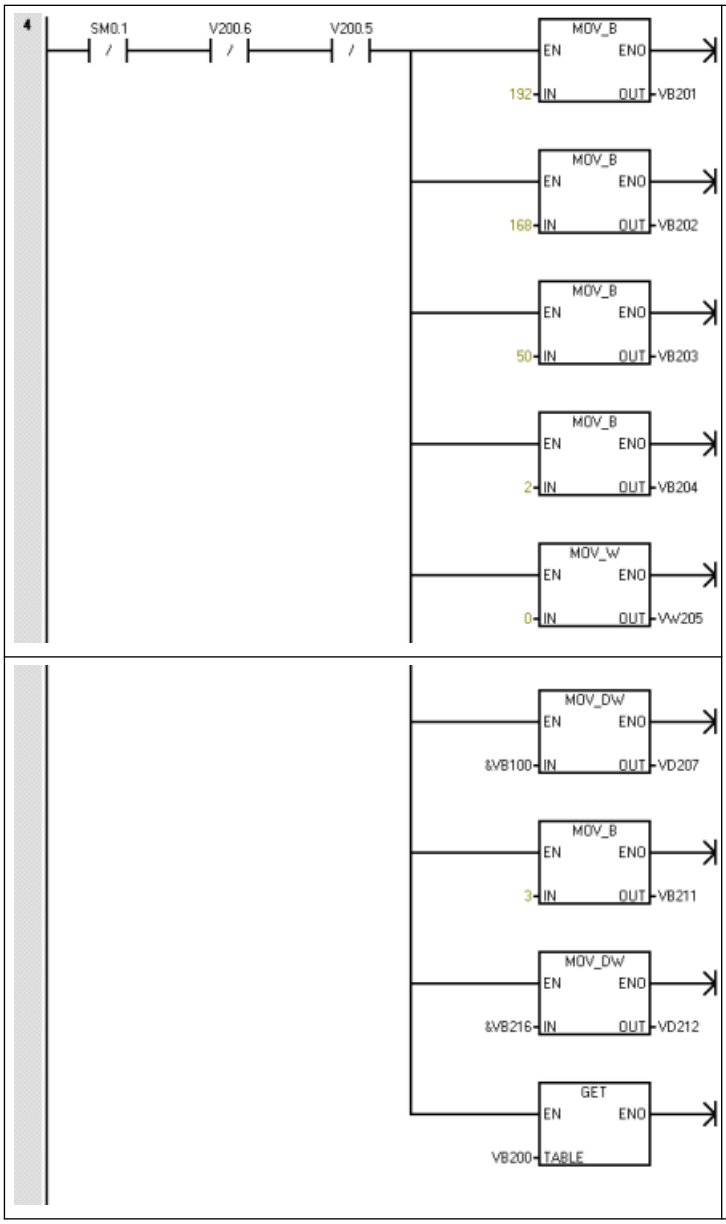
GET_ TABLE buffer	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**PUT_ TABLE buffer**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
VB214	local station (this CPU) =								VB314	local station (this CPU) =							
VB215	(&VB216)								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 9-6 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, VW305** **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.



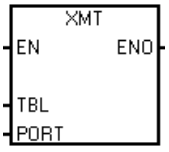
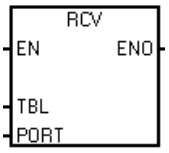
Network 4
LDN SM0.1
AN V200.6
AN V200.5
MOVB 192, VB201
MOVB 168, VB202
MOVB 50, VB203
MOVB 2, VB204
MOVW 0, VW205
MOVD &VB100, VD207
MOVB 3, VB211
MOVD &VB216, VD212
GET VB200

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.

9.3.2 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 TBL, PORT**	The Transmit instruction (XMT) is used in Freeport mode to transmit data by means of the communications port(s).
	**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	Operands
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, SMB130 and 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)

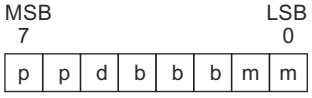
The Freeport mode is disabled and normal communications are re-established (for example, HMI device access) when the CPU is in STOP mode.

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.

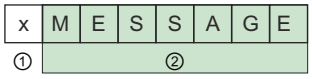


pp	Parity select		d	Data bits per character	
	00 =	No parity		0 =	8 bits per character
	01 =	Even parity		1 =	7 bits per character
	10 =	No parity			
	11 =	Odd			

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.



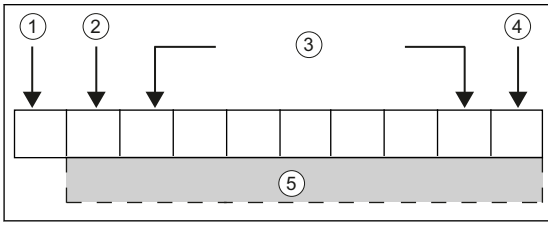
① Number of bytes to transmit
② Characters of the message

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. The operation of sending a BREAK is identical to that of sending any other message. When a BREAK transmission is completed, a transmission interrupt is generated, and SM4.5, SM4.6 or SM20.0 will indicate the current status of the transmission 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.

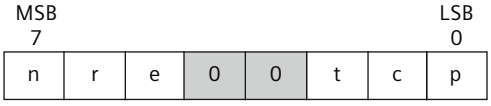
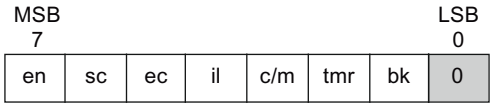
As shown in the following table, the Receive instruction allows you to select the message start and message end conditions, using SMB86 through SMB94 for port 0, SMB186 through SMB194 for port 1 and SMB405 throgh SMB413 for port 2.

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  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  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 receiving if the time period in SMW92 or SMW192 is exceeded. bk: 0 = Ignore break conditions. 1 = Use break condition as start of messagedetection.
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 the 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 timeout 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.

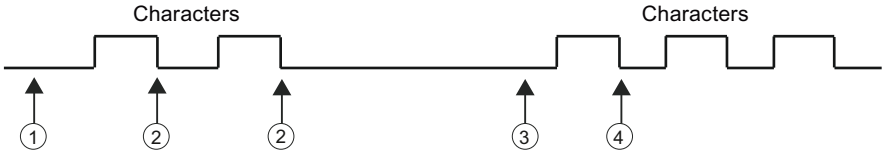
Note

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.
   Settings: il = 1, sc = 0, bk = 0, SMW90/SMW190/SMW409 = idle line timeout in milliseconds



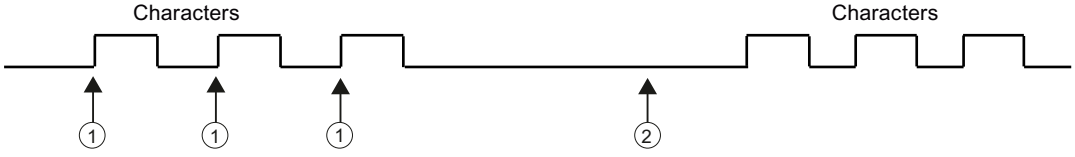
① Receive instruction is executed: Starts the idle time
② Restarts the idle time
③ Idle time is detected: Starts the Receive Message function
④ First character is placed in the message buffer

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.
   Settings: 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.
   Settings: 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.
   Settings: 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.
   Settings: 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.
   Settings: 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.
   Settings: 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

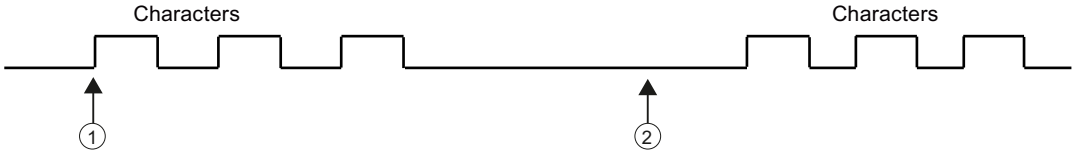
The Receive instruction supports several ways to terminate a message. The message can be terminated on one or a combination of the following:

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.
   Settings: 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.
   Settings: 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.
   Settings: 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

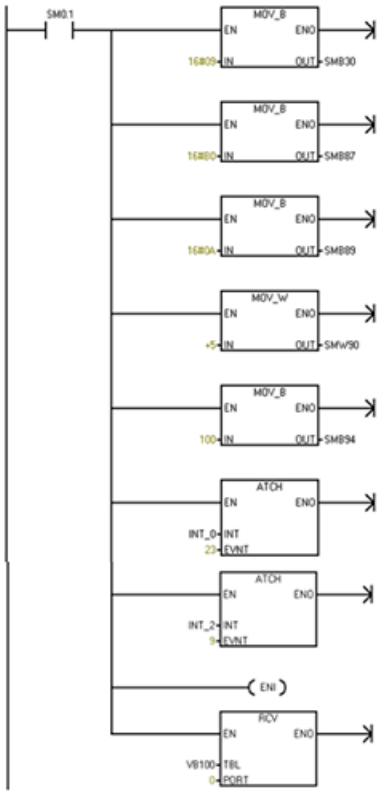
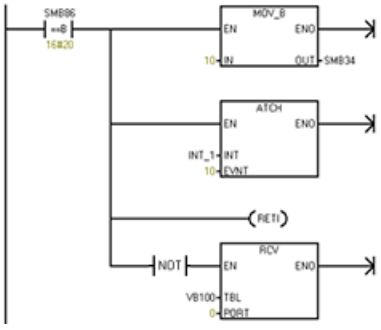
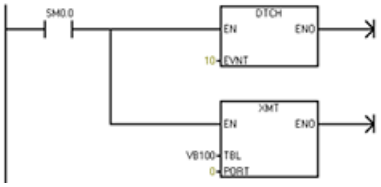
To allow complete flexibility in protocol support, you can also receive data using character interrupt control. Each character received generates an interrupt. The received character is placed in SMB2, and the parity status (if enabled) is placed in SM3.0 just prior to execution of the interrupt routine 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.

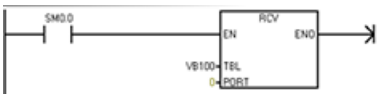
When the character interrupt is used at high baud rates (38.4 Kbps to 115.2 Kbps), the time between interrupts is very short. For example, the character interrupt for 38.4 Kbps is 260 microseconds, for 57.6 Kbps is 173 microseconds, and for 115.2 Kbps is 86 microseconds. Ensure that you keep the interrupt 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

<table>
<tr><td>MAIN</td><td>Network 1</td><td colspan="2">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.</td></tr>
<tr><td></td><td rowspan="10"> 
</td><td>LD SM0.1
MOVB 16#09, SMB30</td><td>On the first scan:
1. Initialize the free port:
- Select 9600 baud.
- Select 8 data bits.
- Select no parity.</td></tr>
<tr><td>MOVB 16#B0, SMB87</td><td>2. Initialize the RCV message control byte:
- Enable RCV.
- Detect message end character.
- Detects whether the line idle condition is used as the message start condition.</td></tr>
<tr><td>MOVB 16#0A, SMB89</td><td>3. Set end of message character to hex 0A (line feed).</td></tr>
<tr><td>MOVW +5, SMW90</td><td>4. Set idle line timeout to 5 ms.</td></tr>
<tr><td>MOVB 100, SMB94</td><td>5. Set maximum number of characters to 100.</td></tr>
<tr><td>ATCH INT_0, 23</td><td>6. Attach interrupt 0 to the Receive Complete event.</td></tr>
<tr><td>ATCH INT_2, 9</td><td>7. Attach interrupt 2 to the Transmit Complete event.</td></tr>
<tr><td>ENI</td><td>8. Enable user interrupts.</td></tr>
<tr><td>RCV VB100, 0</td><td>9. Enable receive box with buffer at VB100.</td></tr>
<tr><td>INT 0</td><td>Network 1</td><td colspan="2">Network 1</td></tr>
<tr><td></td><td> 
</td><td>LDB= SMB86, 16#20
MOVB 10, SMB34
ATCH INT_1, 10
CRETI
NOT
RCV VB100, 0</td><td>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.</td></tr>
<tr><td>INT 1</td><td>Network 1</td><td colspan="2">Network 1</td></tr>
<tr><td></td><td> 
</td><td>LD SM0.0
DTCH 10
XMT VB100, 0</td><td>10-ms Timer interrupt:
1. Detach timer interrupt.
2. Transmit message back to user on port.</td></tr>
</table>

INT 2	Network 1	Network 1	
		LD SM0.0 RCV VB100, 0	Transmit Complete interrupt: Enable another receive.

9.3.3 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.

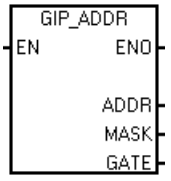
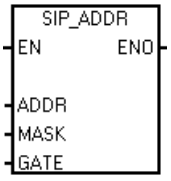
LAD / FBD	STL	Description
	**GPA ADDR, PORT**	The GET_ADDR instruction reads the station address of the CPU port specified in PORT and places the value in the address specified in ADDR.
	**SPA ADDR, PORT**	The SET_ADDR instruction sets the port station address (PORT) to the value specified in ADDR. The 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)

9.3.4 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":

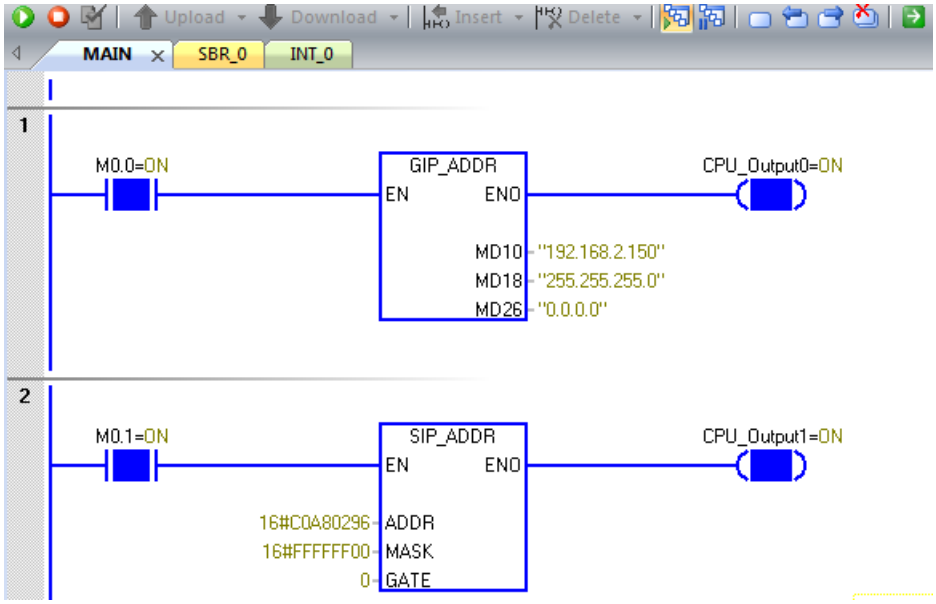
Octet decimal value	Hexadecimal value
255	FF
255	FF
255	FF
0	00

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.



9.3.5 Open user communication

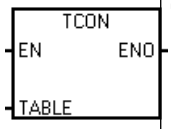
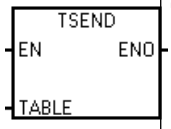
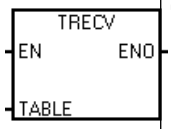
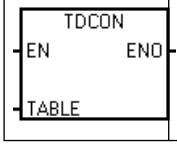
The Open User Communication (OUC) instructions give your program a way to communicate over an Ethernet network to another Ethernet capable device. The other Ethernet device can be another S7-200 SMART CPU or another third party device that supports either UDP, TCP, or ISO-on-TCP protocol. Your program controls all aspects of the communication from selecting the protocol, initiating the connection, sending data, receiving data, and terminating the connection.

9.3.5.1 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 9-7 OUC instructions

LAD/FBD	STL	Description
	**TCON table**	TCON initiates a UDP, TCP, or ISO-on-TCP communications connection from the CPU to a communication partner.
	**TSEND table**	TSEND sends data to another device.
	**TRECV table**	TRECV retrieves data over an existing communication connection.
	**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" 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" for further information.

Table 9-8 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 311) for the error code listing. Refer to "Ports and TSAPs" 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 311).
- 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 9-9 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 IP Address [3]							
6								
7								
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 9-10 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 (2 bytes)							
3								
4	Connection Type [2]							
5	Remote IP Address [3]							
6								
7								
8								
9	Remote TSAP [4] String of 2 to 16 characters (3 to 17 bytes)							
to								
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 311) for the error code listing. Refer to "Ports and TSAPs" 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 311).
- 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 9-11 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 9-12 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 311) for the error code listing. Refer to "Ports and TSAPs" 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 311).
- 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 9-13 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								

9.3.5.2 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

9.4 Compare

9.4.1 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

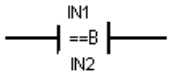
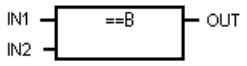
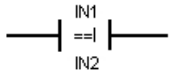
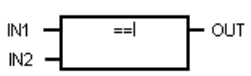
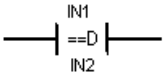
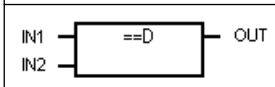
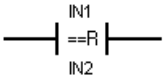
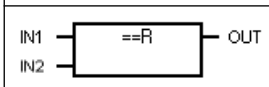
Six comparison types are available:

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

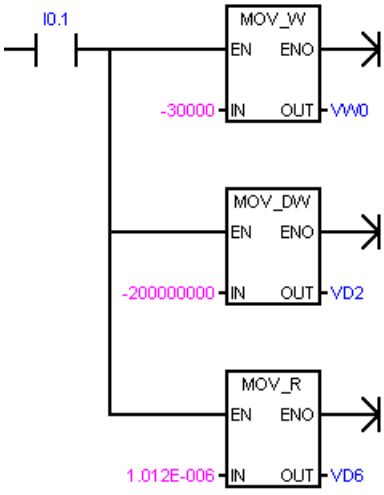
- Illegal indirect address is encountered (any compare instruction)
- Illegal real number (for example, NaN) is encountered for compare real instruction

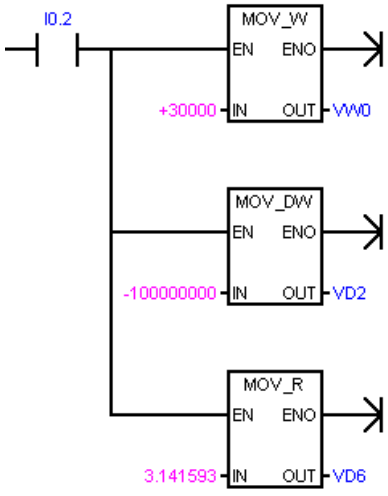
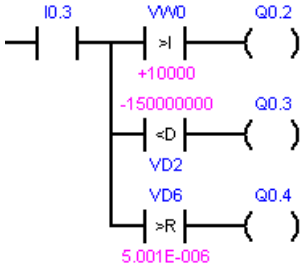
To prevent these conditions from occurring, ensure that you properly initialize pointers and values that contain real numbers before executing compare instructions that use these values.

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

9.4.2 Compare character strings

The compare string instructions can compare two ASCII character strings.

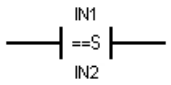
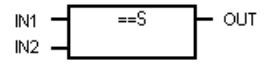
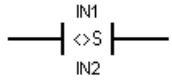
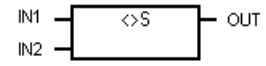
For **LAD** and **FBD:** When the comparison is TRUE, the compare instruction turns ON the contact (LAD), or output (FBD).

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.

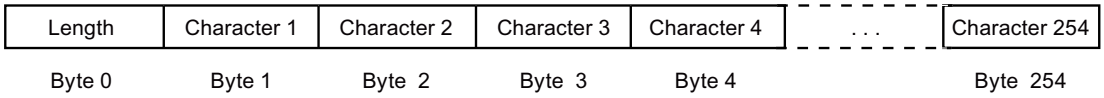
Compare instructions are executed regardless of the state of power flow.

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).



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.

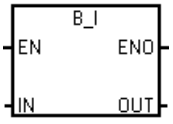
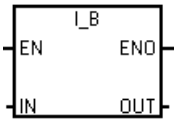
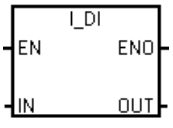
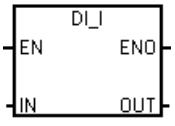
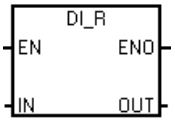
See also Constants

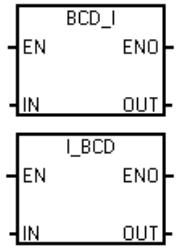
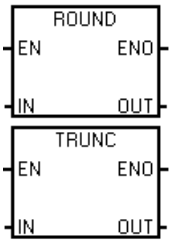
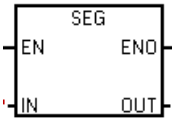
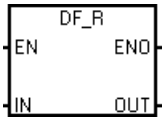
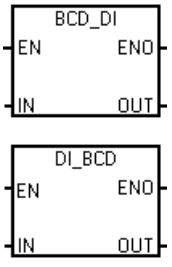
9.5 Convert

9.5.1 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.

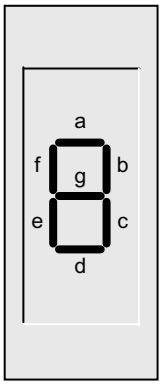
LAD / FBD	STL	Description
	**BCDI OUT** **IBCD OUT**	**BCD to Integer**: Convert the binary-coded decimal WORD data type value IN to an integer WORD data type value and load the result in the address assigned to OUT. The valid range for IN is 0 to 9999 BCD. **Integer to BCD**: Convert the input integer 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 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 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.
	**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.
	**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

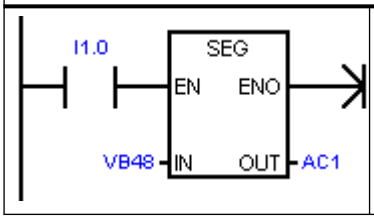
(IN) LSD	Segment Display	(OUT) - gfe dcba
0	0	0011 1111
1	1	0000 0110
2	2	0101 1011
3	3	0100 1111
4	4	0110 0110
5	5	0110 1101
6	6	0111 1101
7	7	0000 0111



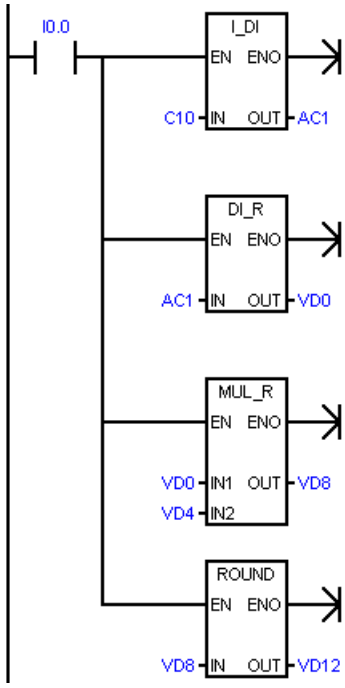
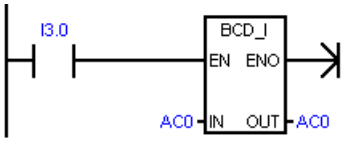
(IN) LSD	Segment Display	(OUT) - gfe dcba
8	8	0111 1111
9	9	0110 0111
A	A	0111 0111
B	b	0111 1100
C	C	0011 1001
D	d	0101 1110
E	E	0111 1001
F	F	0111 0001

Example: Using SEG to display the numeral 5 on a seven-segment display

LAD		STL

		`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

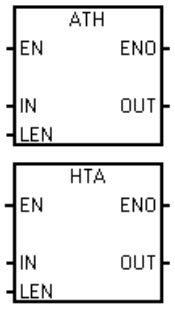
9.5.2 ASCII character array conversion

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

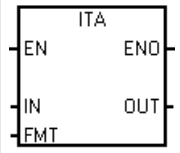
Input / output	Data type	Operand
IN, OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC
LEN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant

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 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

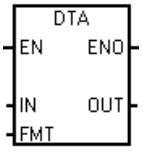
MSB LSB

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 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

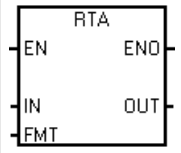
FMT

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

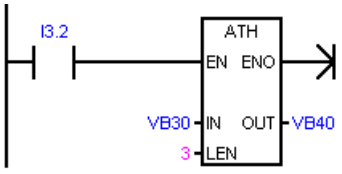
FMT

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

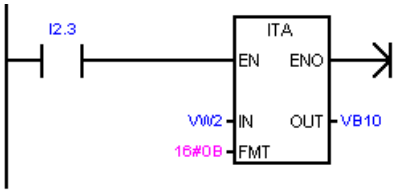
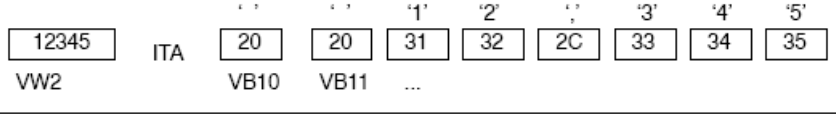
	Out	Out +1	Out +2	Out +3	Out +4	Out +5
in = 1234.5	1	2	3	4	.	5
in = -0.0004				0	.	0
in = -3.67526			-	3	.	7
in = 1.95				2	.	0

Example: ASCII to Hexadecimal

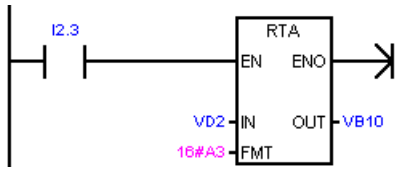
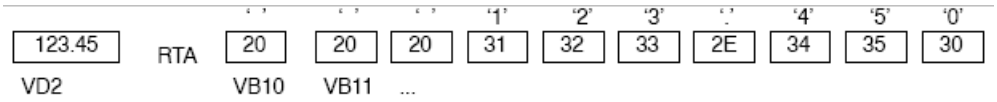
LAD	STL
	**Network 1** **LD I3.2** **ATH VB30, VB40, 3**
	

[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

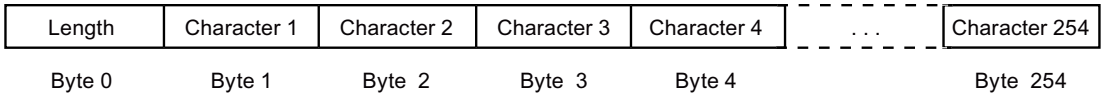
Assigning a constant value for instructions

9.5.3 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).



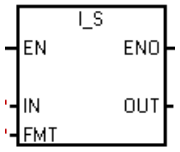
If a constant string parameter is entered directly in the program editor (126 characters maximum) or a variable string is initialized in the data block editor (254 characters maximum), the string assignment must begin and end with double quote characters.

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
	**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

9.5.4 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

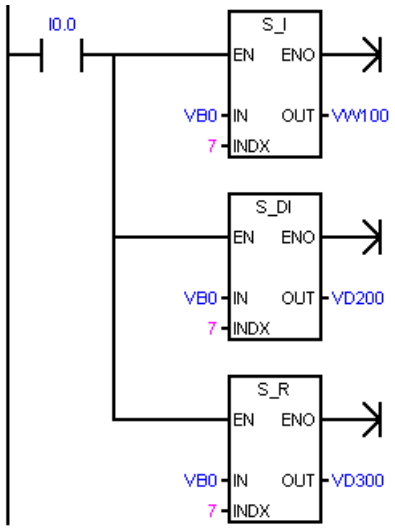
Valid Input Strings
for Real Numbers

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

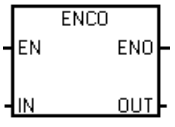
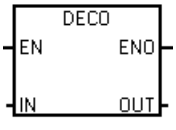
11	'T'	'e'	m'	'p'	' '	' '	'9'	'8'	'.'	'6'	'F'

After executing the network:
VW100 (integer) = 98
VD200 (double integer) = 98
VD300 (real) = 98.6

See also

Assigning a constant value for instructions

9.5.5 Encode and decode

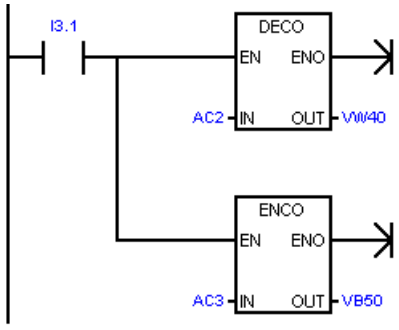
LAD / FBD	STL	Description
	**ENCO IN, OUT**	Encode writes the bit number of the least significant bit set in the input word IN, to the least significant "nibble" (4 bits) of the output byte OUT.
	**DECO IN, OUT**	Decode sets the bit in the output word OUT that corresponds to the bit number represented 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



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.

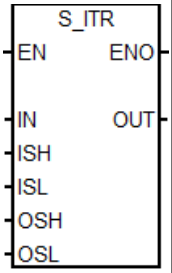
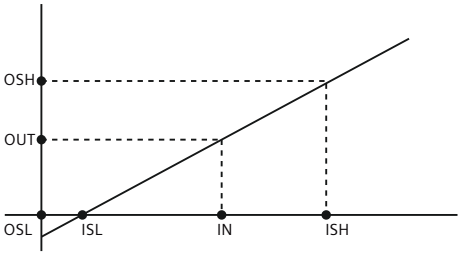
STL

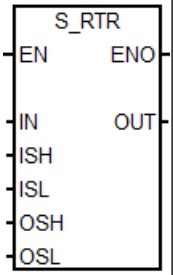
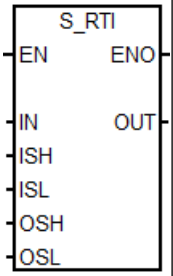
```
Network 1
LD I3.1
DECO AC2, VW40
ENCO AC3, VB50
```



9.5.6 Scale

You can use the "Scale" instruction to scale the value at the IN input by mapping it to a specified value range.

<table>
<tr><th>LAD / FBD</th><th>STL</th><th>Description</th><th></th></tr>
<tr><td> 
</td><td>SITR IN, ISH, ISL, OSH, OSL, OUT</td><td>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.</td><td rowspan="3">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</td></tr>
<tr><td> 
</td><td>SRTR IN, ISH, ISL, OSH, OSL, OUT</td><td>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.</td></tr>
<tr><td> 
</td><td>SRTI IN, ISH, ISL, OSH, OSL, OUT</td><td>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.</td></tr>
</table>

Non-fatal errors with ENO = 0

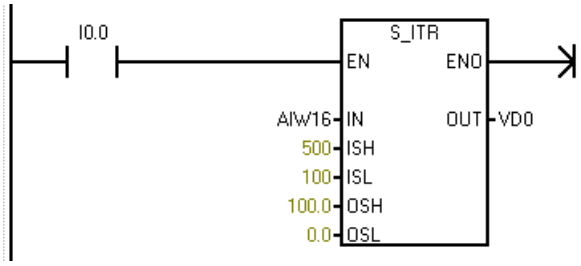
- 0X0090: invalid parameter, when (ISH ≤ ISL) or (OSH ≤ OSL) or IN is out of range [ISL, ISH]).

	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



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.

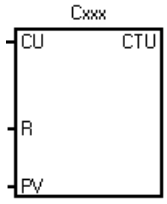
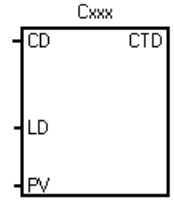
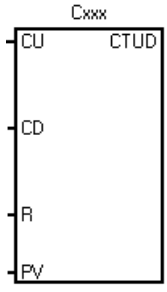
STL

```
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

9.6 Counters

9.6.1 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.

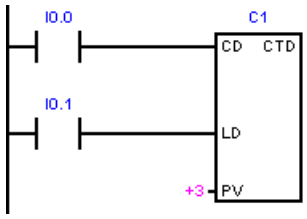
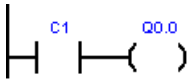
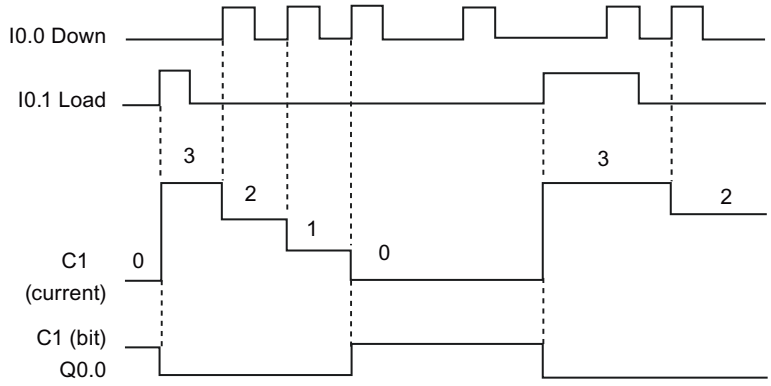
See also Configuring the retentive ranges - system block configuration

Counter operation

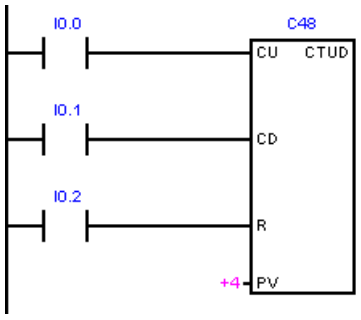
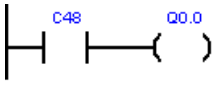
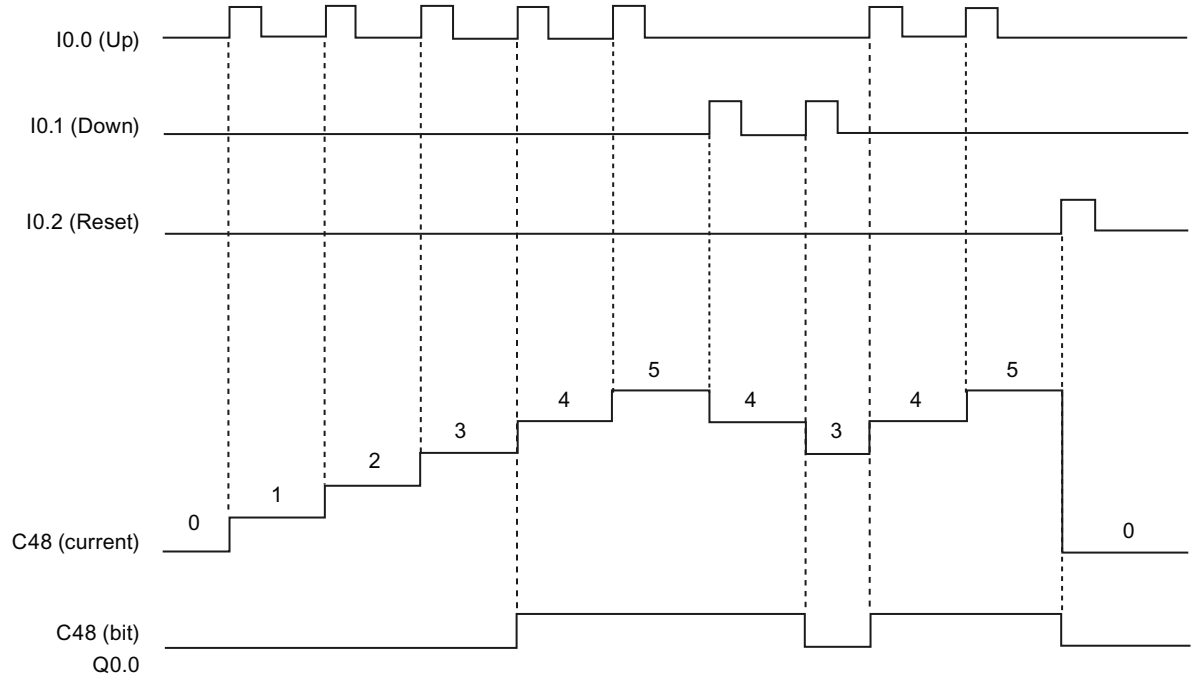
Type	Operation	Counter bit	Power cycle / first scan
CTU	• CU increments the current value. • Current value continues to increment until it reaches 32,767.	The counter bit is set ON when: Current value >= Preset	• Counter bit is OFF. • Current value can be retained [1]
CTD	• CD decrements the current 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.

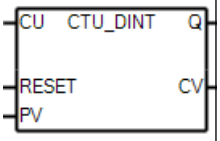
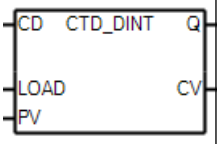
Example CTD count down

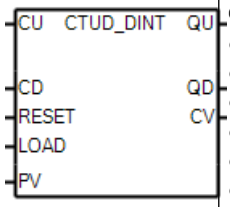
<table>
<tr><th>LAD</th><th></th><th>STL</th></tr>
<tr><td> 
</td><td>Count down counter C1 current value counts from 3 to 0
With I0.1 OFF, I0.0 OFF-ON decrements C1 current value
I0.1 ON loads countdown preset value 3</td><td>Network 1
LD I0.0
LD I0.1
CTD C1, +3</td></tr>
<tr><td> 
</td><td>C1 bit is ON when counter C1 current value = 0</td><td>Network 2
LD C1
= Q0.0</td></tr>
<tr><td colspan="3">Timing diagram
 
</td></tr>
</table>

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 		

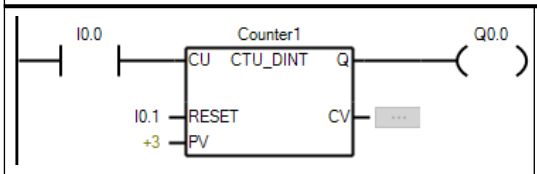
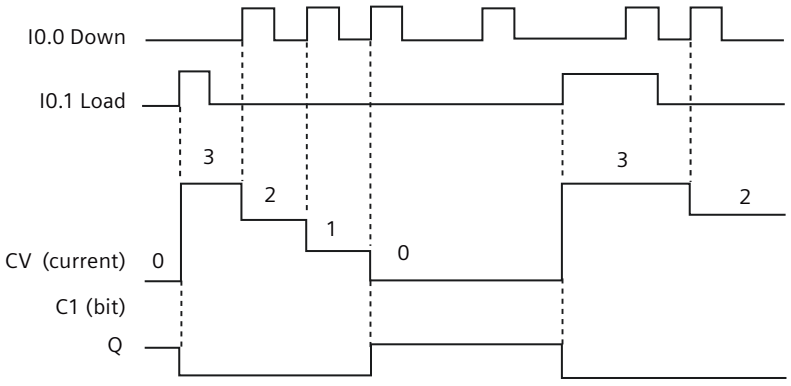
9.6.2 Counter instructions (IEC)

LAD/FBD	STL	Description
	**CTUDINT ctu1, ctu1.RESET, ctu1.PV, ctu1.CV**	**LAD/FBD:** The "increment counter" instruction is used to increment the value of the CV parameter. When the signal state of the parameter CU changes from "0" to "1" (rising edge), the instruction is executed and the current counter value of the parameter CV is incremented by one. The counter value is incremented each time a rising edge is detected, until it reaches the high limit for the data type specified at the CV output. When the high limit is reached, the signal state of the CU input no longer has an effect on the instruction. You can query the count status of the Q parameter. The signal state of parameter Q is determined by parameter PV. If the current counter value is greater than or equal to the value of parameter PV, the signal state of parameter Q is set to "1". In all other cases, the signal state of the Q parameter is "0". When the signal state of the RESET parameter changes to "1", the value of the CV parameter is reset to "0". As long as the signal state of the RESET parameter is "1", the signal state of the CU parameter has no effect on the instruction. **STL**: RESET input is the top of stack value**.** CU count up input is loaded in the second stack level.
	**CTDDINT ctd1, ctd1.LOAD, ctd1.PV, ctd1.CV**	**LAD/FBD:** The "Count down" instruction is used to decrement the value at the CV parameter. When the signal state of the parameter CD changes from "0" to "1" (rising edge), the instruction is executed and the current counter value of the parameter CV is decremented by 1. Each time a rising edge is detected, the counter value is decremented by 1 until it reaches the low limit of the specified data type, or until it reaches 0. When the 0 limit is reached, the signal state of the CD input no longer has an effect on the instruction. You can query the count status of the Q parameter. If the current counter value equals "0", the signal state of parameter Q is set to "1". In all other cases, the signal state of the Q parameter is "0". The value of the LOAD parameter is set to the value of the CV parameter when the signal state of the PV parameter changes to "1". As long as the signal state of the LOAD parameter is "1", the signal state of the CD parameter has no effect on the instruction. **STL**: LOAD load input is the top of stack value. CD count down input value is loaded in the second stack level.

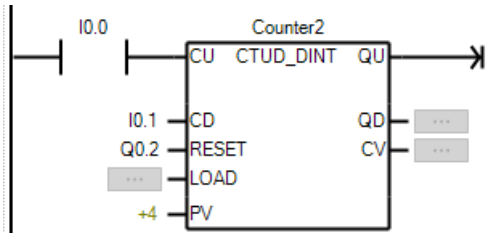
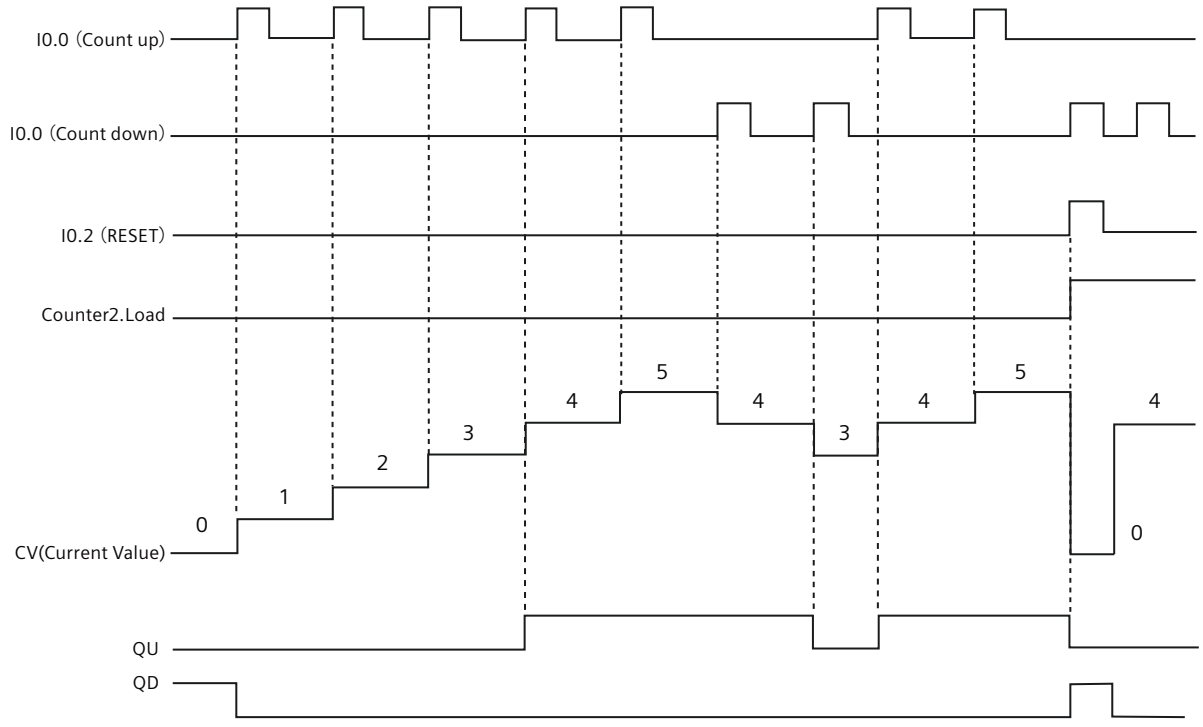
LAD/FBD	STL	Description
	**CTUDDINT ctud1,** **ctud1.CD,** **ctud1.RESET,** **ctud1.LOAD,** **ctud1.PV,** **ctud1.QD,** **ctud1.CV**	**LAD/FBD:** Use the "Count up and down" instruction to increment or decrement the counter value at the CV parameter. When the signal state at the CU parameter changes from "0" to "1" (rising edge), the current counter value is incremented by 1 and stored in the CV parameter. When the signal state of the CD parameter changes from "0" to "1" (rising edge), the counter value of the CV parameter is decremented by 1. If there is a rising edge at the CU and CD inputs in a program cycle, the current counter value of the CV parameter remains unchanged. After the counter value reaches the upper limit of the data type specified by parameter CV, it stops incrementing. When the limit is reached, the counter value is no longer incremented on a rising edge. Once the counter reaches 0, the value will no longer decrease. When the signal state in the LOAD parameter changes to "1", the counter value of the CV parameter is set to the value of the PV parameter. As long as the signal state of the LOAD parameter is "1", the signal state of the CU and CD parameter has no effect on the instruction. The counter value is set to 0 when the signal state of the RESET parameter changes to "1". As long as the RESET parameter has signal state "1", a change in the signal state of the CU, CD and LOAD parameters has no effect on the "Count up and down" instruction. The status of the up-counter can be queried in the QU parameter. If the current counter value is greater than or equal to the value of parameter PV, the signal state of parameter QU is set to "1". In all other cases, the signal state of the QU parameter is "0". The status of the down counter can be queried in the QD parameter. If the current counter value is less than or equal to 0, the signal state of parameter QD is set to "1". In all other cases, the signal state of the QD parameter is "0". **STL**: 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	Operands
CU	BOOL	Power flow
CD	BOOL	I, Q, V, M, T, C, SM, S, L, global variables, FB member variables
RESET	BOOL	I, Q, V, M, T, C, SM, S, L, global variables, FB member variables
LOAD	BOOL	I, Q, V, M, T, C, SM, S, L, global variables, FB member variables
PV	DINT	I, Q, V, M, T, C, SM, AC, L, *AC, constant, global variable, FB member variable
Q	BOOL	Power flow
QU	BOOL	Power flow
QD	BOOL	I, Q, V, M, T, C, SM, S, L, global variables, FB member variables
CV	DINT	I, Q, V, M, T, C, SM, AC, L, *AC, constants, global variables, FB member variables

Example CTD count down

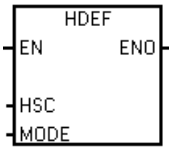
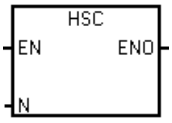
<table>
<tr><th>LAD</th><th></th><th>STL</th></tr>
<tr><td> 
</td><td>The current value CV counts from 3 to 0.
When I0.1 is off, a rising edge at I0.0 decreases the current value CV
I0.1 On loads countdown preset value 3</td><td>Network 1
LD I0.0
CTUDINT Counter1, I0.1, +3,
Counter1.CV
= Q0.0</td></tr>
<tr><td colspan="3">Timing diagrams
 
</td></tr>
</table>

Example CTUD count up/down

<table>
<tr><th>LAD</th><th></th><th>STL</th></tr>
<tr><td> 
</td><td>I0.0 counts up
I0.1 counts down
I0.2 resets current value to 0</td><td>Network 1
LD I0.0
CTUDDINT Counter2, I0.1,
Q0.2, Counter2.LOAD, +4,
Counter2.QD, Counter2.CV</td></tr>
<tr><td colspan="3">Timing diagrams
 
</td></tr>
</table>

9.6.3 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 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 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 346) 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 346)" 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 348)
- High-speed counter summary (Page 346)
- Example initialization sequences for high-speed counters (Page 359)
- Noise reduction for high-speed inputs (Page 348)

9.6.4 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）.

9.6.5 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. 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.

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.

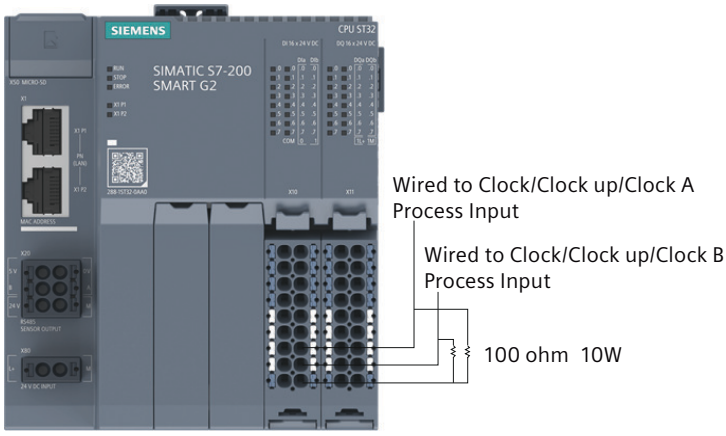


Figure 9-1 Pull-down resistor wiring for open-collector HSC input drivers

9.6.6 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 346)" 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).

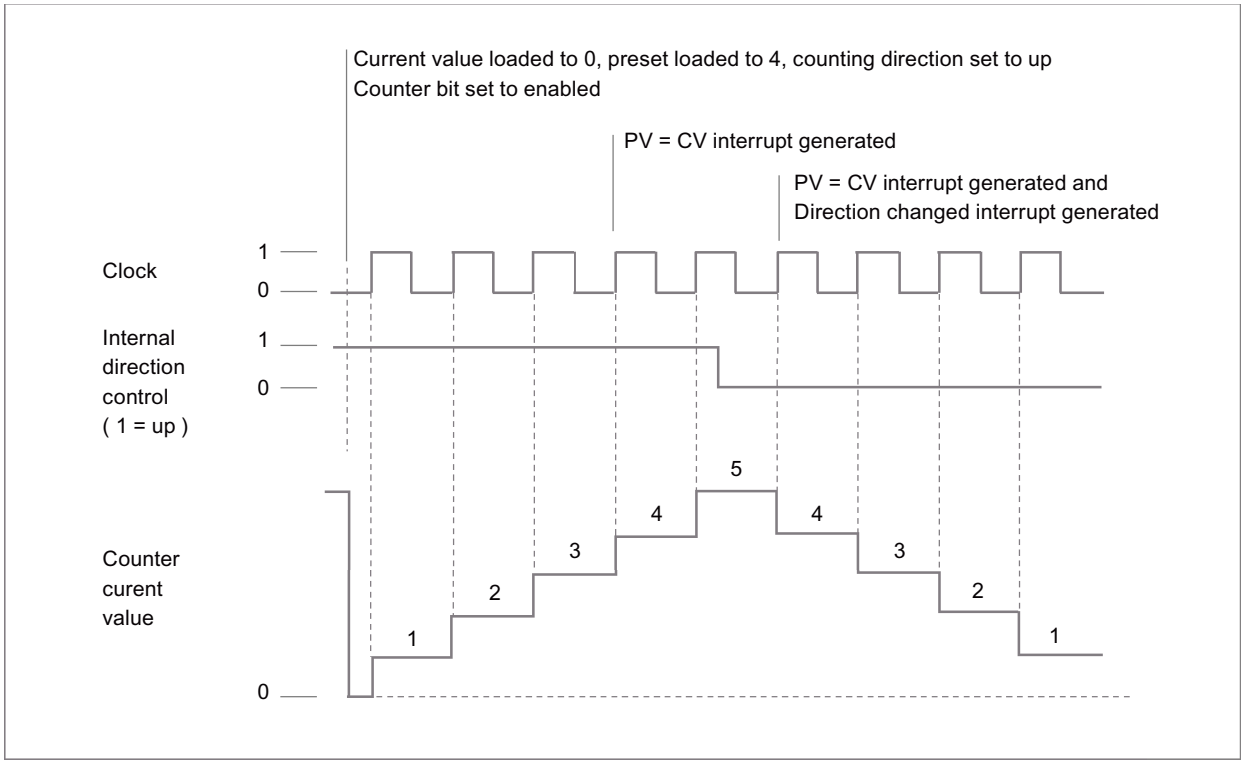
HDEF instruction sets the counting mode

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.

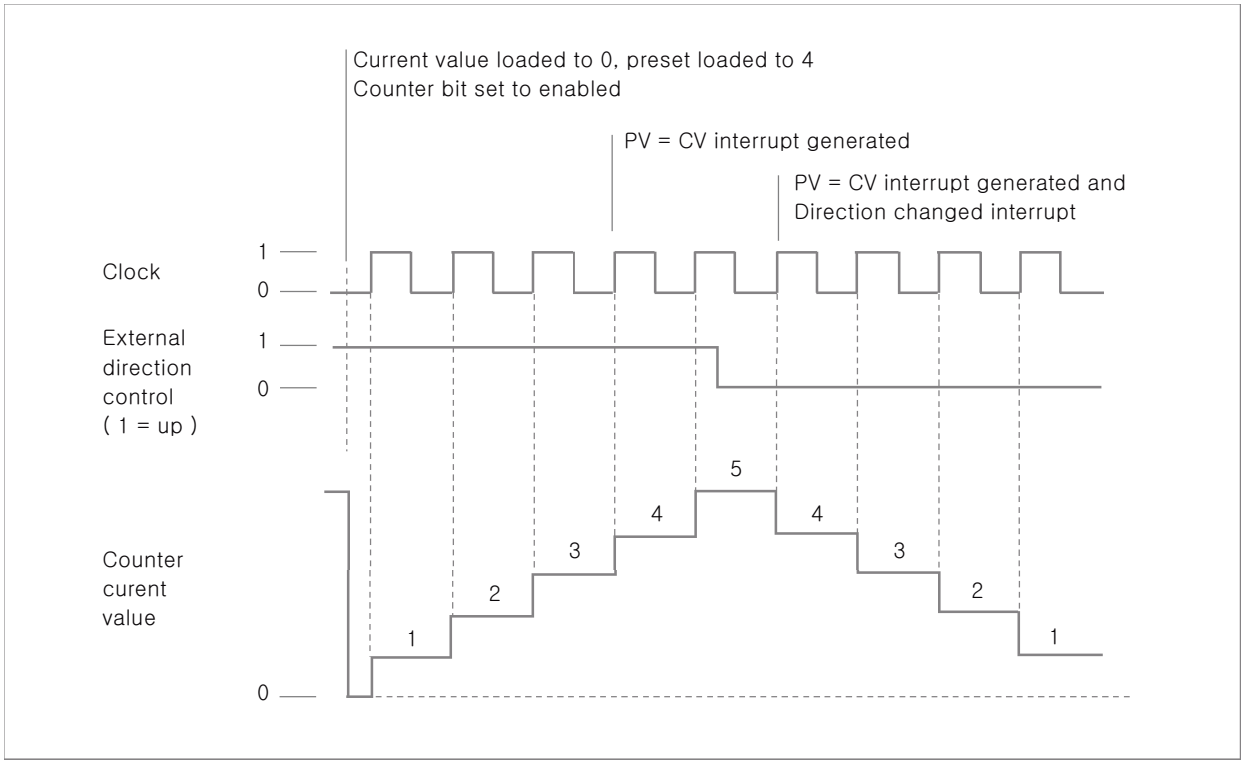
Mode	Description	Input assignment		
0	Single-phase counter with internal direction control	Clock		
1		Clock		Reset
3	Single-phase counter with external direction control	Clock	Direction	
4		Clock	Direction	Reset
6	Two-phase counter with 2 clock inputs	Clock Up	Clock Down	
7		Clock Up	Clock Down	Reset
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

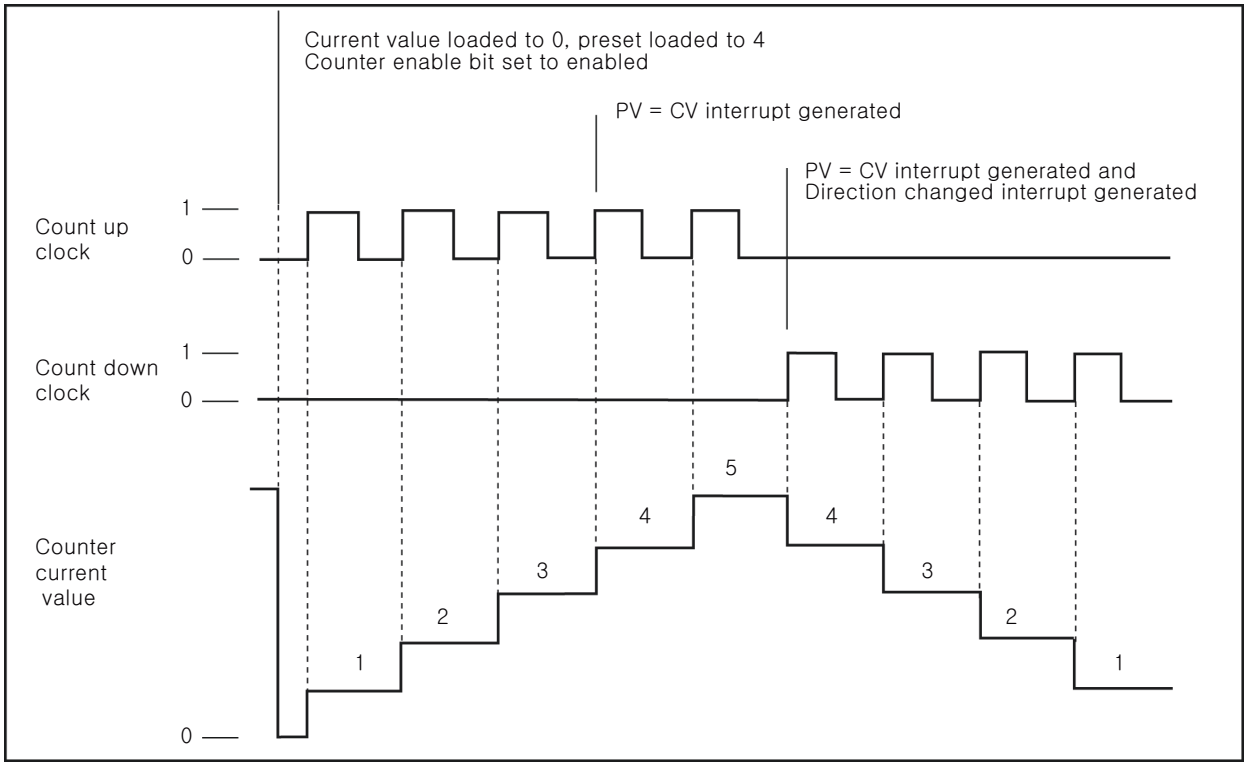


HSC modes 3 and 4

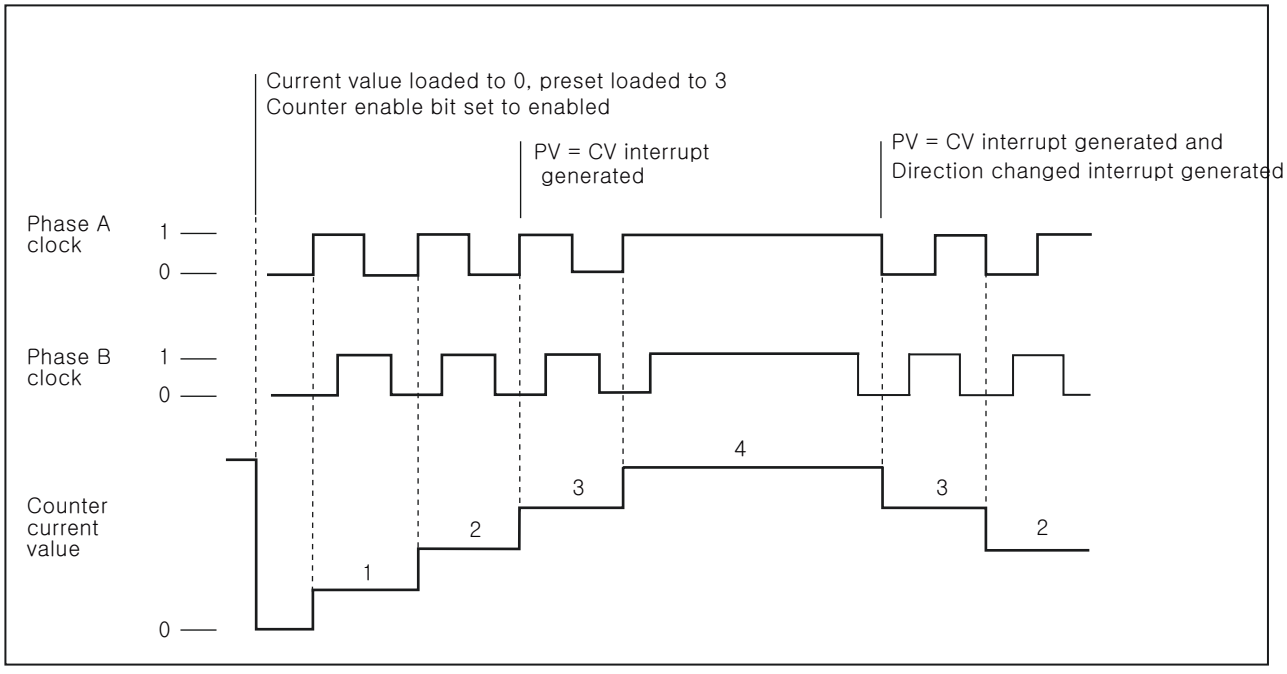


HSC modes 6 and 7

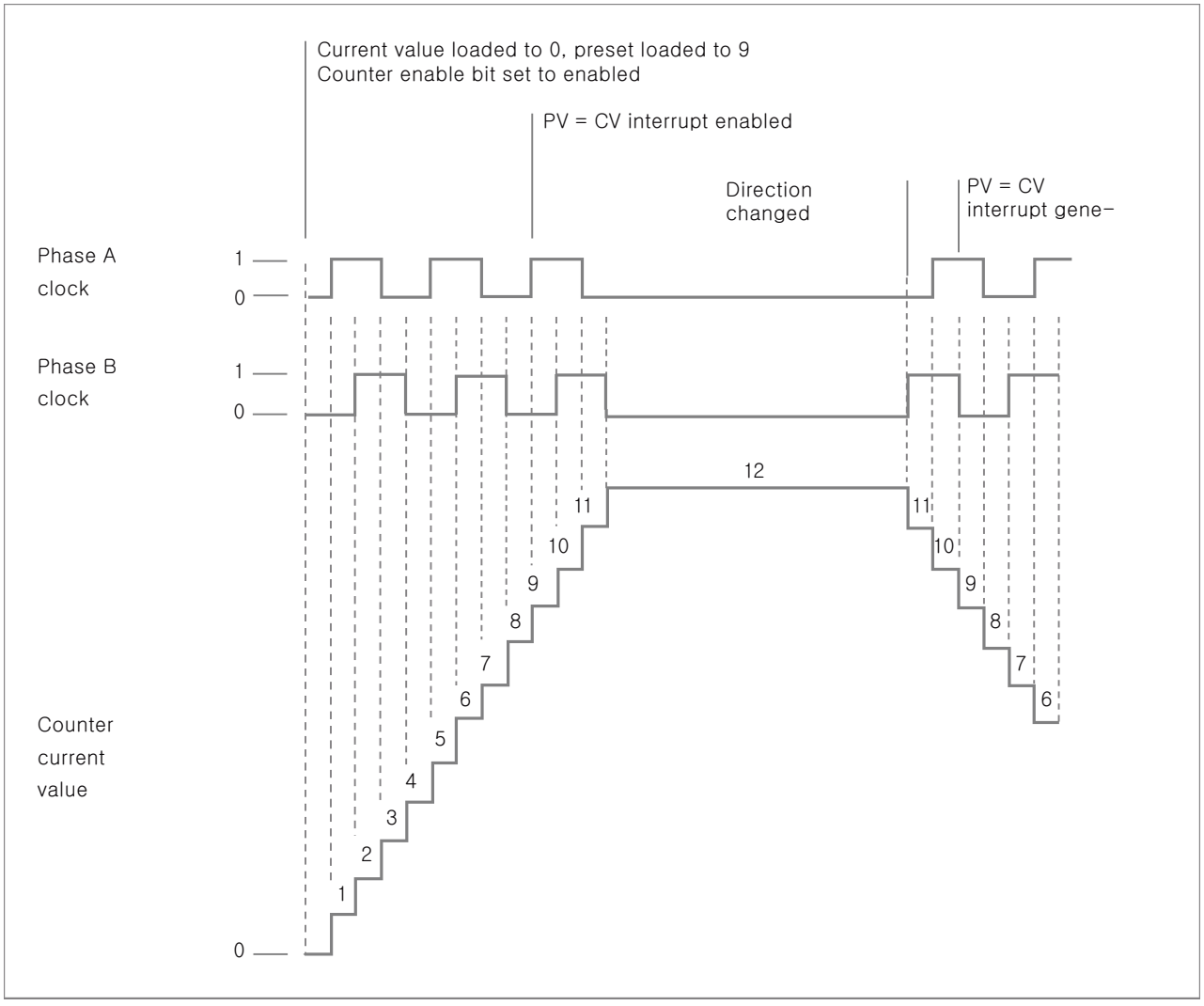
When you use counting modes 6 or 7, and rising edges on both the up clock and down clock inputs occur within 0.3 microseconds of each other, the high-speed counter could see these events as happening simultaneously. If this happens, the current value is unchanged and no change in counting direction is 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)



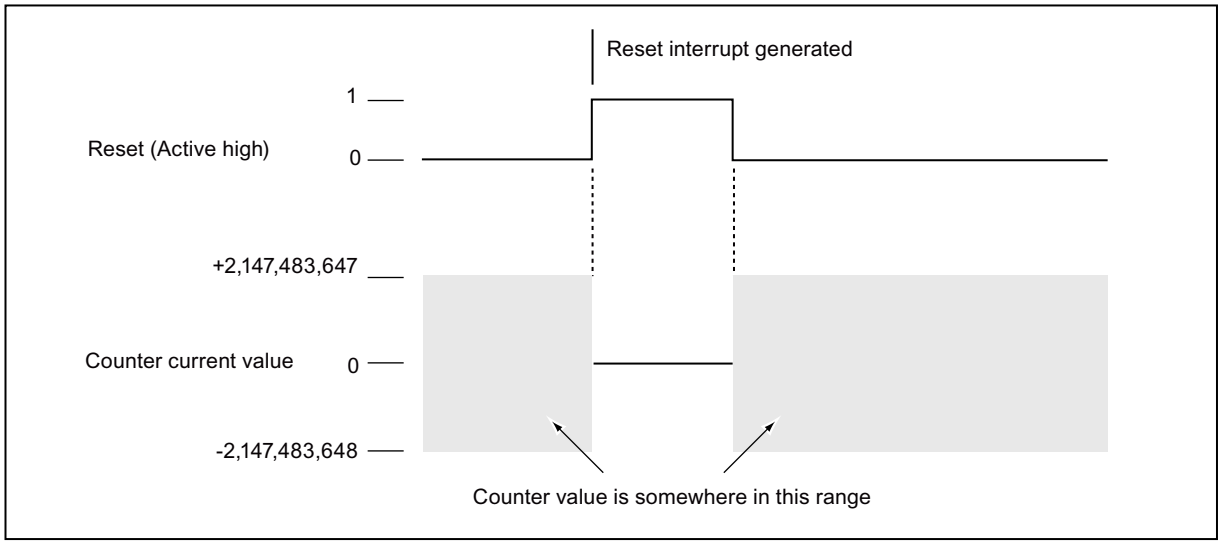
HSC modes 9 and 10 (AB quadrature phase 4x)



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.

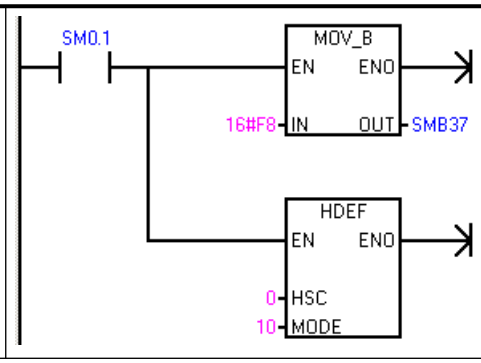
Once the HDEF instruction has been executed, you cannot change the counter setup unless you first place the CPU in STOP mode.

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

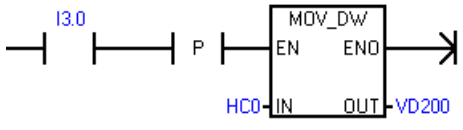
Read the HSC current value with your program

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):

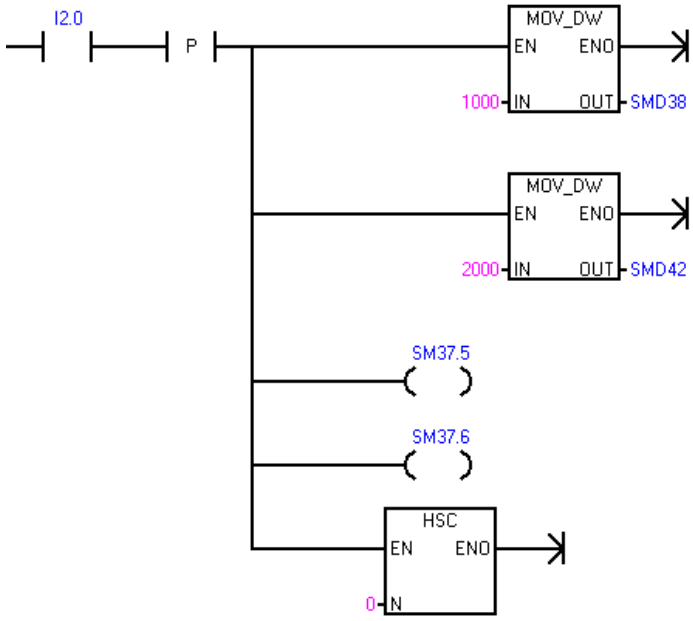
1. Load the value to be written into the appropriate SM new current value and/or new preset value (see the table below). Loading these new values does not affect the high-speed counter yet.
2. Set or clear the appropriate bits in the appropriate control byte to indicate whether to update the current 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 398).

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 9-14 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 344)
- High-speed counter summary (Page 346)
- Example initialization sequences for high-speed counters (Page 359)

9.6.7 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):

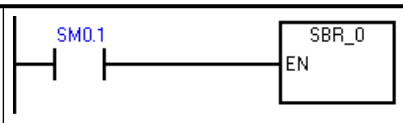
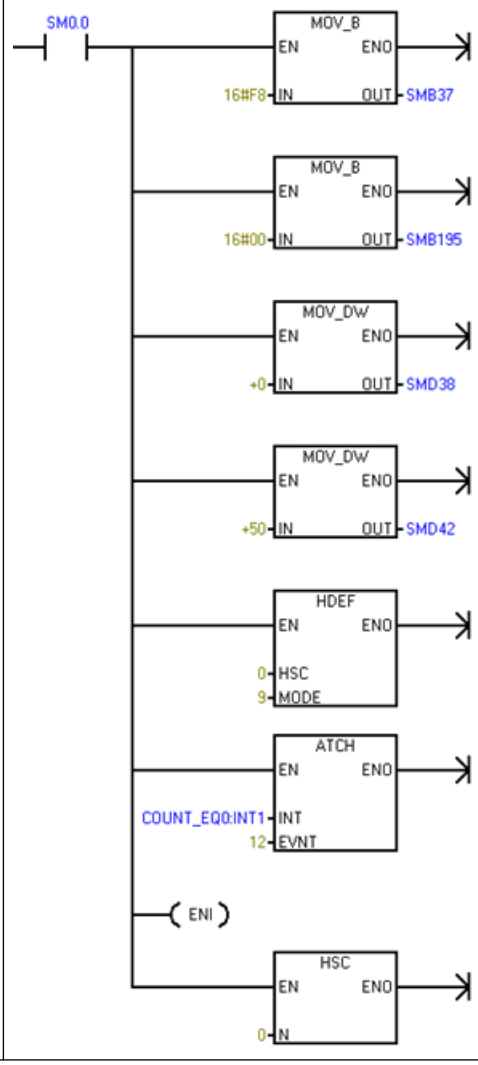
1. Load SMB37 to write the desired preset value:
   SMB37 = 16#A0
   - Enables the counter
   - Writes the new preset value
2. Load SMD42 (double-word-sized value) with the desired preset value.
3. Execute the HSC instruction to cause the CPU to program HSC0.

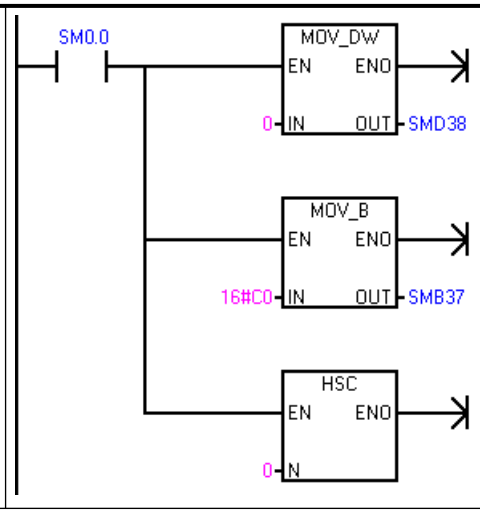
Disabling a high-speed counter (any mode)

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 344)
- High-speed counter summary (Page 346)
- High-speed counter programming (Page 348)
- Interrupt instructions (Page 398)

9.6.8 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

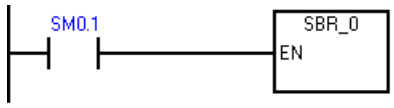
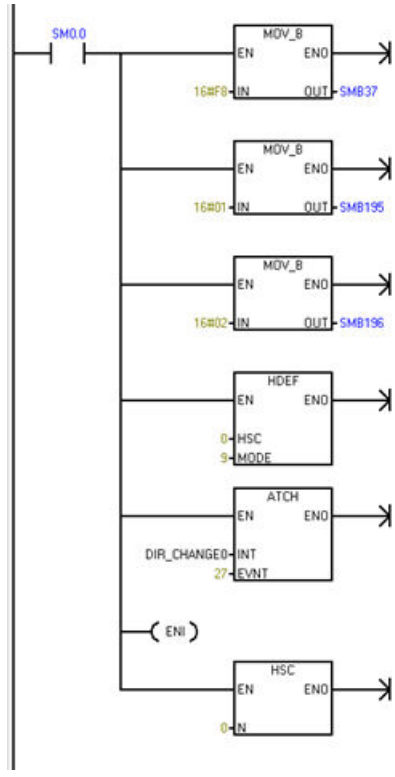
   SMB37 = 16#98
   - Enables the counter
   - Sets the direction of the HSC to count up
2. Execute the HSC instruction to cause the CPU to program HSC0.

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:

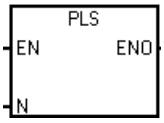
- High-speed counter instructions (Page 344)
- High-speed counter summary (Page 346)
- High-speed counter programming (Page 348)
- Interrupt instructions (Page 398)

9.7 Pulse output

9.7.1 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 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.

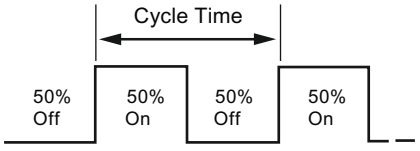
Before enabling PTO/PWM operation, set the value of the process image register for Q0.0, Q0.1, Q0.3, Q0.2 and Q0.4 to 0.

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.

9.7.2 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. See the following figure. 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 9-15 Pulse count and frequency in the PTO function

Pulse count / frequency	Response
Frequency < 1 Hz	Frequency defaults to 1 Hz
Frequency > 200,000 Hz	Frequency defaults to 200,000 Hz

Pulse count / frequency	Response
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 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 9-16 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.

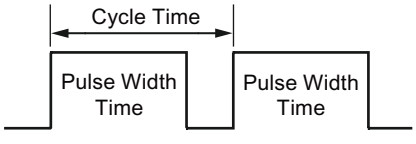
[2] Entering a value of 0 for the number of segments generates a non-fatal error. No PTO output is generated.

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.

9.7.3 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.

9.7.4 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 398).) 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 9-17 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 9-18 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				

9.7.5 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.

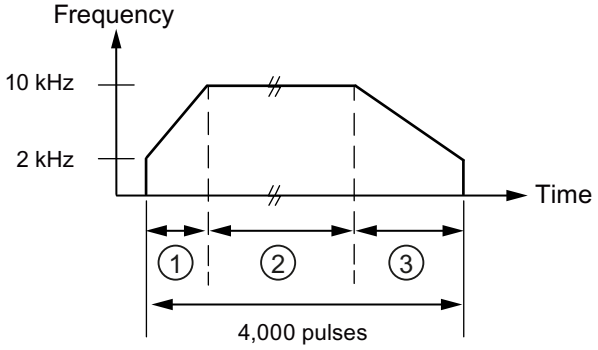
The figure below illustrates sample profile table values required to generate an output waveform:

- 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 9-19 Profile table values

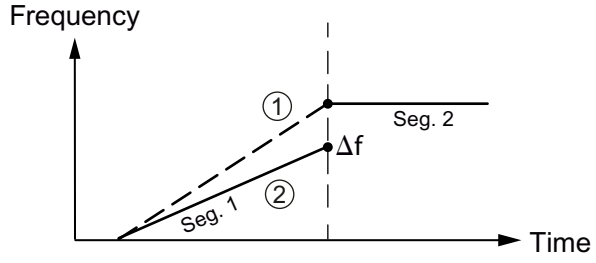
Address	Value	Description	
VB500	3	Total number of segments	
VD501	2,000	Starting frequency (Hz)	Segment 1
VD505	10,000	Ending frequency (Hz)	
VD509	200	Number of pulses	

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:

$\mathbf{CT_{Final} = CT_{Initial} + (\Delta CT * PC)}$

$\mathbf{F_{Initial} = 1 / CT_{Initial}}$

$\mathbf{F_{Final} = 1 / CT_{Final}}$

where:

$\mathbf{CT_{Initial}}$	Starting cycle time (s) for this segment
ΔCT	Delta cycle time (s) of this segment
PC	Quantity of pulses in this segment
$\mathbf{CT_{Final}}$	Ending cycle time (s) for this segment
$\mathbf{F_{Initial}}$	Starting frequency (Hz) for this segment
$\mathbf{F_{Final}}$	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:

$\mathbf{\Delta F = F_{Final} - F_{Initial}}$

$\mathbf{T_s = PC / (F_{min} + (| \Delta F | / 2))}$

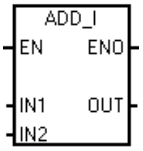
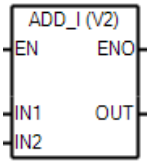
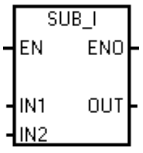
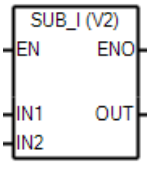
$\mathbf{A_s = \Delta F / T_s}$

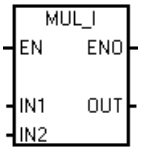
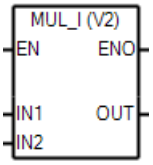
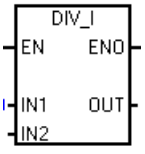
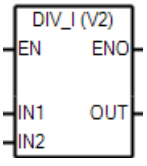
where:

$\mathbf{T_s}$	Time duration (s) of this segment
$\mathbf{A_s}$	Frequency acceleration (Hz/s) of this segment
PC	Quantity of pulses in this segment
$\mathbf{F_{min}}$	Minimum frequency (Hz) for this segment
ΔF	Delta (total change in) frequency (Hz) for this segment

9.8 Math

9.8.1 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

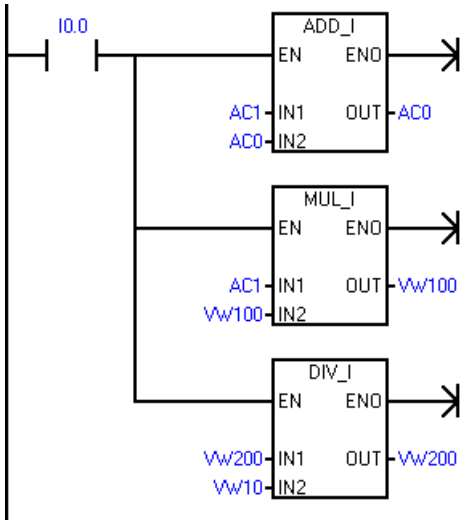
contain valid status. If SM1.3 is set during a divide operation, then the other math status bits are left unchanged.

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



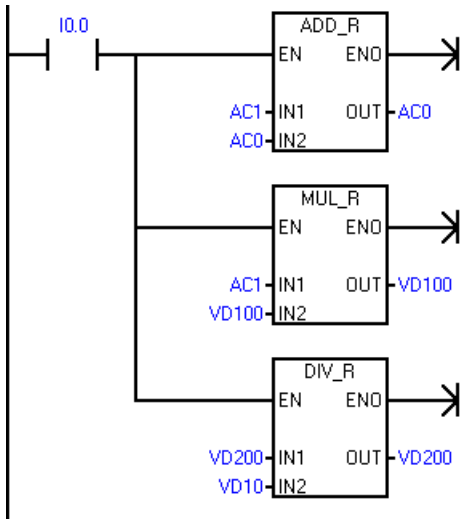
STL

```
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

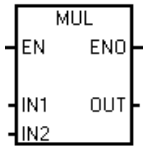
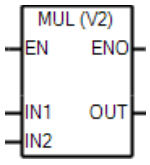
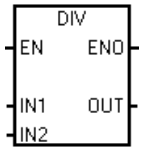
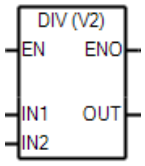
Example: Real math instructions

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

9.8.2 Multiply integers to double integer and divide integer with remainder

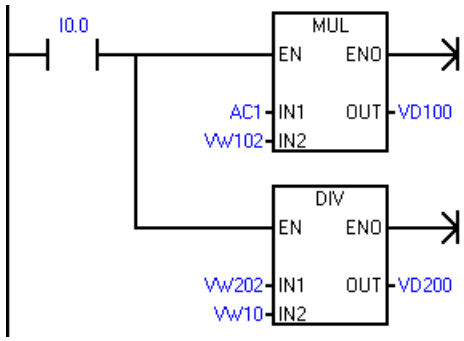
LAD / FBD	STL	Description
	**MUL IN1, IN2, OUT**	The multiply integer to double Integer instruction multiplies two 16-bit integers and produces a 32-bit product. In STL, the least-significant word (16 bits) of the 32-bit OUT is used as one of the factors. • LAD, FBD and STL: IN1 * IN2 = OUT • STL legacy: 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**

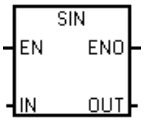
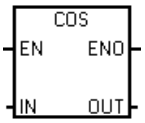
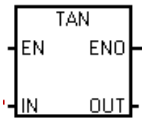
[1] VD100 contains: VW100 and VW102, and VD200 contains: VW200 and VW202.

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

9.8.3 Trigonometry, natural logarithm/exponential, and square root

Trigonometric and inverse trigonometric instructions

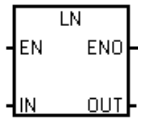
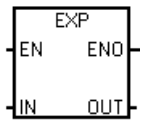
LAD / FBD	STL	Description
	**SIN IN, OUT**	The sine (SIN), cosine (COS), and tangent (TAN) instructions evaluate the trigonometric function of the angle value IN and place the result in OUT. The arcsine (ARC_SIN), arccosine (ARC_COS), and arctangent (ARC_TAN) instructions evaluate the inverse trigonometric 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 IN, OUT**	
	**TAN IN, OUT**	
	**ARCSIN IN, OUT**	
	**ARCCOS 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

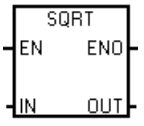
LAD / FBD	STL	Description
	**LN IN, OUT**	The Natural Logarithm instruction (LN) performs the natural logarithm of the value in IN and places the result in OUT. The Natural Exponential instruction (EXP) performs the exponential operation of e raised to the power of the value in IN and places the result 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 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

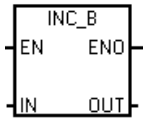
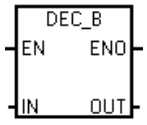
LAD / FBD	STL	Description
	**SQRT IN, OUT**	The Square Root instruction (SQRT) takes the square root of a real number (IN) and produces a real number result OUT. • SQRT (IN)= OUT To obtain other roots: • 5 cubed = 5^3 = EXP(3*LN(5)) = 125 • The cube root of 125 = 125^(1/3) = EXP((1/3)*LN(125))= 5 • The 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.

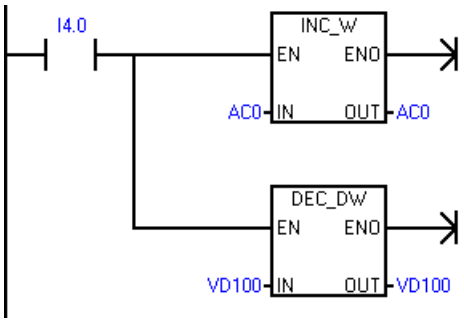
9.8.4 Increment and decrement

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

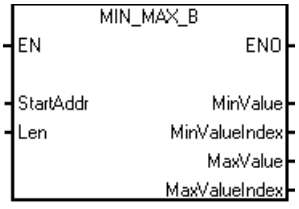
Example: Increment and decrement

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

9.8.5 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.

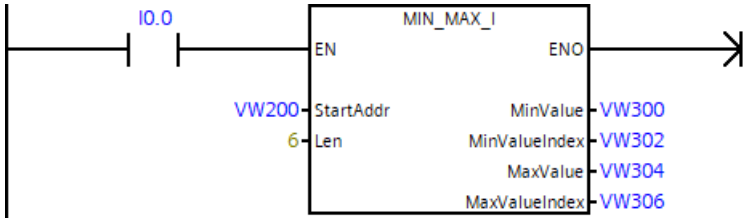
Non-fatal errors with ENO = 0

- 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

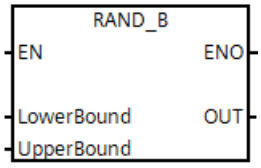
Get MinValue, MinValueIndex, MaxValue and MaxValueIndex in source 6 word address sequence (VW200 to VW210) to destination word address sequence (VW300 to VW306).

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

9.8.6 Get a random value

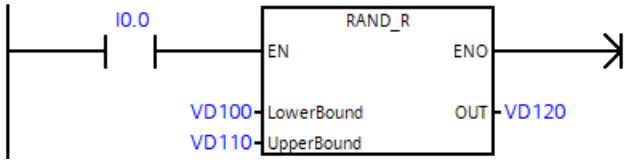
LAD/FBD	STL	Description
 RAND_I RAND_DI RAND_R	**RANDB** **LowerBound** **UpperBound OUT** **RANDI** **LowerBound** **UpperBound OUT** **RANDDI** **LowerBound** **UpperBound OUT** **RANDR** **LowerBound** **UpperBound OUT**	The Random instructions generate a random value in the set range to the OUT output.

Non-fatal errors with ENO = 0

- 0X0090: invalid parameter, when LowerBound > UpperBound.

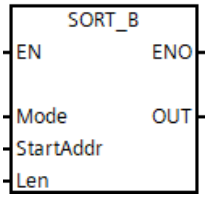
Input/output	Data type	Operand
LowerBound	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD, Constant
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
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	

9.8.7 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.

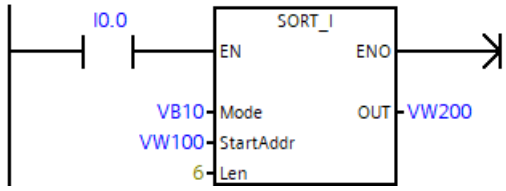
Non-fatal errors with ENO = 0

- 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

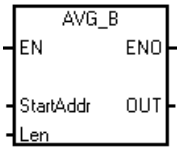
[1] 0: Ascending order; 1: Descending order

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

9.8.8 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.

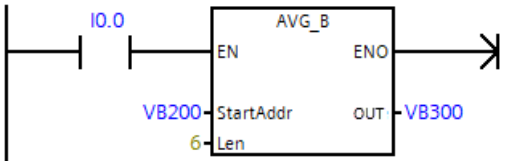
Non-fatal errors with ENO = 0

- 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

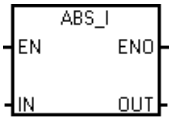
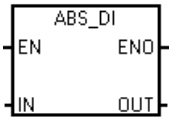
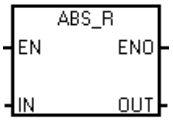
Input / output	Data type	Operand
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *AC, *LD
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

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					

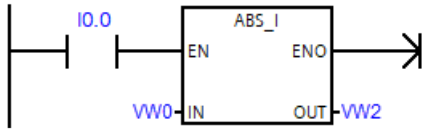
9.8.9 Get absolute values

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.

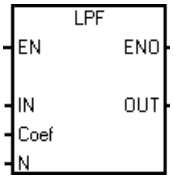
Non-fatal errors with ENO = 0	SM bits affected
• 0X0090: invalid parameter	• SM1.0 Result of operation = zero

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	

9.8.10 Low-pass filter

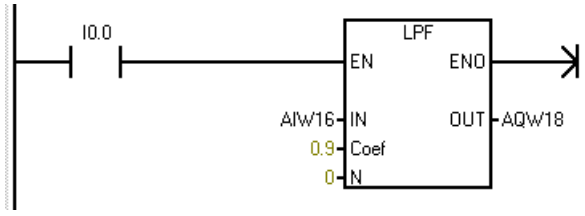
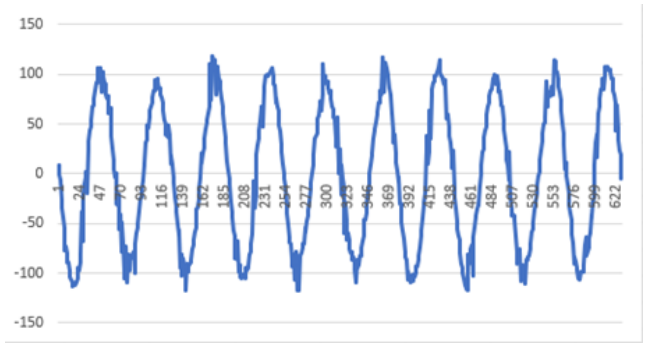
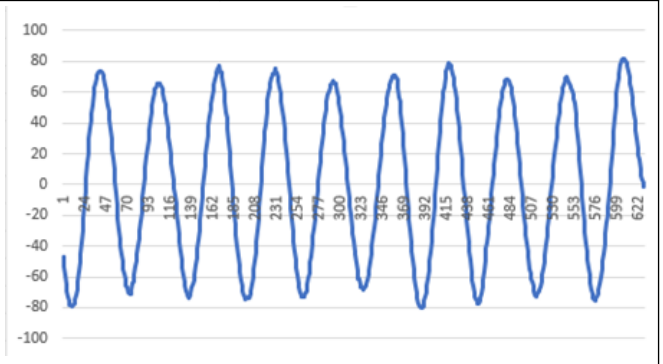
LAD / FBD	STL	Description
	**LPF** **IN,Coef,N,OUT**	You can use the "Low-pass filter" instruction to attenuate high frequency components, such as noise in a signal. The result of the instruction is calculated using the following equation: y(n) = Coef*y(n-1) + （1- Coef）* x(n); y(n) is the current filter 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

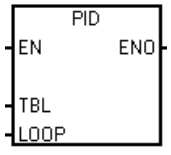
- 0X0090: invalid parameter, when the value of N and Coef is out of range.

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

9.8.11 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 1217).

Auto-Tune capability has been incorporated into the PID instruction. Refer to "PID loops and tuning" for a detailed description of auto-tuning. The "PID Tune control panel" 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

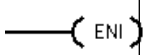
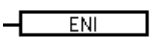
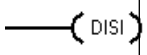
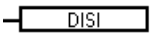
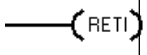
The setpoint of the low range and the setpoint of the high range should correspond to the process variable low range and high range.

9.9 Interrupt

9.9.1 Interrupt instructions

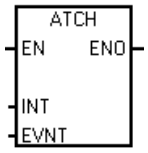
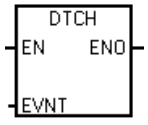
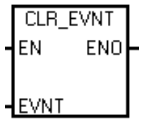
When you make the transition to RUN mode, interrupts are initially disabled. In RUN mode, you can enable interrupt processing by executing the ENI (enable Interrupt) instruction. Executing the DISI (disable interrupt) instruction inhibits the processing of interrupts; however, active interrupt events will continue to be queued.

ENI, DISI, and CRETI

LAD	FBD	STL	Description
		**ENI**	The enable interrupt instruction globally enables processing of all attached interrupt events.
		**DISI**	The disable interrupt instruction globally disables processing of all interrupt events.
		**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 INT, EVNT**	The attach interrupt instruction associates an interrupt event EVNT with an interrupt routine number INT and enables the interrupt event.
	**DTCH EVNT**	The detach interrupt instruction disassociates an interrupt event EVNT from all interrupt routines and disables the interrupt event.
	**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

9.9.2 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 CPU ST32
0	I0.0 Rising edge	Y
1	I0.0 Falling edge	Y
2	I0.1 Rising edge	Y
3	I0.1 Falling edge	Y

Event	Description	CPU SR40/ST40 CPU SR60/ST60 CPU ST32
4	I0.2 Rising edge	Y
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

Event	Description	CPU SR40/ST40 CPU SR60/ST60 CPU ST32
49	I72.0 Rising edge (signal board slot 1, input 0)	Y
50	I72.0 Falling edge (signal board slot 1, input 0)	Y
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

9.9.3 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.

9.9.4 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.

9.9.5 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

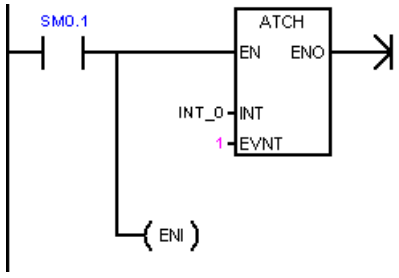
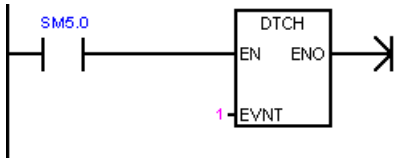
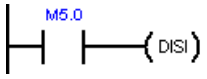
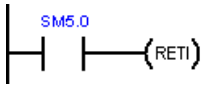
The following table shows all interrupt events, with their priority and assigned event number.

Queue	Queue depth for all S7-200 SMART CPU models
Communications queue	4
I/O interrupt queue	16
Timed interrupt queue	8

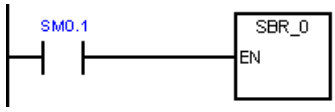
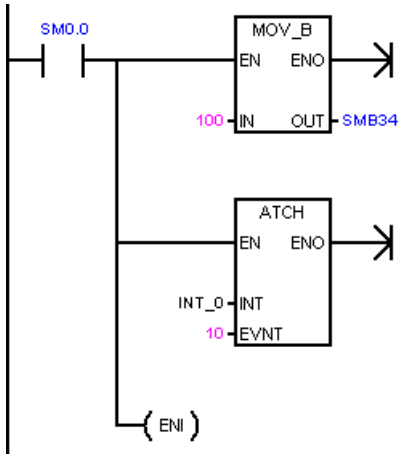
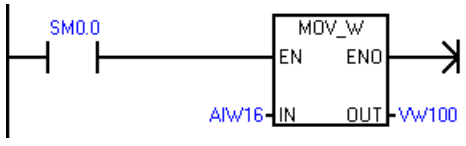
Interrupt queue overflow bits

Description (0 = No Overflow, 1 = Overflow)	SM Bits
Communications queue	SM4.0
I/O interrupt queue	SM4.1
Timed interrupt queue	SM4.2

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

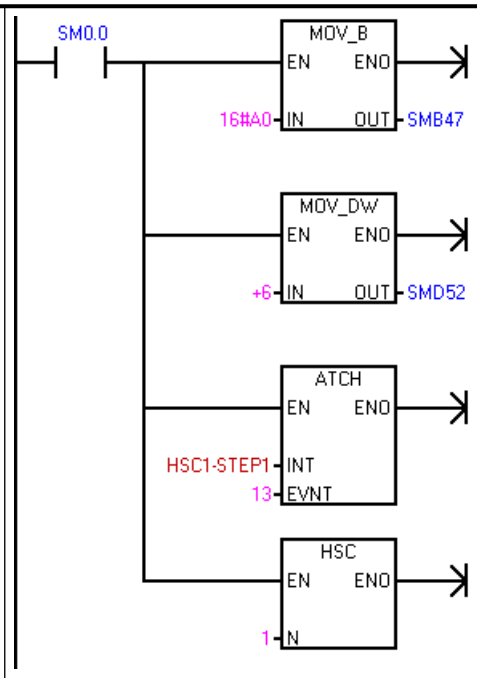
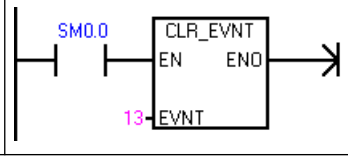
LAD			STL
SBR 1 Network 1		HSC instruction wizard: Set control bits, write preset. PV = 6 Attach interrupt HSC1_STEP1: CV = PV for HC1 Configure HSC 1.	**Network 1** **LD SM0.0** **MOVB 16#A0, SMB47** **MOVD +6, SMD52** **ATCH HSC1_STEP1, 13**
SBR 1 Network 2		Clear unwanted interrupts caused by machine vibration.	**Network 2** **LD SM0.0** **CEVNT 13**

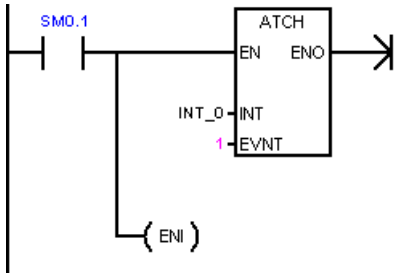
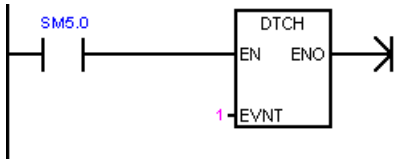
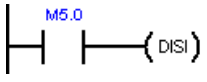
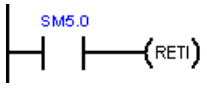
Table 9-20 Priority order for interrupt events

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

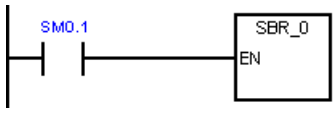
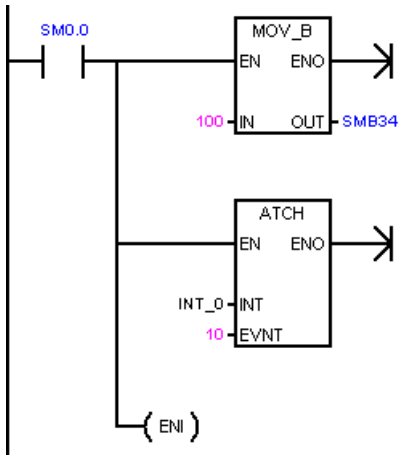
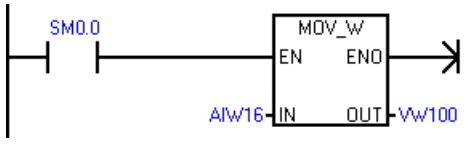
Priority group	Event	Description
Discrete Medium Priority	19	PTO0 Pulse count complete
	20	PTO1 Pulse count complete
	34	PTO2 Pulse count complete
	45	PTO3 Pulse count complete
	67	PTO4 Pulse count complete
	0	I0.0 Rising edge
	2	I0.1 Rising edge
	4	I0.2 Rising edge
	6	I0.3 Rising edge
	1	I0.0 Falling 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

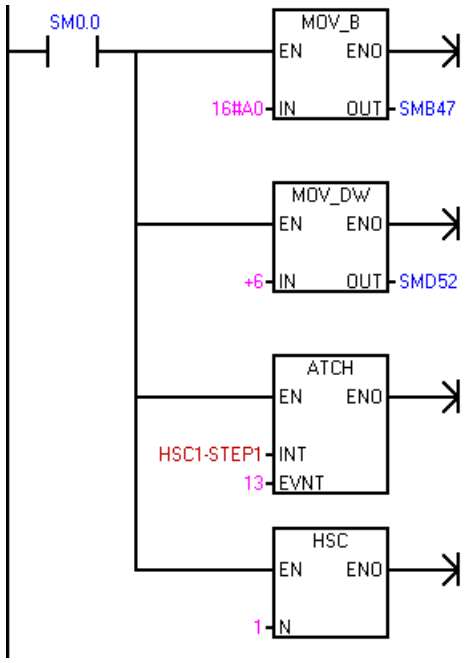
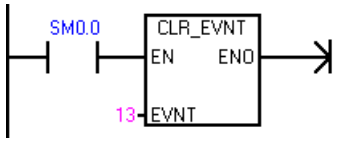
Example 1: Input signal edge detector interrupt

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

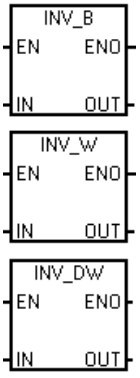
LAD			STL
MAIN Network 1		On the first scan, call subroutine 0.	**Network 1** **LD SM0.1** **CALL SBR_0**
SBR 0 Network 1		Set the interval for the timed interrupt 0 to 100 ms. Attach timed interrupt 0 (Event 10) to INT_0. Global interrupt enable.	**Network 1** **LD SM0.0** **MOVB 100, SMB34** **ATCH 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**

9.10 Logical operations

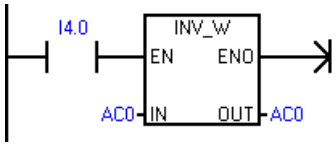
9.10.1 Invert

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

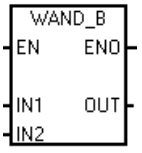
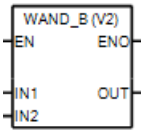
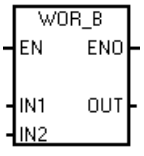
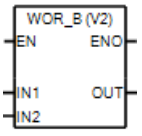
Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address	• SM1.0 Result of operation = zero

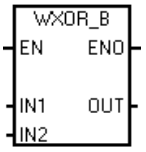
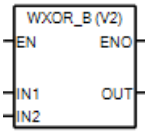
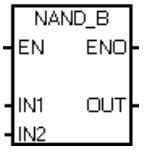
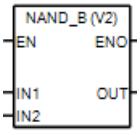
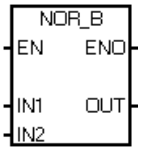
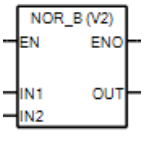
Input / output	Data type	Operand
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC,*VD, *LD, *AC
	WORD	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

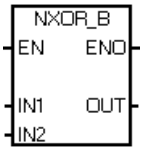
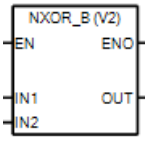
Example: Invert instruction

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**

9.10.2 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**	

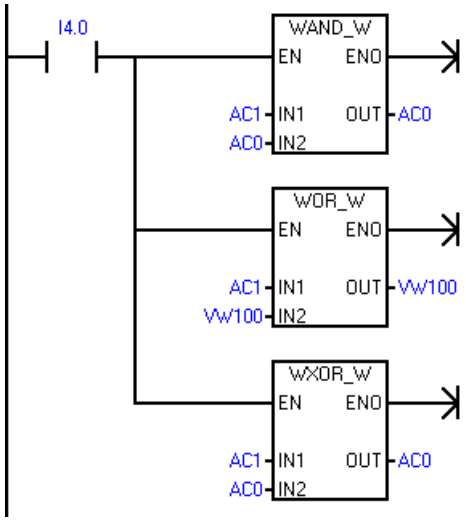
LAD / FBD	STL	Description
 WXOR_W WXOR_DW	**XORB** **IN1,IN2,OUT** **XORW** **IN1,IN2,OUT** **XORD** **IN1,IN2,OUT**	The Exclusive OR Byte, Exclusive OR Word, and Exclusive OR Double Word instructions logically XOR the corresponding bits of two input values IN1 and IN2 and load the result into a memory 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**	

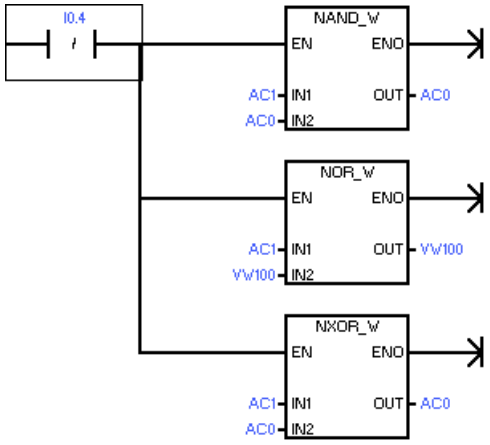
Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address	• SM1.0 Result of operation = zero

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

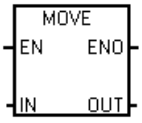
LAD		STL
	AC1 0001 1111 0110 1101 AND AC0 1101 0011 1110 0110 equals AC0 0001 0011 0110 0100 AC1 0001 1111 0110 1101 OR VW100 1101 0011 1010 0000 equals VW100 1101 1111 1110 1101 AC1 0001 1111 0110 1101 XOR AC0 0001 0011 0110 0100 equals AC0 0000 1100 0000 1001	**Network 1** **LD I4.0** **ANDW AC1, AC0, 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**

9.11 Move

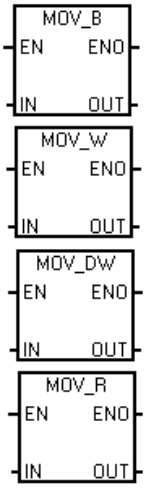
9.11.1 Move

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 errors with ENO = 0	SM bits affected
• 0006H Indirect address	None

Input/output	Data type	Operands
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, Constant
	WORD, INT	IW, QW, VW, MW, SMW, SW, LW, 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, LW
	DWORD, DINT, REAL	ID, QD, VD, MD, SMD, SD, LD
	UDT	
	ARRAY	
	STRING	
	FB Instance	

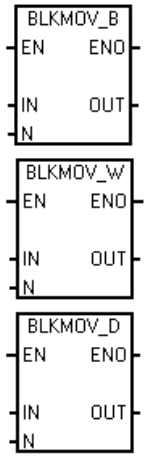
9.11.2 Move byte, word, double word, or real

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.

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

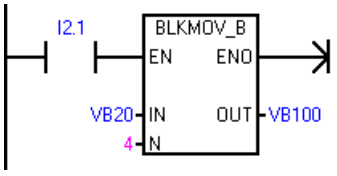
9.11.3 Block move

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

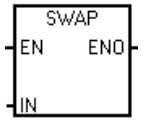
Input / output	Data type	Operand
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC
	WORD, INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AIW, *VD, *LD, *AC
	DWORD, DINT	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC
	WORD, INT	IW, QW, VW, MW, SMW, SW, T, C, LW, 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

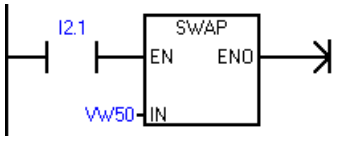
9.11.4 Swap bytes

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

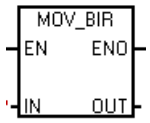
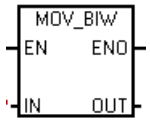
Input / output	Data type	Operand
IN	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC

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

9.11.5 Move byte immediate (read and write)

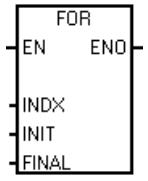
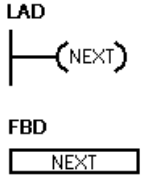
LAD / FBD	STL	Description
	**BIR IN, OUT**	The Move Byte Immediate Read instruction reads the state of physical input IN and writes the result to the memory address OUT, but the process image register is not updated.
	**BIW IN, OUT**	The Move Byte Immediate Write instruction reads the data from the memory 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

9.12 Program control

9.12.1 FOR-NEXT loop

LAD / FBD	STL	Description
	**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.
	**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.

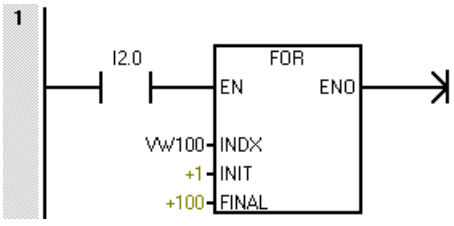
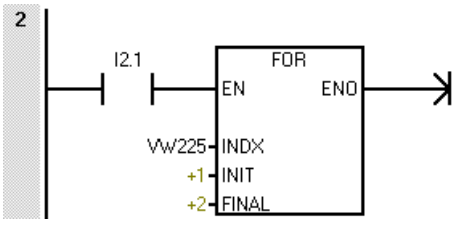
If you enable a FOR-NEXT loop, the execution loop continues until it finishes the iterations, unless you change the FINAL value from within the loop itself. You can change the values while the FOR-NEXT loop is in the looping process. When the loop is enabled again, it copies the INIT value to the INDX value (current loop number).

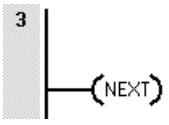
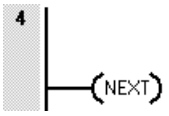
For example, given an INIT value of 1 and a FINAL value of 10, the instructions between the FOR instruction and the NEXT instruction are executed 10 times with the INDX value being incremented: 1, 2, 3, ... 10.

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
	End of inside loop	**Network 3** **NEXT**
	End of outside loop	**Network 4** **NEXT**

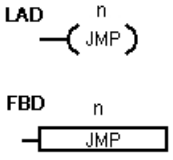
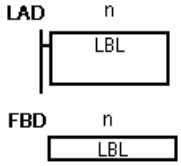
9.12.2 JMP (jump to label)

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

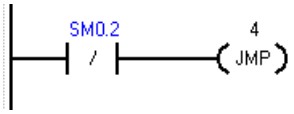
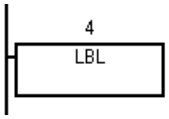
You cannot jump from the main program to a label in either a subroutine, a function block or an interrupt routine. Likewise, you cannot jump from a subroutine or interrupt routine to a label outside that subroutine or interrupt routine.

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
	**JMP N**	The JMP (Jump) instruction performs a branch to the label N within the program.
	**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**

9.12.3 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.

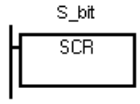
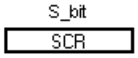
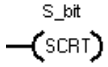
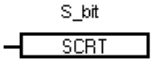
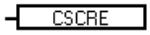
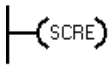
Multiple POUs accessing the same S bit could result in unexpected process operation, possibly resulting in death or severe personal injury.

Check your program to ensure that multiple POUs do not access the same S bit.

Note

SCR programming restrictions

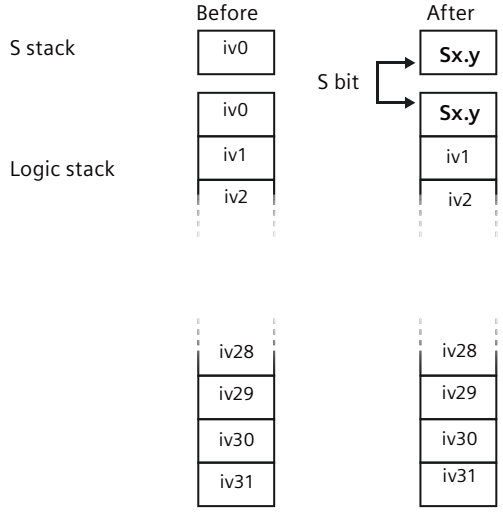
- You cannot jump into or out of an SCR segment; however, you can use Jump and Label instructions to jump around SCR segments or to jump within an SCR segment.
- You cannot use the END instruction in an SCR segment.

LAD	FBD	STL	Description
		**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.
		**SCRT S_bit**	
		**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). 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.
		**SCRE**	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).

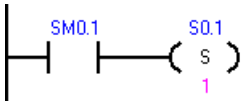
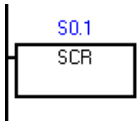
One way to logically divide a program into multiple steps is to use SCR segments. SCR segments can divide your program into a single stream of sequential steps, or into multiple streams that can be active simultaneously. It is possible to have a single stream conditionally diverge into multiple streams, and to have multiple streams conditionally re-converge into a single stream.

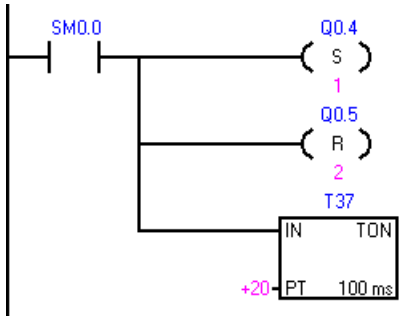
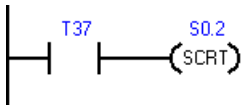
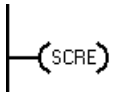
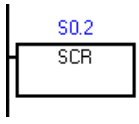
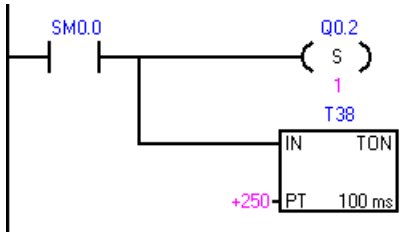
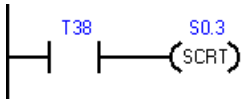
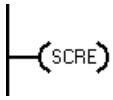
SCR operations

- 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

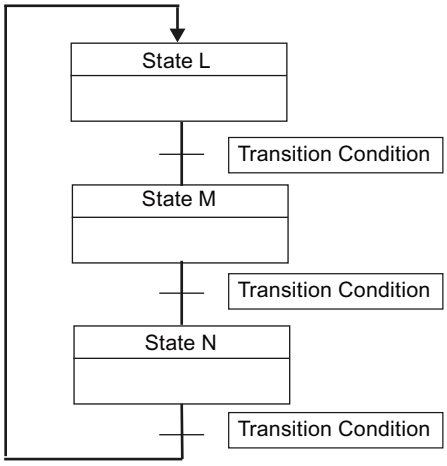
In the following sample program, the first scan bit SM0.1, is used to set S0.1, which will be the active State 1 on the first scan. After a 2-second delay, T37 causes a transition to State 2. This transition deactivates the State 1 SCR (S0.1) segment and activates the State 2 SCR (S0.2) segment.

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**

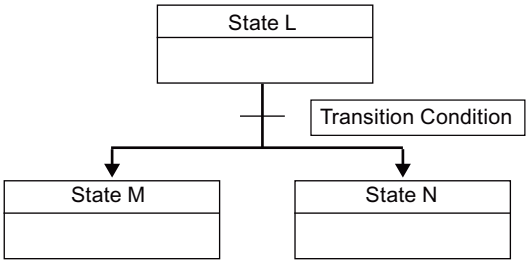
Sequential control flow

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.



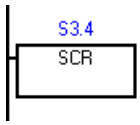
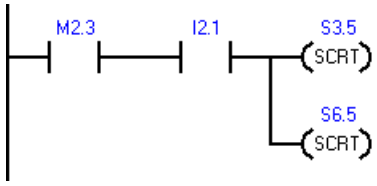
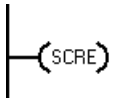
Divergent control flow

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.

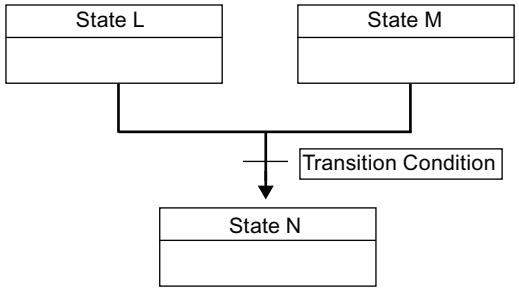
Example: SCR divergent flow control

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**

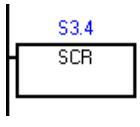
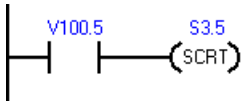
Convergent flow control

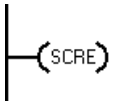
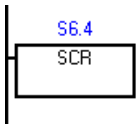
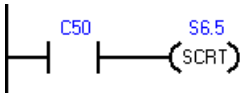
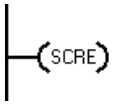
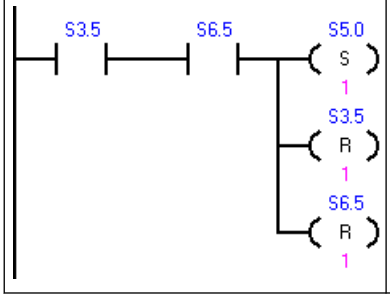
When streams converge, all incoming streams must be complete before the next state is executed.

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.



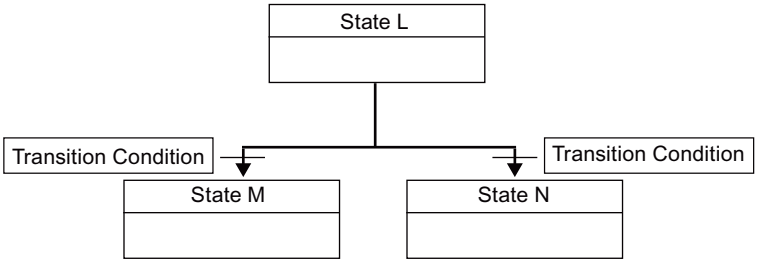
Example: SCR convergent flow control

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**

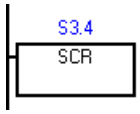
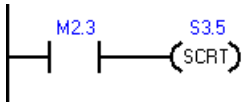
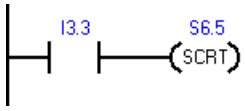
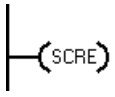
LAD		STL
	End of SCR region for state L	**Network 3** **SCRE**
	Beginning of state M control region	**Network 4** **LSCR S6.4**
	Transition to state M'	**Network 5** **LD C50** **SCRT S6.5**
	End of SCR region for state M	**Network 6** **SCRE**
	When both State L' and State M' are activated: • Enable 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

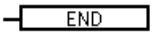
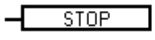
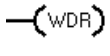
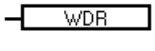
In other situations, a control stream might be directed into one of several possible control streams, depending upon which transition condition becomes true first.



Example: SCR divergent flow control, depending of transition conditions

LAD		STL
	Beginning of state L control region	**Network 1** **LSCR S3.4**
	Transition to state M	**Network 2** **LD M2.3** **SCRT S3.5**
	Transition to state N	**Network 3** **LD I3.3** **SCRT S6.5**
	End of SCR region for state L	**Network 4** **SCRE**

9.12.4 END, STOP, and WDR (watchdog timer reset)

LAD	FBD	STL	Description
		**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**	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**	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.

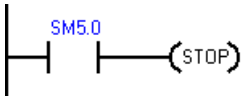
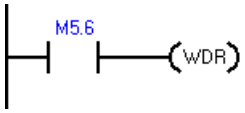
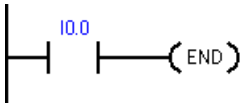
Watchdog timer operation

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**

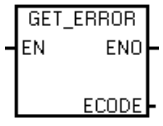
Note

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

9.12.5 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

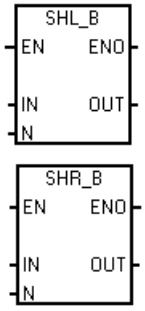
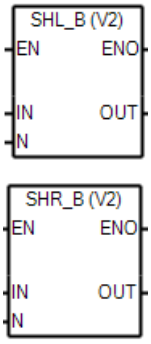
PLC non-fatal error codes (Page 1184)

PLC non-fatal error SM flags (Page 1186)

9.13 Shift and rotate

9.13.1 Shift and rotate

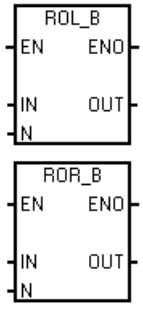
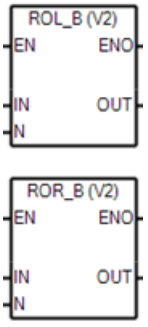
Shift instructions (only the byte size LAD box is illustrated, the others are similar)

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)

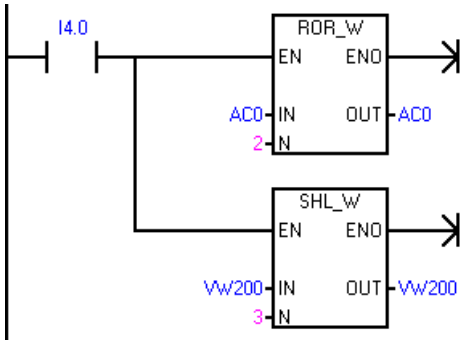
LAD / FBD	STL	Rotate type	Description
	**RLB IN, N, OUT** **RRB IN, N, OUT**	Rotate left byte Rotate right byte	The rotate instructions rotate the bit values of input value IN right or left by the bit position rotate count N and load the result in the memory location assigned to OUT. The rotate operation 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

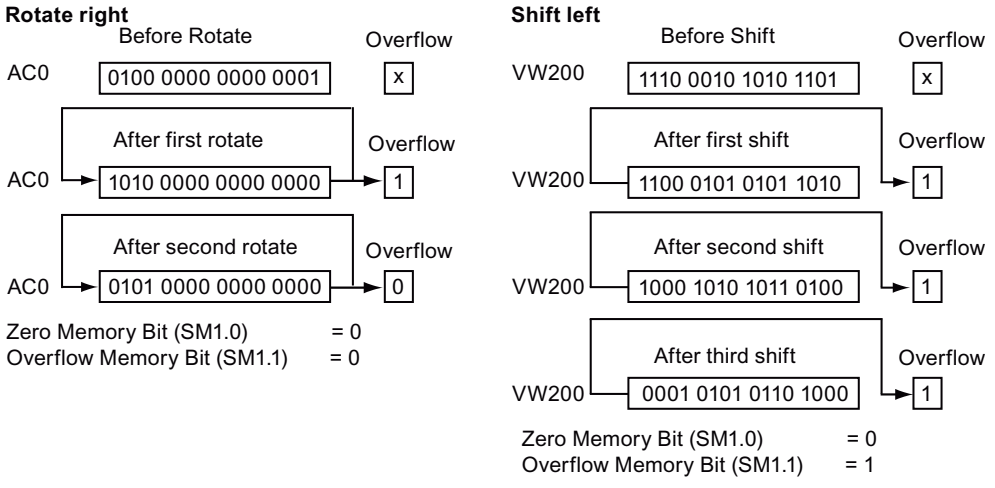
Example: Shift and Rotate instructions

LAD



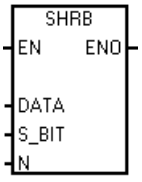
STL

```
Network 1
LD I4.0
RRW AC0, 2, AC0
SLW VW200, 3, VW200
```



9.13.2 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 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

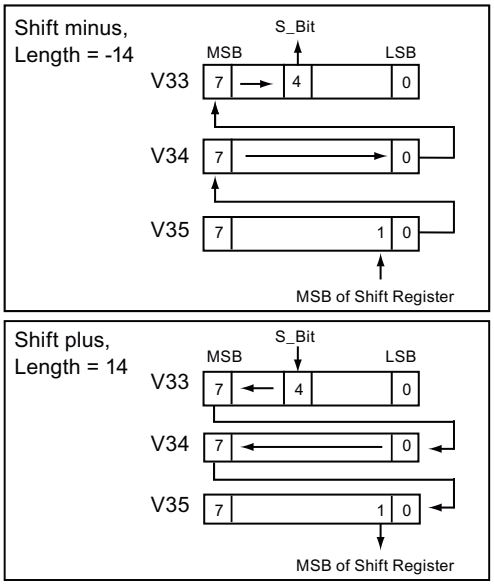
Use the following equation to compute the address of the most significant bit of the Shift Register (MSB.b):

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.



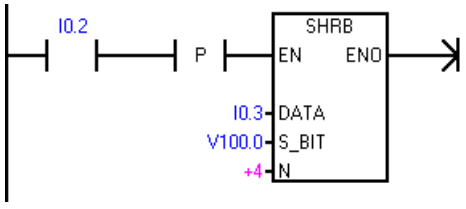
A Shift Minus operation is indicated by a negative value of length N. The input value of DATA shifts into the most significant bit of the shift register, and shifts out of the least significant bit location assigned by S_BIT. The data shifted out is then placed in the overflow memory bit SM1.1.

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

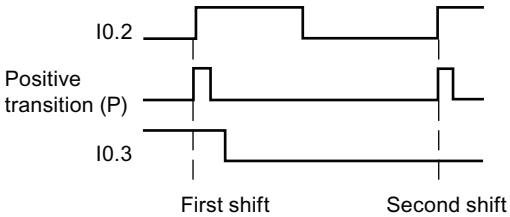
LAD

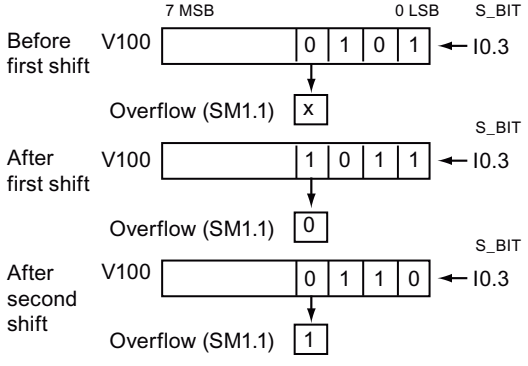


STL

```
Network 1
LD I0.2
EU
SHRB I0.3, V100.0, +4
```

Timing Diagram

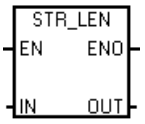




9.14 String

9.14.1 String (Get length, copy, and concatenate)

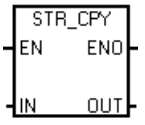
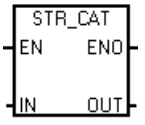
SLEN (String length)

LAD / FBD	STL	Description
	**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
	**SCPY IN, OUT**	The copy string instruction copies the string assigned by IN to the string assigned by 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.

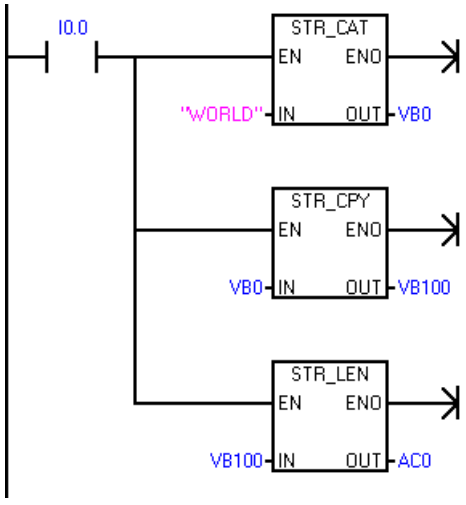
Error conditions with ENO = 0	SM bits affected
• 0006H Indirect address • 0091H Operand out of range	None

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'	' '

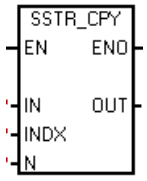
After executing the program

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

9.14.2 Copy substring from string

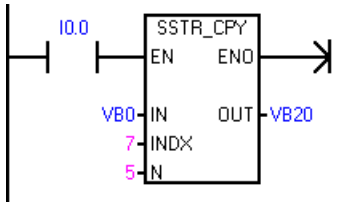
LAD / FBD	STL	Description
	**SSCPY IN, INDX, N, OUT**	The copy substring from string instruction copies the assigned number of characters N from the string specified by IN, starting at the index INDX, to a new string assigned by OUT. Note: The SSTR_CPY instruction operates on bytes and not characters. 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**

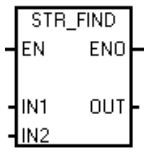
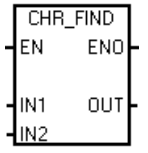
Before executing the program

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'

9.14.3 Find string and first character within string

LAD / FBD	STL	Description
	**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.
	**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.

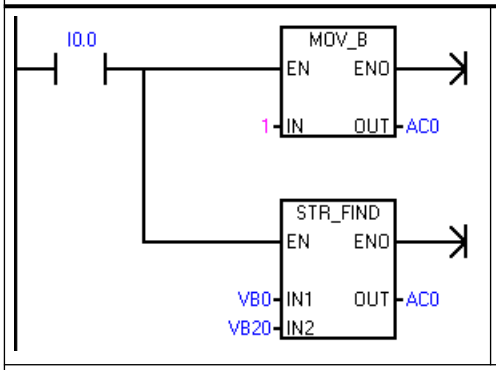
Note: Because Chinese characters are not represented by a single byte, and the string instructions operate on bytes and not characters, unexpected results can occur with the STR_FIND and CHR_FIND instructions with strings containing Chinese characters.

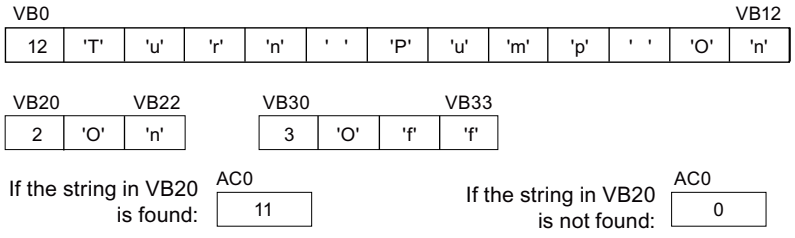
Non-fatal errors with ENO = 0	SM bits affected
• 0006H Indirect address • 0091H Operand out of range • 009BH Index = 0	None

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).

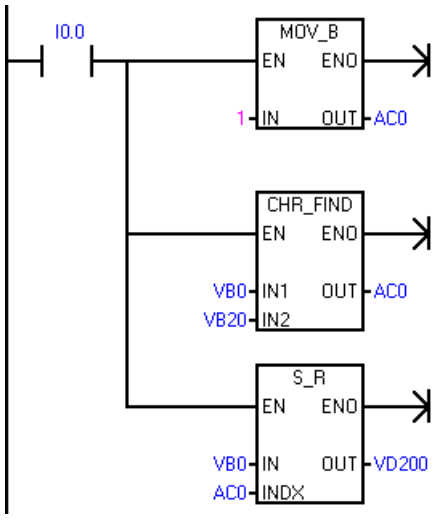
LAD		STL
	1. Set AC0 to 1. (AC0 is used as the OUT parameter.) 2. Search the string at VB0 for the string at VB20 ('On'), starting at the first position (AC0=1).	**Network 1** **LD I0.0** **MOVB 1, AC0** **SFND VB0, VB20, AC0**



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.

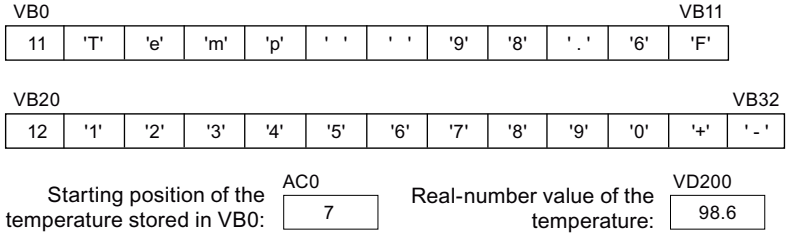
LAD



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
```

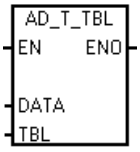


Starting position of the temperature stored in VB0: AC0 = 7

Real-number value of the temperature: VD200 = 98.6

9.15 Table

9.15.1 Add to table

LAD / FBD	STL	Description
	**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

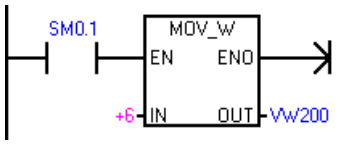
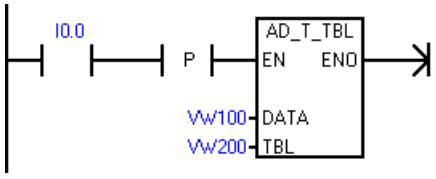
Input / output	Data type	Operand
DATA	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
TBL	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, *VD, *LD, *AC **Note**: TBL of this instruction doesn't have DB2 area support.

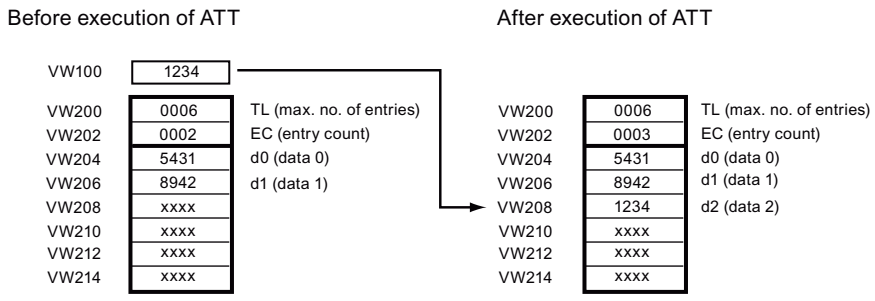
Note

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.

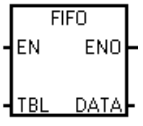
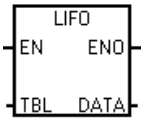
Example: Add to Table instruction

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**



9.15.2 First-in-first-out and last-in-first-out

Table 9-21 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

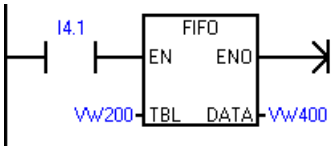
Input / output	Data type	Operand
TBL	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, *VD, *LD, *AC
DATA	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AQW, *VD, *LD, *AC

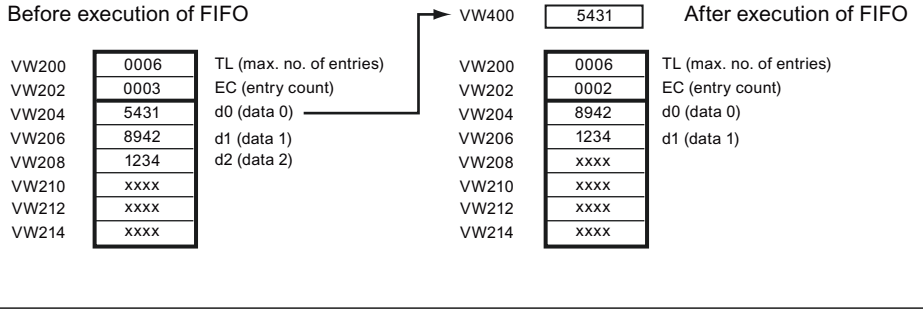
Note

All table read and table write instructions must be activated by edge trigger instructions.

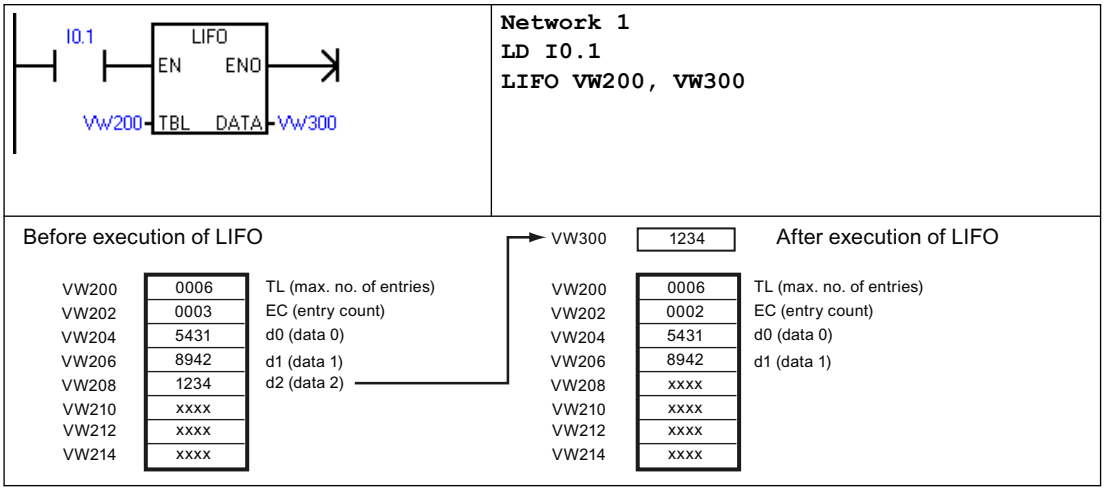
To create a table, you must first make an entry for the maximum number of table entries before any entries can be put in the table.

Example: FIFO instruction

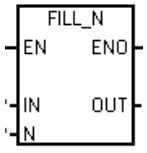
LAD	STL
	**Network 1** **LD I4.1** **FIFO VW200, VW400**



Example: LIFO instruction



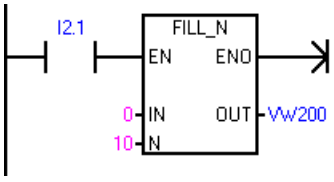
9.15.3 Memory fill

LAD / FBD	STL	Description
	`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

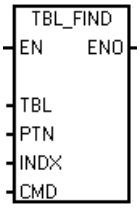
Input / output	Data type	Operand
IN	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
OUT	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AQW, *VD, *LD, *AC

Example: Memory fill instruction

LAD	STL
	**Network 1** **LD I2.1** **FILL +0, VW200, 10**



9.15.4 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

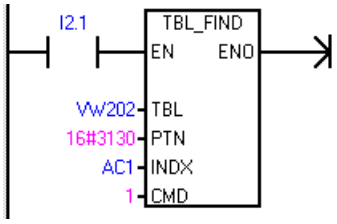
VW200	0006	TL (max. no. of entries)
VW202	0006	EC (entry count)
VW204	xxxx	d0 (data 0)
VW206	xxxx	d1 (data 1)
VW208	xxxx	d2 (data 2)
VW210	xxxx	d3 (data 3)
VW212	xxxx	d4 (data 4)
VW214	xxxx	d5 (data 5)

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.

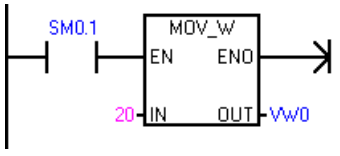
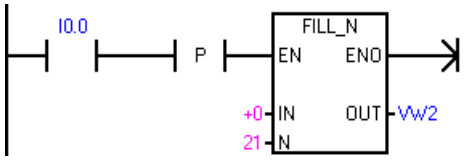
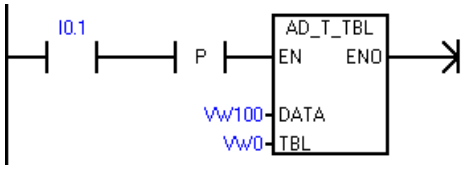
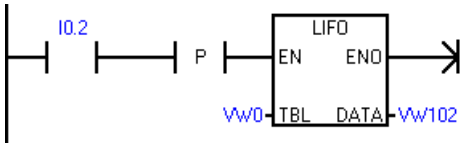
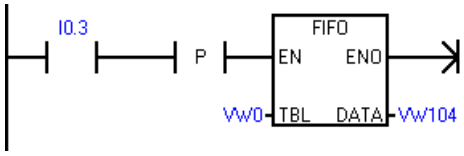
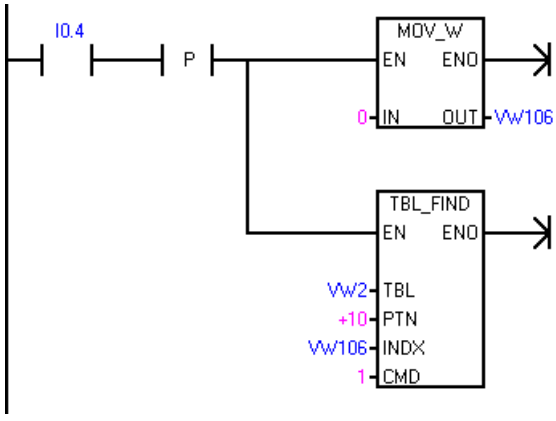
AC1 0 — Before the table can be searched again, the INDX value must be reset to 0.

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.

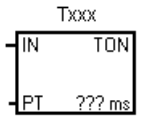
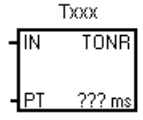
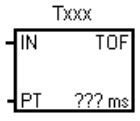
Before you work with a table, assign the maximum number of table entries. Otherwise, you cannot make entries in the table. Also, be sure that all read and write commands are activated with edge instructions.

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**

9.16 Timers

9.16.1 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.			

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.

See also

Timer programming tips and examples (Page 458)

9.16.2 Timer programming tips and examples

Timer type

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), 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.

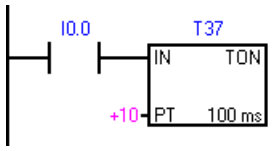
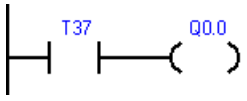
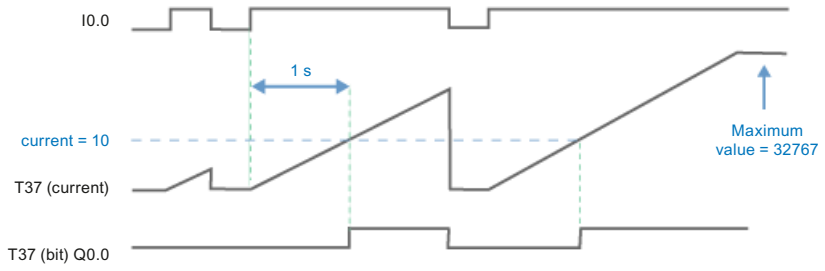
Since the timer can be started anywhere within a 10-ms interval, the preset must be set to one time interval greater than the minimum desired timer interval. For example, to guarantee a timed interval of at least 140 ms using a 10-ms timer, the preset time value should be set to 15.

100-millisecond resolution

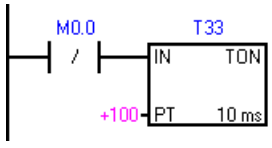
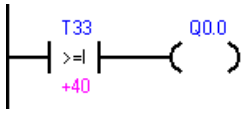
The 100-ms timers count the number of 100-ms timer intervals that have elapsed since the active 100-ms timer was last updated. Update this type of timer by the following methods: When executing the timer instruction, the accumulated number of 100-millisecond intervals (since the previous scan cycle) is added to the current value of the timer.

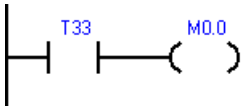
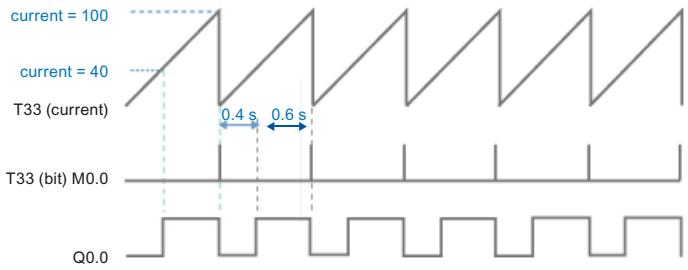
Since the timer can be started anywhere within a 100-ms interval, the preset must be set to one time interval greater than the minimum desired timer interval. For example, to guarantee a timed interval of at least 2100 ms using a 100-ms timer, the preset time value should be set to 22.

TON On-delay timer example

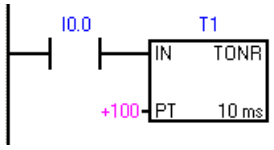
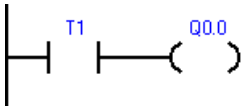
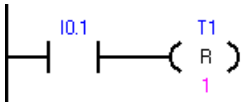
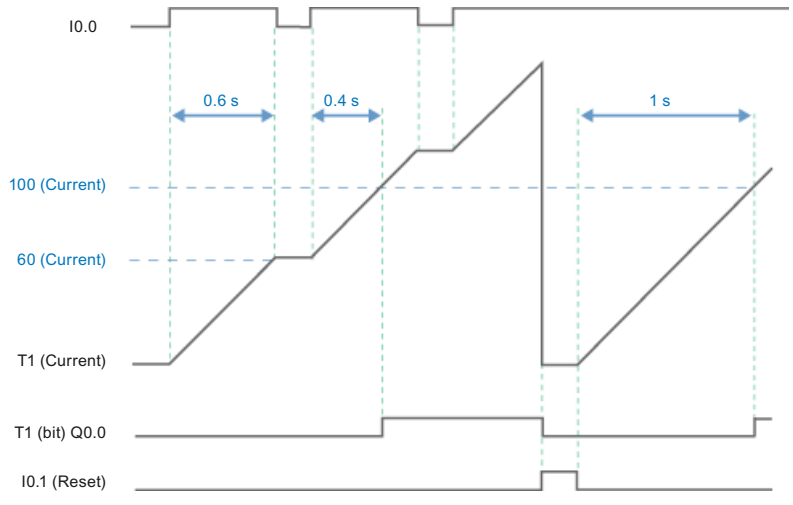
LAD		STL
	100 ms timer T37 times out after 1 s (10 x 100 ms) • I0.0 ON = T37 enabled, • I0.0 OFF = disable and reset T37	**Network 1** **LD I0.0** **TON T37, +10**
	T37 bit is controlled by timer T37	**Network 2** **LD T37** **= Q0.0**
Timing diagrams 		

TON self-resetting On-delay timer example

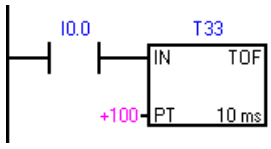
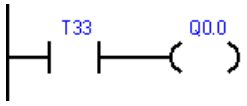
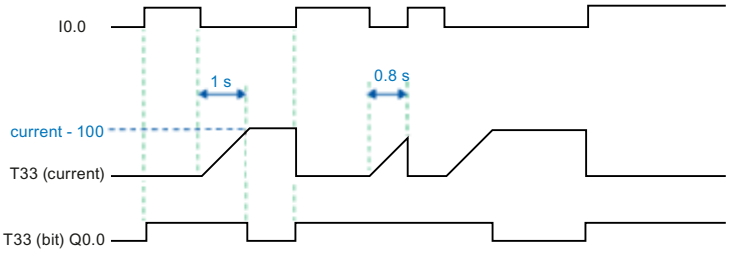
LAD		STL
	10 ms timer T33 times out after 1 s (100 x 10 ms). M0.0 pulse is too fast to monitor with Status view.	**Network 1** **LDN M0.0** **TON T33, +100**
	The Compare contact becomes TRUE at a rate that is visible in Status view. After (40 x 10 ms), Q0.0 turns on, with a signal waveform of 40% low level and 60% high level	**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 diagrams 		

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 diagrams 		

TOF off-delay timer example

LAD		STL
	10-ms timer T33 times out after 1 s (100 x 10 ms). • I0.0 rising edge = T33 enabled, • I0.0 falling edge = 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 diagrams 		

Effect of timer resolution on when timer bits and current time values are updated

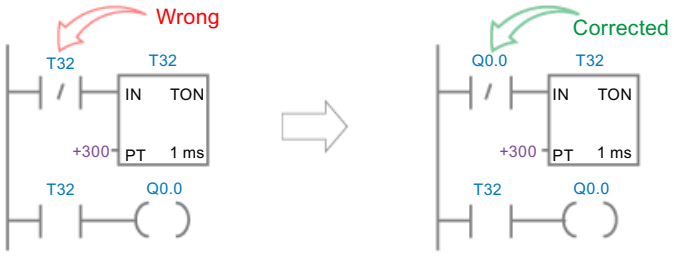
- **1 ms** timer: The timer bit and the current value are updated asynchronous to the scan cycle. For scans greater than 1 ms, the timer bit and the current value are updated multiple times throughout the scan.
- **10 ms** timer: The timer bit and the current value are updated at the beginning of each 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: For timers with a resolution of 100 ms, 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.

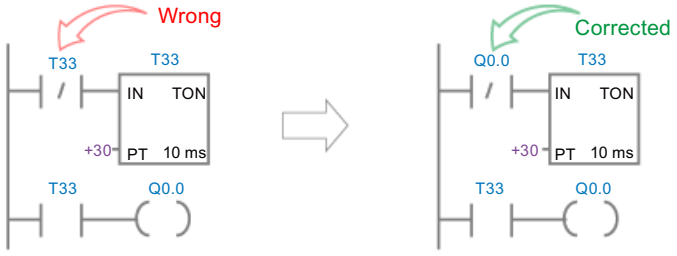
1 ms timer

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.



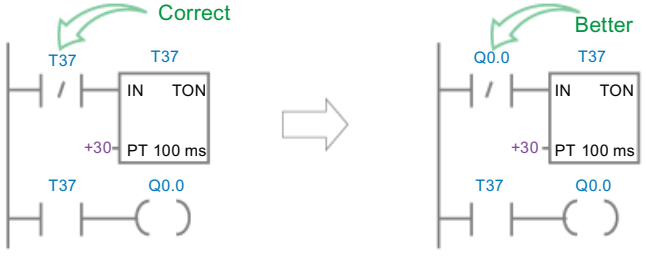
10 ms timer

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.

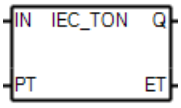
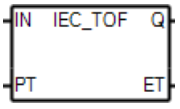
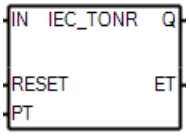


100 ms timer

Q0.0 is always turned ON for one scan whenever the timer's current value reaches the preset value.



9.16.3 Timer instructions (IEC)

LAD/FBD	STL	Description
	**IECTON xxx, PT, ET**	The "On Delay" instruction can be used to delay the setting of output Q by a preset time PT. The instruction is started when the logic operation result of input IN changes from "0" to "1" (rising edge). When the instruction is initiated, ET starts timing. After time PT has elapsed, the signal state of output Q will change to "1". Output Q remains set as long as the start input is still "1". When the signal state of the start input changes from "1" to "0", the Q output is reset. The timer function is started again when a new rising edge is detected at the start input. The current time value can be queried at the ET output. The timer value starts from 0 and can accumulate to a maximum of 2147483647. ET will be reset to zero as soon as the input IN signal status changes to "0". The resolution of IEC_TON is 1 ms.
	**IECTOF xxx, PT, ET**	The "Off Delay" instruction can be used to delay the reset of output Q by a preset time PT. When the logical operation result of input IN changes from "0" to "1" (rising edge), output Q will be set. When the input IN signal changes back to "0" (falling edge), ET starts timing. As long as ET is still timing, output Q will remain set. When ET accumulates to PT, the timer ends and the output Q is reset. If the signal state of input IN changes to "1" before the ET timer ends, the timer is reset, and the signal state of output Q will remain "1". The current time value can be queried at the ET output. The timer value starts from 0 and ends after reaching PT. When the ET timer ends, the output ET will remain in the state it was in when the timer ended until the input IN changes back to "1". If the signal state of the input IN switches to "1" before the ET timer ends, the ET will be reset to 0. The resolution of IEC_TOF is 1 ms.
	**IECTONR xxx, RESET, PT, ET**	The "Time Accumulation" instruction allows you to accumulate time values within the time range set by parameter PT. When the signal state of the input IN changes from "0" to "1" (rising edge), the time measurement is started and ET begins timing. During the timing period, the system will continuously accumulate the time value when the input IN is in the "1" state. The accumulated time is displayed and can be queried through the output ET. When ET reaches the set time PT, the output Q will be set to "1". Even if the input IN signal changes from "1" to "0" (falling edge), Q remains set. Inputting RESET will force a reset of outputs ET and Q, regardless of the state of the start input signal. The resolution of IEC_TONR is 1 ms.

Input/output	Data type	Operands
IN	BOOL	Power flow
PT	DWORD	I, Q, V, M, SM, T, C, L, AC, *AC, constant, global variables, FB member variables
RESET	BOOL	I, Q, V, M, SM, S, T, C, L, global variables, FB member variables
Q	BOOL	Power flow
ET	DWORD	I, Q, V, M, SM, T, C, L, AC, *AC, global variables, FB member variables

9.16.4 Timer programming tips and examples (IEC)

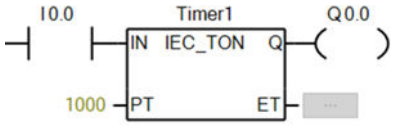
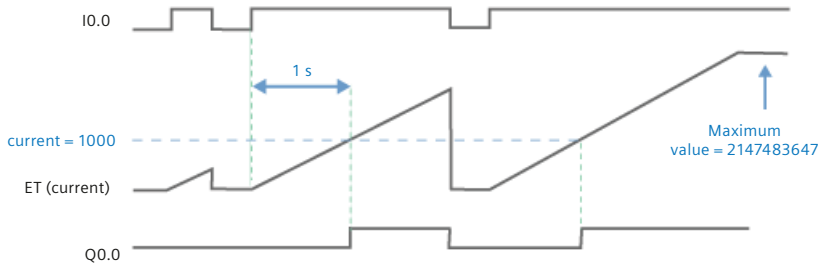
Timer type

You can use timers to implement time-based counting functions. The S7-200 SMART instruction set provides three different types of IEC timers.

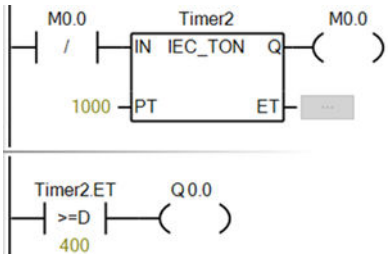
- On-Delay Timer (IEC_TON) for timing a single interval
- Retentive On-Delay Timer (IEC_TONR) for accumulating a number of timed intervals
- Off-Delay Timer (IEC_TOF) for extending time past an off (or FALSE), such as for cooling a motor after it is turned off

Example

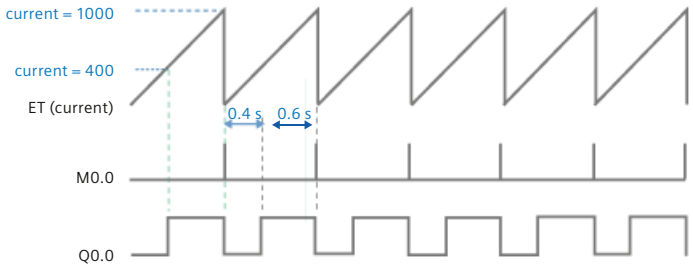
TON On-delay timer example

<table>
<tr><th>LAD</th><th></th><th>STL</th></tr>
<tr><td> 
</td><td>The timer times out after 1000ms (1s).
• I0.0 ON = Timer Enable
• I0.0 OFF = disable and reset timers</td><td>Network 1
LD I0.0
IECTON Timer1, Timer1.PT, Timer1.ET
= Q0.0</td></tr>
<tr><td colspan="3">Timing diagrams
 
</td></tr>
</table>

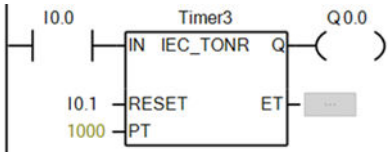
TON self-resetting On-delay timer example

LAD		STL
	After 1000 ms, the timer times out. After the timeout, M0.0 is set, which resets the timer. The M0.0 pulse speed is too fast to be monitored using the Status view. Turn on Q0.0 after 400ms for a 40% OFF / 60% ON waveform.	**Network 1** **LDN M0.0** **IECTON Timer2, 1000, Timer2.ET** **= M0.0** **Network 2** **LDD>= Timer4.ET, 400** **= Q0.0**

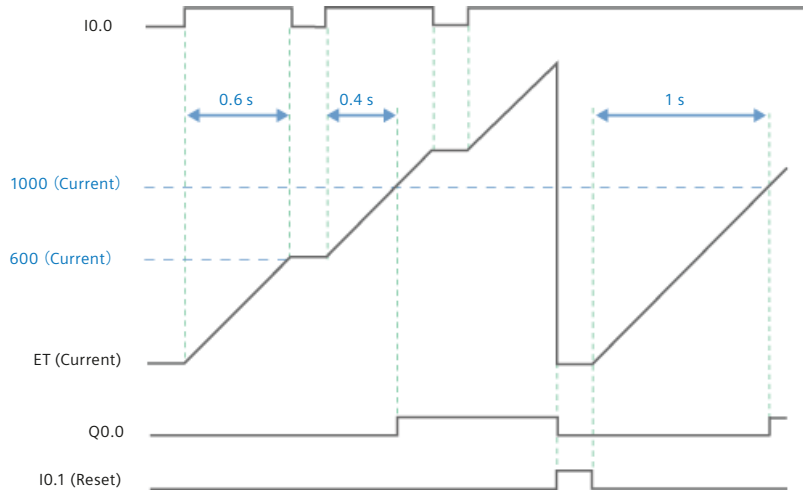
Timing diagrams



TONR retentive on-delay timer example

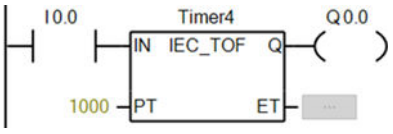
LAD		STL
	Q0.0 is ON after the timer accumulates a total of 1 second. When I0.1 is turned on, the timers (ET and Q) are reset.	**Network 1** **LD I0.0** **IECTONR Timer3, I0.1, 1000, Timer3.ET** **= Q0.0**

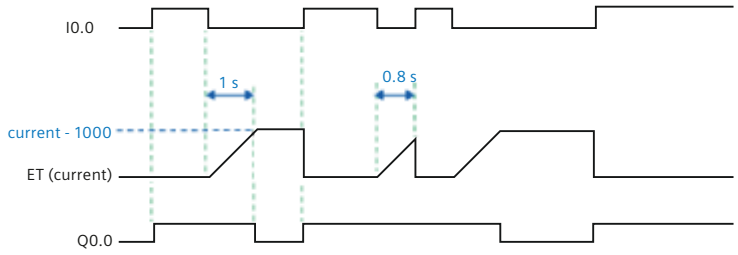
Timing diagrams



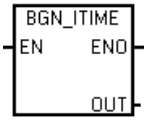
TOF off-delay timer example

LAD		STL

	The timer timed out after 1000ms.
	• Rising edge of I0.0 = Timer reset, Q0.0 set.
	• Falling edge of I0.0 = Timer starts counting, Q0.0 is reset after timeout.
		`Network 1`
		`LD I0.0`
		`IECTOF Timer4, 1000,`
		`Timer4.ET`
		`= Q0.0`

Timing diagrams 		

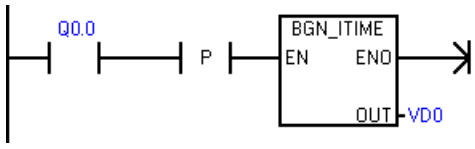
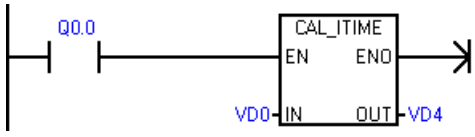
9.16.5 Interval timers

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**

9.17 Subroutine

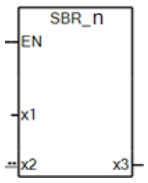
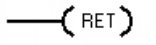
9.17.1 CALL (subroutine) and RET (conditional return)

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	

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, *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

9.17.2 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

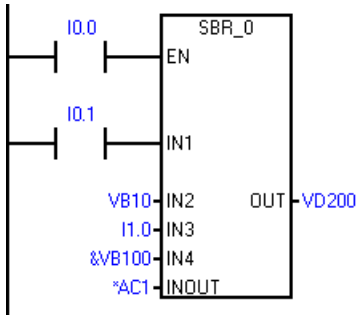
- Power Flow: Boolean power flow is allowed only for bit (Boolean) inputs. This declaration assigns an input parameter to the result of power flow based on a combination of bit logic instructions. Power flow inputs are similar to the EN input in that they connect to bit logic (for ex. LAD contacts) and 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.

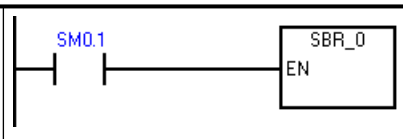
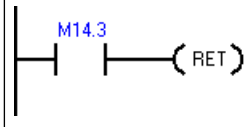
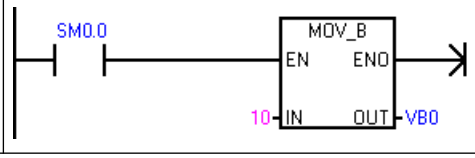
The data element size and type are represented in the coding of the parameters. Assignment of parameter values to local memory in the subroutine is as follows:

- 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`

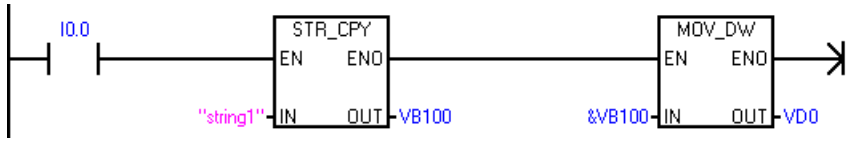
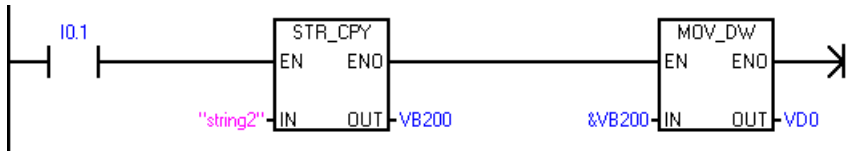
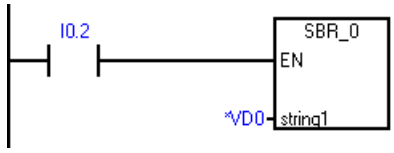
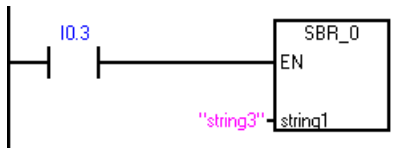
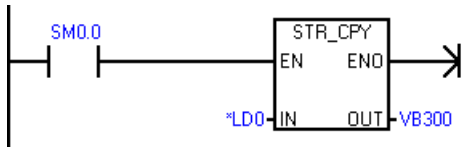
Example: Subroutine and return from subroutine instructions

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**

9.18 Function block

9.18.1 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.

9.18.2 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				

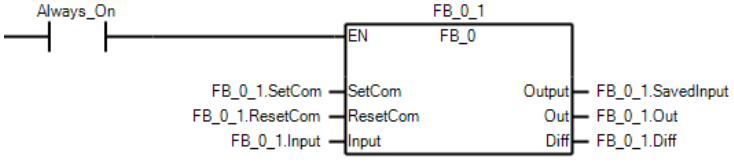
Example: function block call with call parameters

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	

J 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:

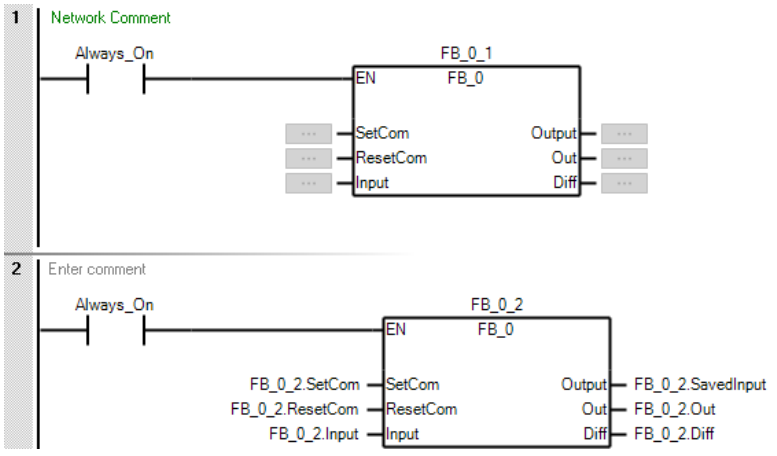
- Parameter values are assigned to local memory in the order specified by the call function block instruction with parameters starting at L 0.0.
- One to eight consecutive bit parameter values are assigned to a single byte starting with Lx.0 and continuing to Lx.7.
- Byte, word, and double word values 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.

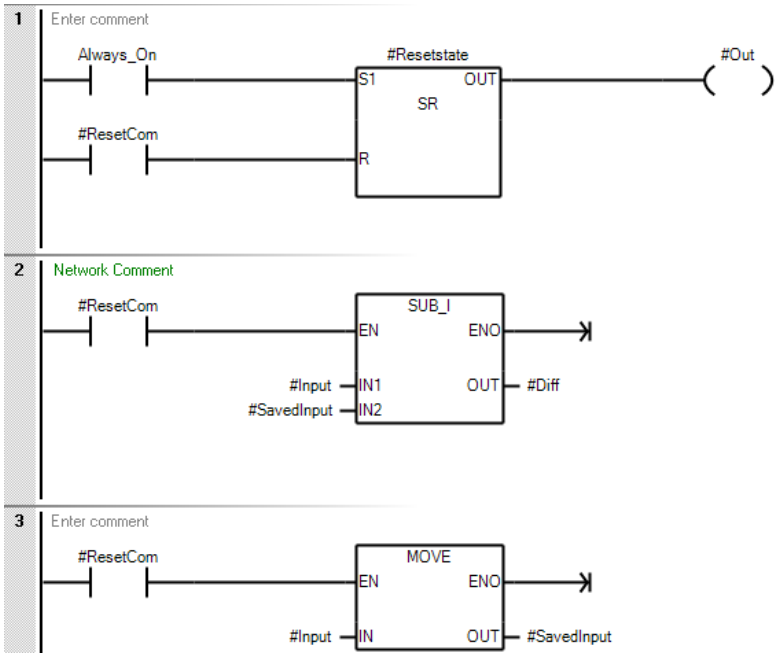
If you are programming in STL, the format of the CALL instruction is:

```
CALL function block number, parameter 1, parameter 2, ... ,
parameter 16
```

Example: function block call using string parameter

<table>
<tr><th>LAD</th><th>STL</th></tr>
<tr><td>MAIN[1]</td><td>


</td><td>
<pre>
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
</pre>
</td></tr>
<tr><td>FB0</td><td>


</td><td>
<pre>
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
</pre>
</td></tr>
</table>

[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.

9.19 PROFINET

9.19.1 Features of the programming instruction "PROFINET"

There are two groups of programming instructions under the folder "PROFINET" :

- RDREC and WRREC (Page 481): 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 485): Read multiple bytes immediately from PROFINET device or write multiple bytes immediately to PROFINET device.

Note

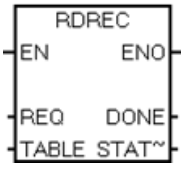
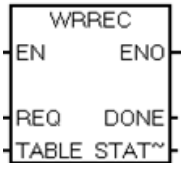
For any legacy instruction that can access the I or Q memory area, it accesses to the I or Q memory area of a PROFINET network.

9.19.2 Read and Write data record

9.19.2.1 Input and output interface of RDREC and WRREC instruction

The RDREC and WRREC instructions are as follows:

Table 9-22 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 9-23 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 482).
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 483).

Note

The supported maximum data record length is 1024 bytes.

Definition of "TABLE" parameters

The following table lists the parameter information of "TABLE":

Table 9-24 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 15 16 17	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.
18 19	Actual record data length	Output	This parameter is valid for the instruction RDREC and returns the actual data length specified by the device.
20 21 22 23	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)*.

Definition of "STATUS" parameters

The following table lists the parameter information of "STATUS":

Table 9-25 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 483).

9.19.2.2 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.

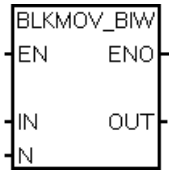
Error code	Description
6	An instance mismatch: The connection is busy with another instance, whose device number, API number, slot number and subslot number are same as the requested instance, but with a different buffer size and data address.
7	The PROFINET device is not connected.
8	The size of the 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.

9.19.3 Read and Write multiple bytes between physical PROFINET and memory address

9.19.3.1 Input and output interface of BLKMOV_BIR and BLKMOV_BIW

The BLKMOV_BIR and BLKMOV_BIW instruction is as follows:

Table 9-26 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

9.19.3.2 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 10

10.1 Overview

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
- RS485:
  - Provides a STEP 7-Micro/WIN SMART V3 connection for programming when using a USB-PPI cable

    Note

    A direct connection to the USB-PPI cable is not supported by the S7-200 SMART G2 CPU. To connect an S7-200 SMART G2 CPU with this cable, you will need to make your own RS485 connection wire.

  - 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)

10.2 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/S7-200 SMART G2 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/S7-200 SMART G2 CPUs or network devices.
  - PROFINET connections: Each PROFINET controller can support ten connections (IO device or drive).

Note

You can program an S7-200 SMART V3.0/S7-200 SMART G2 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 and the signal board port.
  Note: A direct connection to the USB-PPI cable is not supported by the S7-200 SMART G2 CPU. To connect an S7-200 SMART G2 CPU with this cable, you will need to make your own RS485 connection wire.
- Use the RS485 and RS232 ports for HMI access (Data read/write) and Freeport communications.

- CM01 Signal Board (SB) RS232/RS485 port (Port 1, Port 2): Four connections to support HMI devices.

10.3 CPU communication ports

S7-200 SMART V3 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

    Note

    A direct connection to the USB-PPI cable is not supported by the S7-200 SMART G2 CPU. To connect an S7-200 SMART G2 CPU with this cable, you will need to make your own RS485 connection wire.

  - TDs/HMI: RS485 type
  - Freeport (XMT/RCV) including Siemens-provided USS and Modbus RTU libraries
- RS485/RS232 signal board (SB):
  - TDs/HMI: RS485 or RS232 type
  - Freeport (XMT/RCV) including Siemens-provided USS (RS485 only) and Modbus RTU (RS485 or RS232) libraries

10.4 Ethernet

10.4.1 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 55).
- 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.

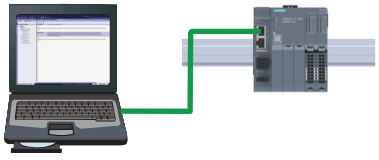
If the connection is terminated (for example, due to a line break or one of the partners breaks the connection), the active partner attempts to re-establish the connection. The passive device will also note the termination of the connection and take actions (for example, revoking the password privileges 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.

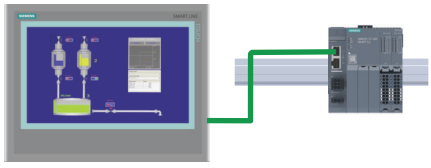
10.4.2 Sample Ethernet network configurations

You have three different types of communication options when using the S7-200 SMART 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



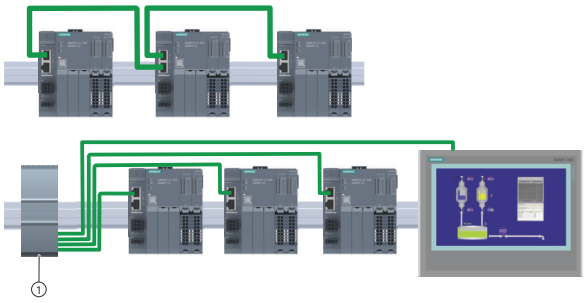
HMI connected to the CPU



A CPU connected to another CPU

Ethernet switching

S7-200 SMART 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.

10.4.3 Assigning Internet Protocol (IP) addresses

10.4.3.1 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.

10.4.3.2 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 indicates the network ID (which network you are on), and the second part indicates the host ID (which is 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

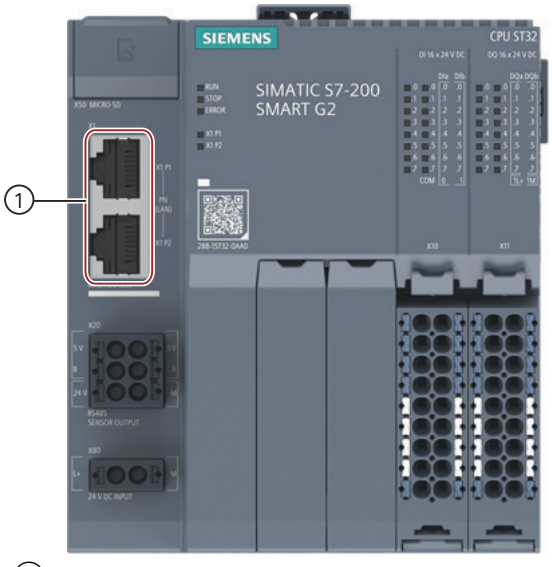
  You must have a unique IP address for each device on your network.

- "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) ports

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

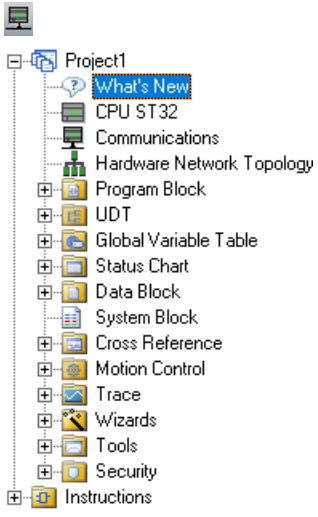
You can have static or dynamic IP information in the CPU:

- 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. Moreover, if you want to change the IP information, you can only do so in the "System block" dialog and download it to the CPU again.
- 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:



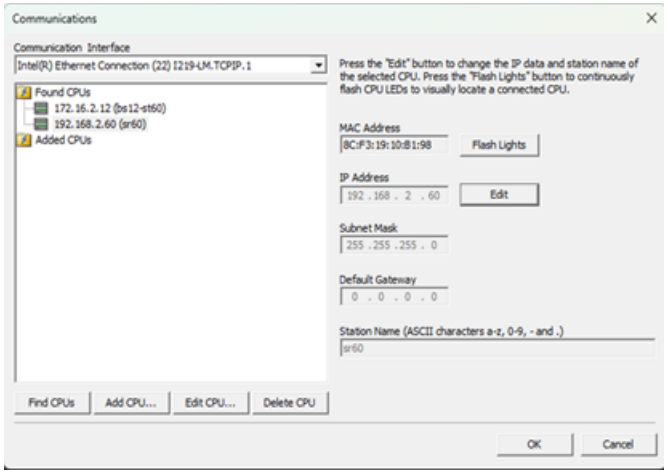
- In the Navigation bar, click the "Communications" button.
- In the Project tree, select the "Communications" node, then press Enter; or double-click the "Communications" node.

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:

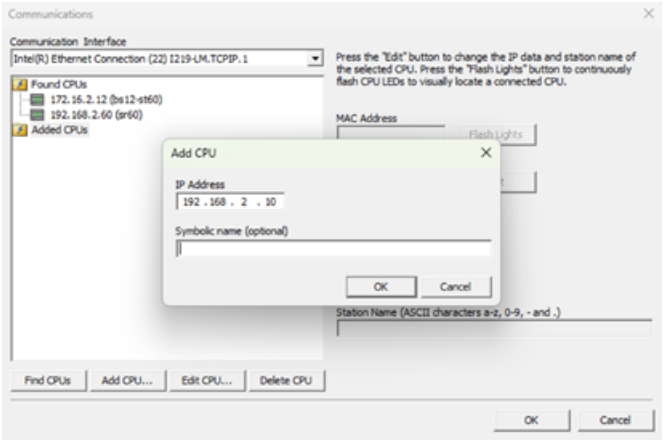
- Click the "Communication Interface" dropdown list, and select the "TCP/IP" Network Interface Card (NIC) for your programming device.
- Click the "Find CPU" button to display all operational CPUs ("Found CPU") on the local Ethernet network. All CPUs have a default IP address.
- Highlight a CPU, and 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:

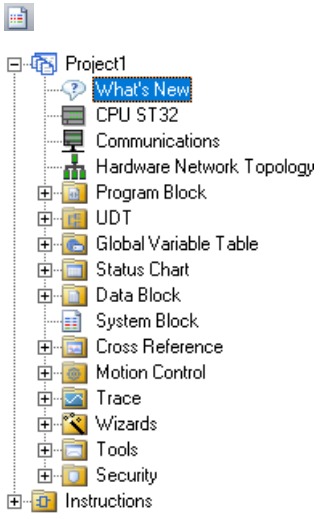
- Click the required CPU.
- If you need to identify which CPU to configure or change, click the "Flash Lights" button. This button flashes the "STOP", "RUN", and "FAULT" lights for the highlighted CPU in the list.
- Click the "Edit" button to make changes in the IP information.
- Change the following 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.

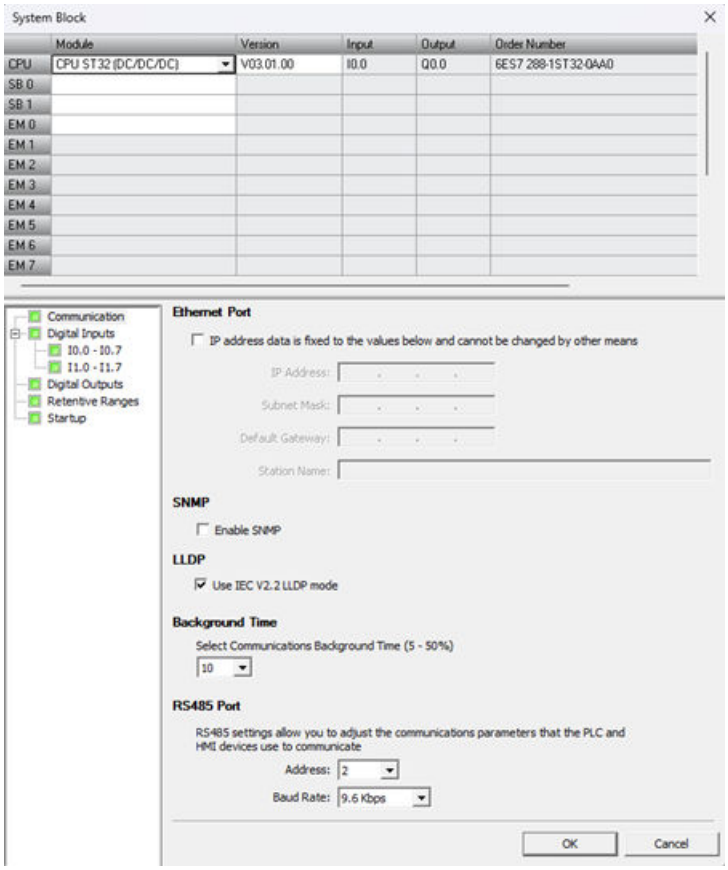
To enter or change IP information, perform the following:

- 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.



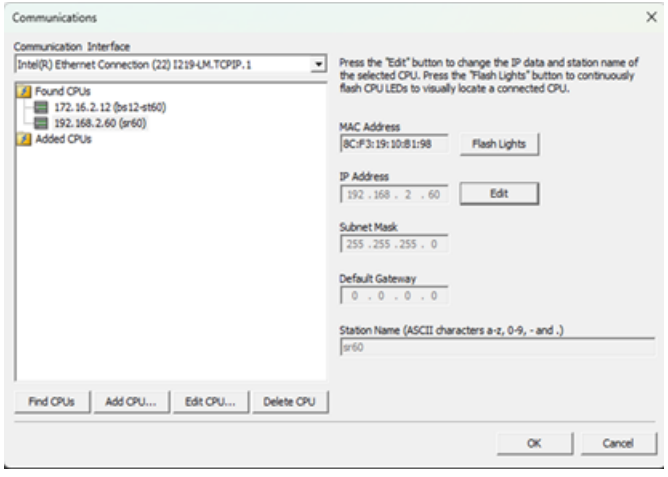
When you check the "IP address data is fixed to the values below and cannot be changed by other means" checkbox in the system block dialog, the IP information that you enter for the onboard Ethernet port is static. Static IP information must be downloaded to the CPU before it is active in the CPU. If 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.

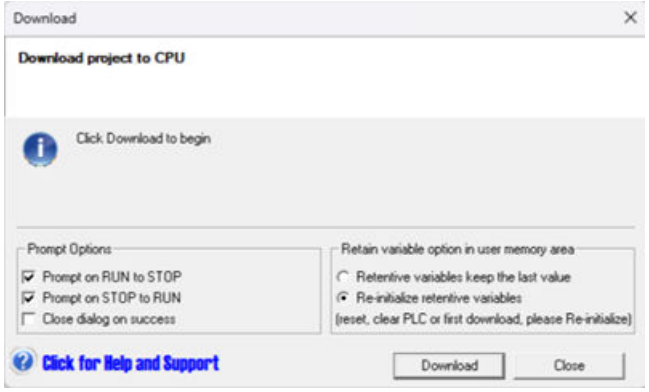
Note

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.

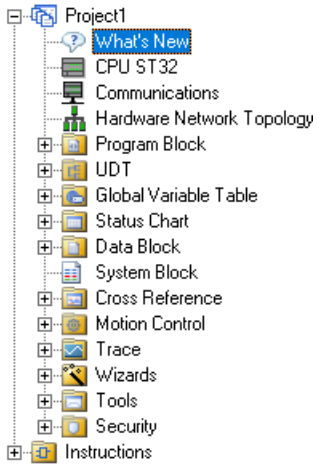
Refer to the "Get IP address and set IP address (Ethernet)" instructions for more information.

10.4.3.3 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

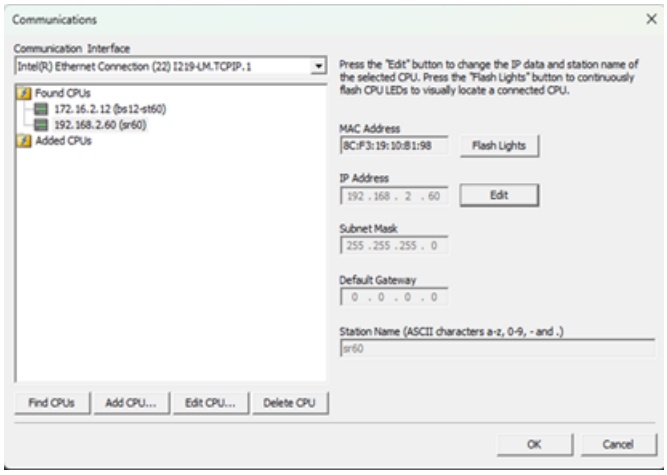


Communications from the Component drop-down list in the Windows area of the View menu ribbon strip

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.



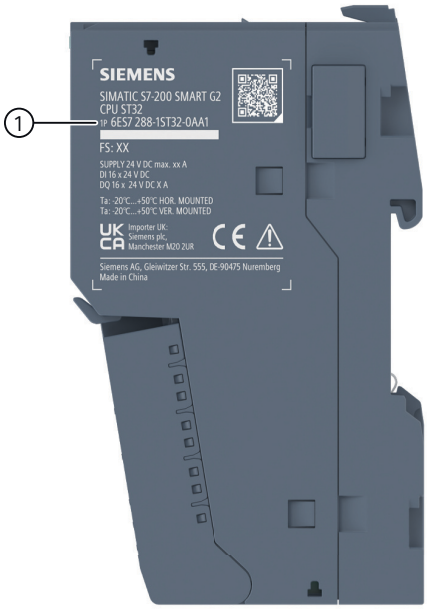
10.4.4 Locating the Ethernet (MAC) address on the CPU

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.

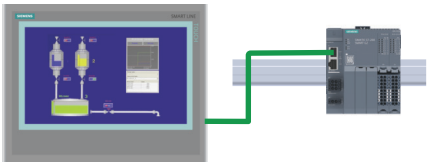
Note

Each CPU is loaded at the factory with two permanent, unique MAC address. You cannot change the MAC address of a CPU.



① MAC address

10.4.5 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 10-1 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" 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 492) 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" for more information.

10.4.6 Open user communication

10.4.6.1 Protocols

Overview

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

UDP (User Datagram Protocol)

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.

TCP (Transmission Control Protocol)

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

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.

10.4.6.2 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.

10.4.6.3 Ports and TSAPs

Overview

Ports and Transport Service Access Points (TSAPs) provide for the routing of messages to the proper receiver within the CPU or other device.

Ports

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.

TSAPs

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.)

10.4.7 PROFINET Communication

10.4.7.1 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 10 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

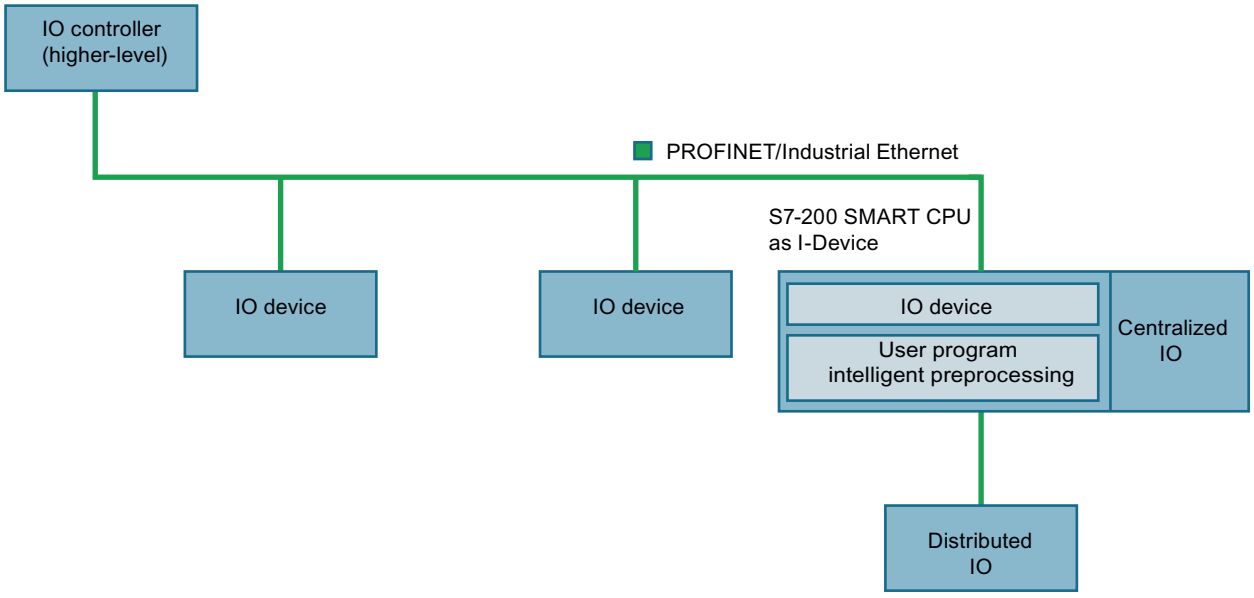
10.4.7.2 I-Device

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.

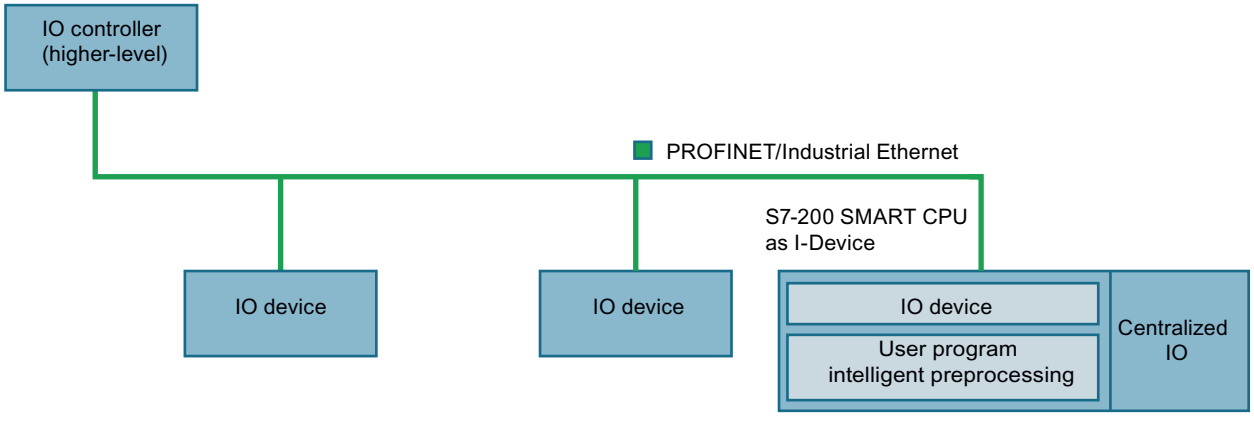


Two types of I-Device are supported for S7-200 SMART V3.0 or later version:

- 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

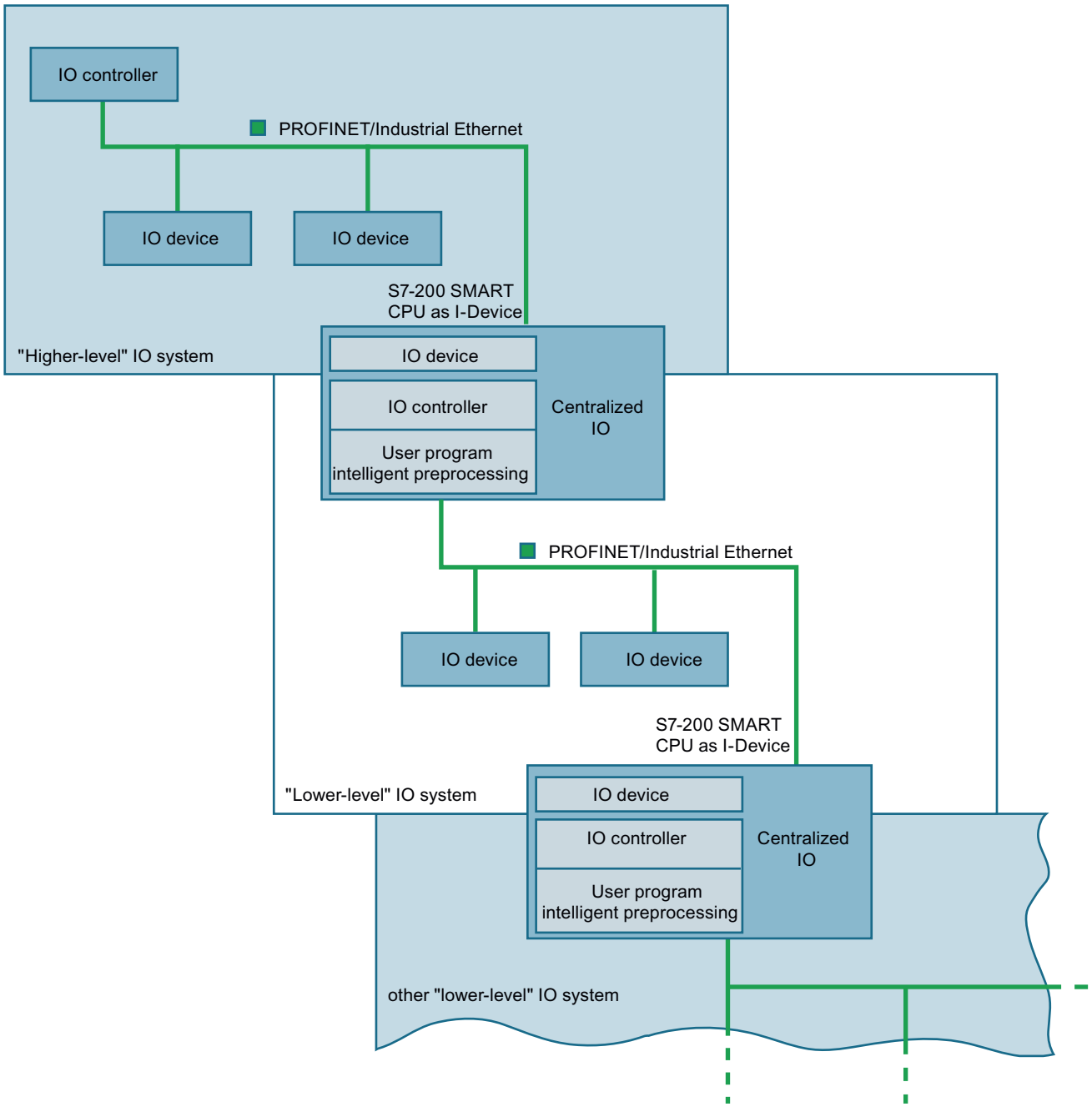
In addition to the role as an IO device, an I-Device can also work as an IO controller on a PROFINET interface.

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 10, and only one higher-level controller is supported.



10.4.7.3 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 2.45 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 514).

Before adding an IO device to the PROFINET network, you can import its GSDML files to STEP 7-Micro/WIN SMART V3.

10.4.7.4 PROFINET device naming

Assigning device names

The devices on your PROFINET network must have an assigned name before connecting to the CPU. Use the "Find PROFINET Device" to assign names to your PROFINET devices if the devices have not already been assigned a name or the name of the device need to be changed.

For each PROFINET device, you must assign the same name in both PROFINET wizard and "Find PROFINET Device" dialog. If a name is missing or does not match in these two place, the PROFINET IO data exchange mode will not run.

10.4.7.5 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.

10.4.7.6 Configuration

Open hardware network topology

You can use the hardware network topology to configure, assign parameters, and link individual PROFINET hardware components.

Creating a PROFINET network

A PROFINET system is comprised of a PROFINET controller and its assigned PROFINET devices. You can select different roles for CPU to configure a PROFINET network.

The PROFINET wizard guides you through the following tasks to create a PROFINET network:

- Configuring CPU as a controller (Page 514)
- Configuring CPU as an I-Device without lower-level PROFINET IO system (Page 522)
- Configuring CPU as an I-Device with lower-level PROFINET IO system (Page 526)

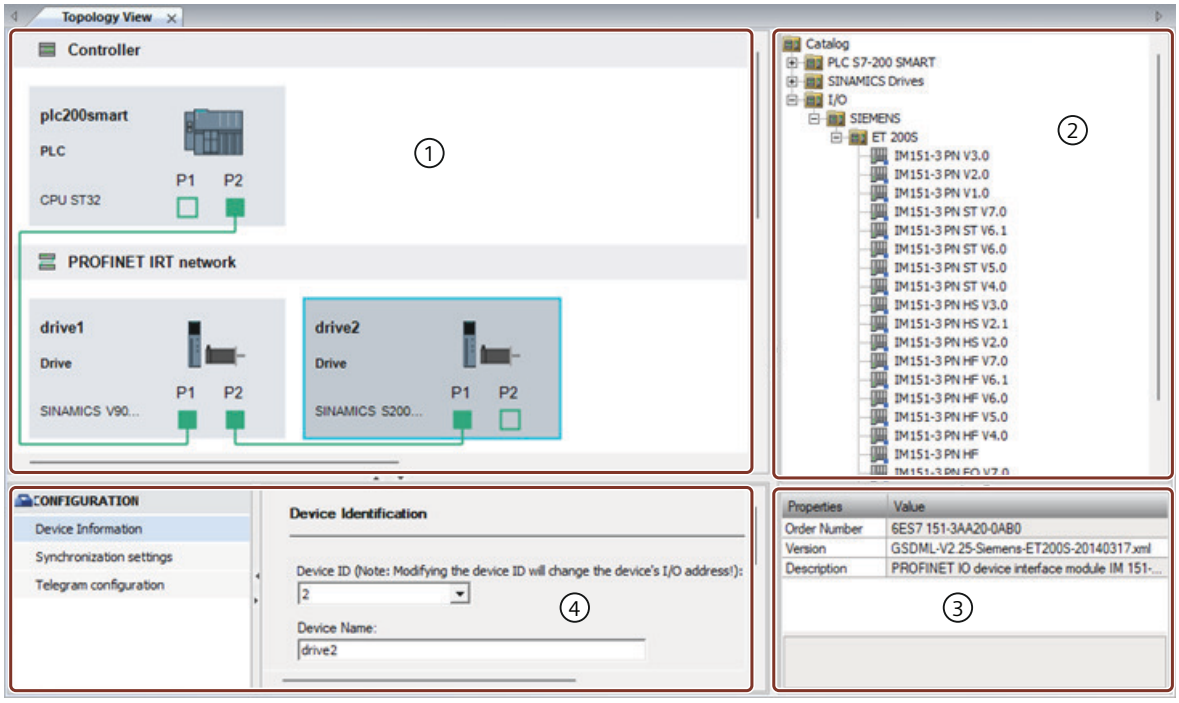
Hardware Network Topology

Hardware network topology functions

Hardware network topology is used for configuring PROFINET RT+IRT networks. Double-click "Hardware Network Topology" in the project tree to open the hardware network topology. Hardware network topology is used to configure and assign parameters to devices and modules.

Hardware network topology structure

The hardware network topology consists of the following parts:



① Topology View (Page 791)
② Device Catalog
③ Device Properties
④ Device Configuration

You can use the mouse to change the separation between areas in the network view. To change the position of the divider, click between the area dividers and change the spacing by moving the divider to the left or right while keeping the mouse button pressed. The Speedy Splitter (the two small arrow keys) allows you to use a single click to minimize a view, maximize a view or restore the last selected split.

To efficiently manage and configure network resources, hardware topology networks provide two views: Topology view and device view. In the topology view, you can configure the entire network topology. When you need to edit and configure a single device and its internal modules, you must enter the device view.

Tool bar

The toolbar provides the following functions:

Icon	Meaning
	Set PLC to RUN mode
	Set PLC to STOP mode
	Compile
	Upload project from PLC to STEP 7-Micro/WIN SMART V3
	Download from STEP 7-Micro/WIN SMART V3 to a PLC
	Topology Comparison
	Scan topology Initiate detection of the current online topology.

Device Catalog

When you open the hardware network topology, the device directory tree will automatically load the following devices:

- PROFINET device embedded in the CPU
- PROFINET devices supported by imported GSD files

Device Properties

When you select a device in the device catalog, the device properties window will display the device's basic properties.

Property	Description
Order number	Read only
Edition	You can select the GSD file version for your device via the drop-down menu. If multiple GSD files containing the same device are installed on your system, you can select one here. After completing the setup, the device version in the tree view will be updated to the version you selected. By default, the system will automatically select the highest version of the GSD file.
Description	Read only

Configuring CPU as a controller

Adding a device to PROFINET network

When IRT and RT devices are added to the topology view, the system automatically assigns a unique ID to these devices. The ID allocation process will start with the smallest available ID and allocate it to each IRT/RT device in ascending order.

For devices added to the PROFINET network, their names will be automatically generated according to the order in which they were added.

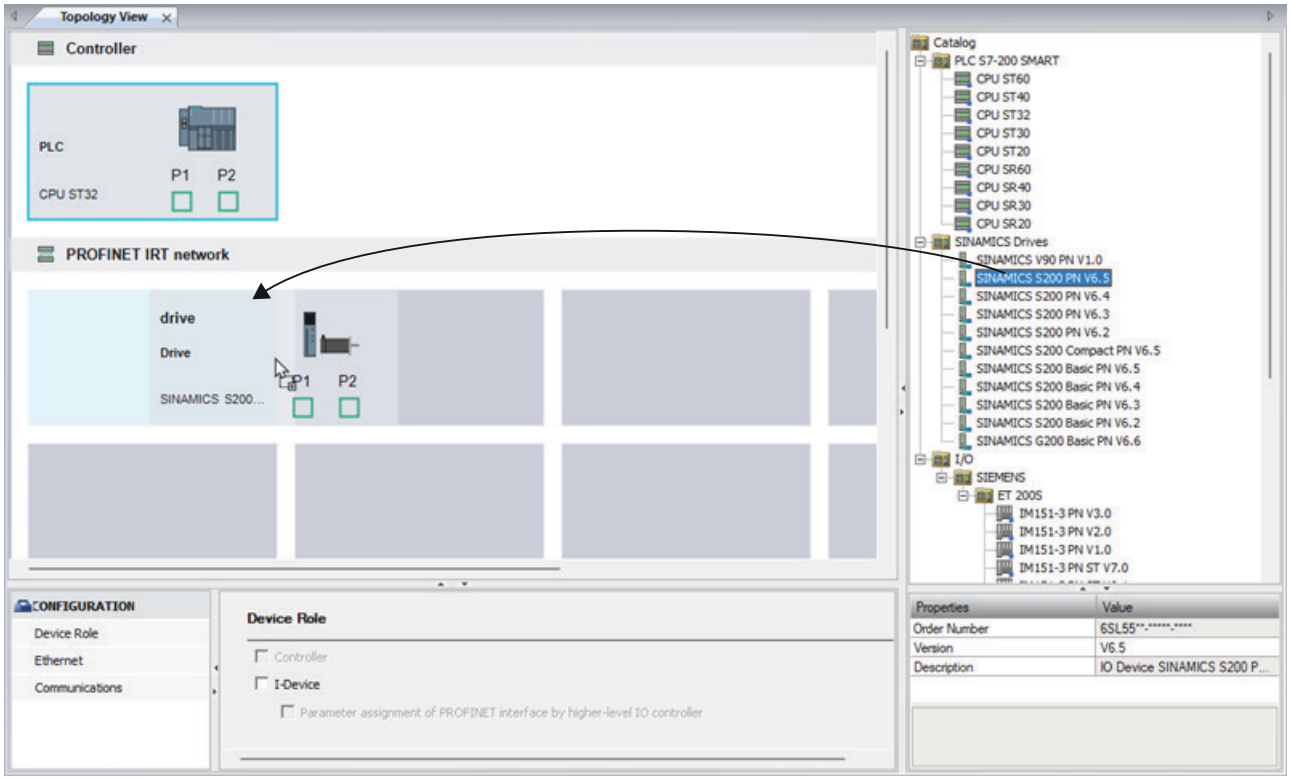
- The drive naming format is drive1, drive2, drive3...
- The I/O module naming format is dev1, dev2, dev3...
- The PLC module naming format is PLC1, PLC2, PLC3...

Dragging and dropping the device

To use the project tree to add a device to the hardware configuration, proceed as follows:

1. Open the hardware network topology.
2. In the device catalog, select the required device.
3. Click on the chosen device to select it.

4. If necessary, set the firmware version through the drop-down list in the device properties window.
5. Drag-and-drop the device into the topology view.

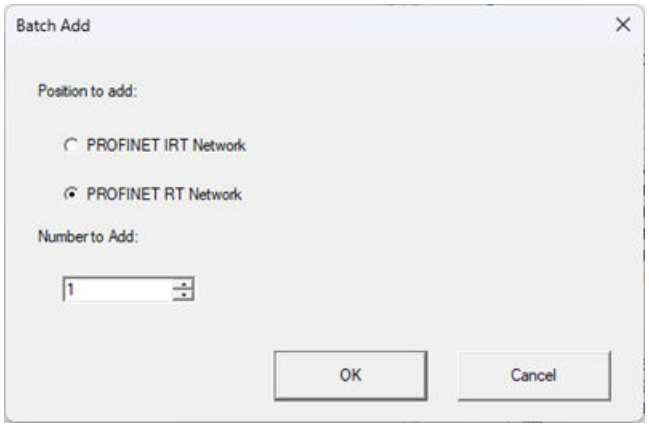


The device's drag-and-drop network depends on the PROFINET type it supports:

- RT device: It can only be dragged to RT networks, not IRT networks.
- IRT device: It can only be dragged to the IRT network.
- Devices that support both RT and IRT: You can drag it to an RT network or an IRT network.

Batch Add

1. In the device tree view, you can add devices in bulk using the "Batch Add" option in the right-click menu.
2. Select the location for adding devices in bulk and the quantity to be added.



3. Click "OK".

PROFINET device variables

When adding PROFINET devices to the configuration of the hardware network topology, STEP 7-Micro/WIN SMART V3 will add PROFINET device variables to the PROFINET device variable table.

Configuring a controller

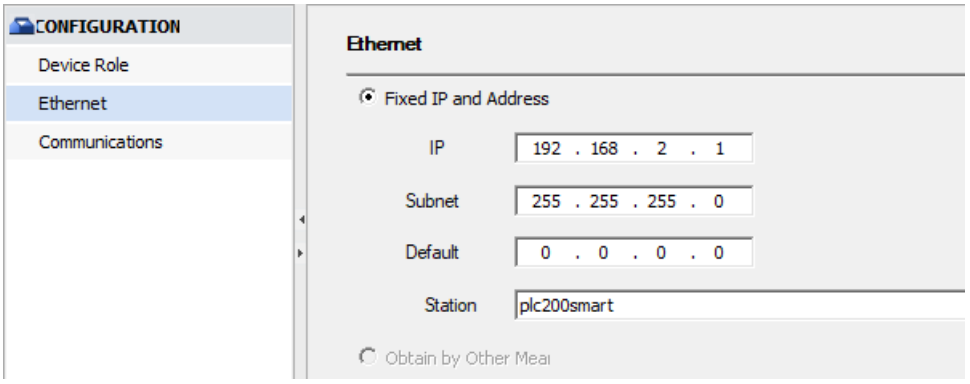
To change the properties and parameters of a controller in the topology view, follow these steps:

1. In the topology view, select PLC in the controller layer.
2. Click "Device Role" in the configuration window.

Note

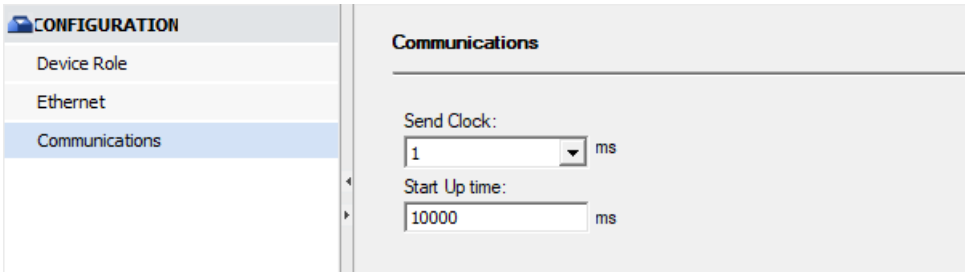
When configuring the CPU as a controller, please ensure that the GSDML file for the PROFINET device has been imported.

3. Click "Ethernet" in the configuration window.



Ether-net	• When acting as a controller: A static IP address must be configured. • When acting as an I-device: Supports fixed IP or dynamic allocation
Station Name	If you change the station name, the device name will be updated in the device view.

4. In the configuration window, click "Communication" and assign "Send Clock" and "Start Up time" to the controller.



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:

– 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" expires, the CPU switches to RUN mode at the end of the "Start Up time".

Note

If the current project's "Send clock" is set to 1ms, when you add an IRT device with a minimum send clock of 2 ms, the system will automatically adjust the "Send clock" to 2 ms and provide a corresponding prompt.

Configuring IRT devices

After inserting a device into the hardware network topology, you can edit the device's default properties in the topology view.

To change device properties and parameters in the topology view, follow these steps:

1. In the topology view, select the device in the IRT layer.
2. Click "Device Information" in the configuration window.

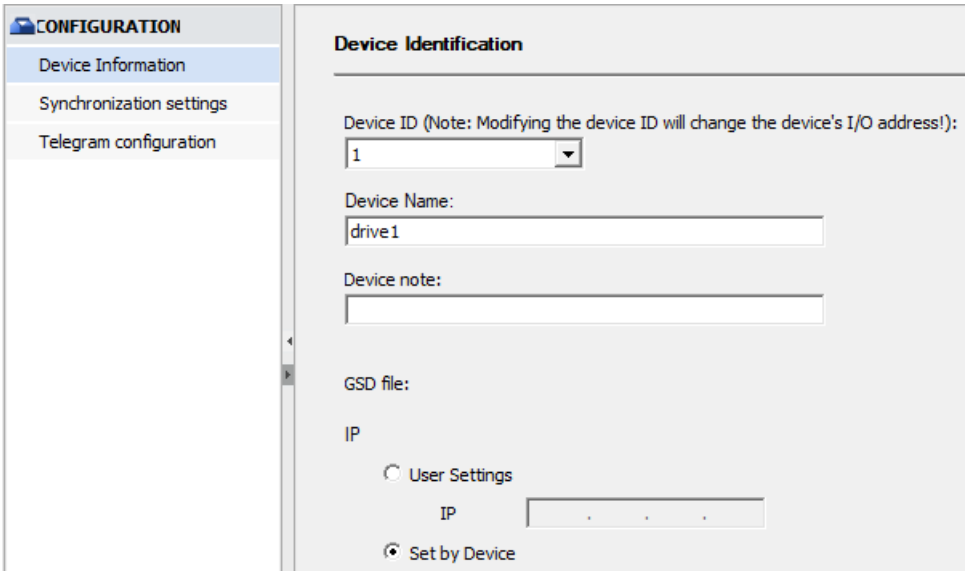
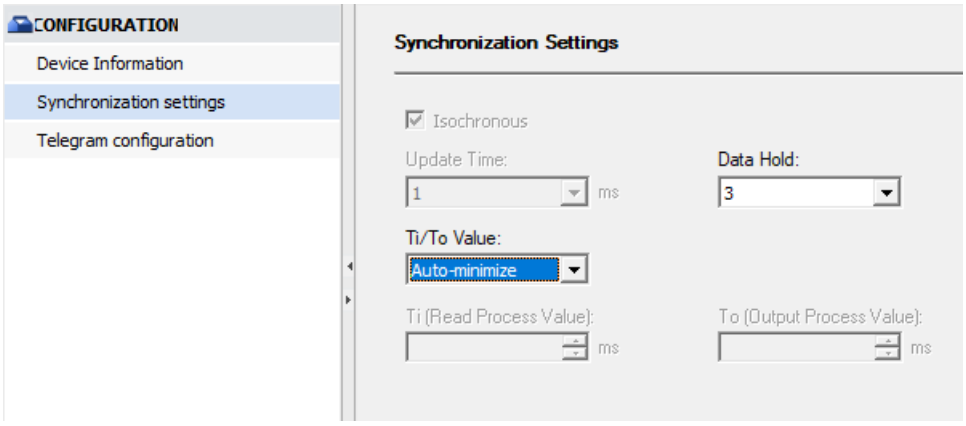


Table header	Meaning
Device ID	When the module connects to the PLC via PROFINET, the PROFINET device number is used.
Device Name	The device name must match the name of the inserted module. Example: • If a module is inserted into a front-end module named "IODevice1", then it must be set to "IODevice1" here.
Device Note	Add comments to the device.
IP	There are two methods for configuring IP addresses: • User settings: The controller assigns IP addresses to the devices. • Fixed IP

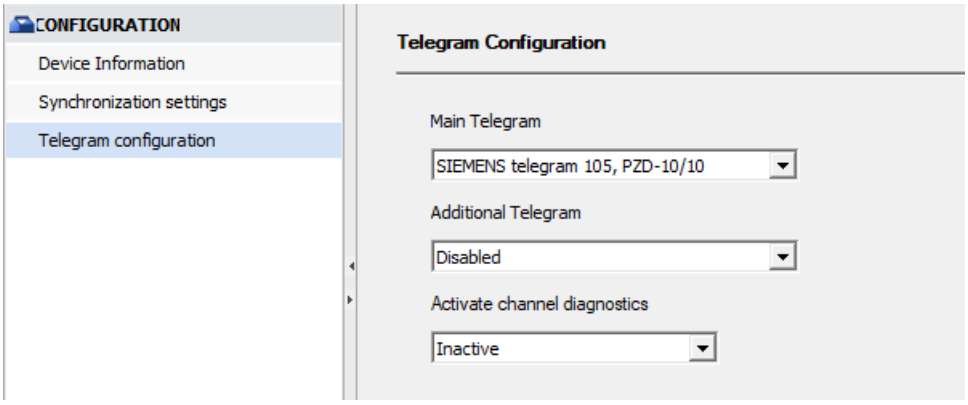
3. Click "Synchronization Settings" in the configuration window.



Isochronous

Update times	The "Send Clock" setting is configured in the "Communications" section of the controller.
Data hold	The data hold time can be set to "n times the update time". The value range is 3-255, with a default value of 3.
Ti/To value	The Ti/To value can be set in the following two ways: • Automatic minimization: Automatically calculate and use the minimum value of Ti/To. • Manually: Ti and To can only be edited when "manual" setting is selected.

4. In the configuration window, click "Telegram Settings" to select or modify the configuration telegram.



- Main Telegram: Telegram 105/5 is optional; the default telegram number is 105.
- **Additional Telegram:** Telegram 750 can be selected or disabled; it is disabled by default.
- Active diagnostics: If the device's GSD file contains the "Active diagnostics" parameter, the diagnostic function will be displayed; otherwise, it will not be displayed.

Communication times T_i, T_o, T_{DC}

When calculating the following error, the transmission times of the setpoint to the drive and of the actual position value to the controller are deducted. The transmission times result from the following communication times:

- T_i: Time to import process values
- T_o: Time to export process values
- T_{DC}: Send clock of the PROFINET interface

The following error is calculated from the delayed position setpoint by the communication times $T_i + T_o + T_{DC}$ and the motion control cycle T_{Servo} minus the actual position in the controller.

Telegram types

Telegram	Brief instructions
Standard telegrams	
5	• Control words STW1 and STW2, status words ZSW1 and ZSW2 • Speed setpoint 32 bit (NSET), actual speed value 32 bit (NACT) • Actual encoder value 1 (G1_XIST1, G1_XIST2) (motor encoder) • Dynamic Servo Control (DSC)[1)] – Velocity precontrol value – Position difference (XERR) – Kv factor gain position control (KPC)
Siemens telegrams (with torque limiting)	
105	• Control words STW1 and STW2, status words ZSW1 and ZSW2 • Speed setpoint 32 bit (NSET), actual speed value 32 bit (NACT) • Actual encoder value 1 (G1_XIST1, G1_XIST2) (motor encoder) • Dynamic Servo Control (DSC)[1)] – Velocity precontrol value – Position difference (XERR) – Kv factor gain position control (KPC) • Torque limiting

1) For use of Dynamic Servo Control (DSC), the motor encoder (first encoder in the telegram) of the drive must be used as the first encoder for the axis.

Configuring RT device

After inserting a device into the hardware network topology, you can edit the device's default properties in the topology view.

To change device properties and parameters in the topology view, follow these steps:

1. In the Topology View, select the device in the RT layer.
2. Click "Device Information" in the configuration window.

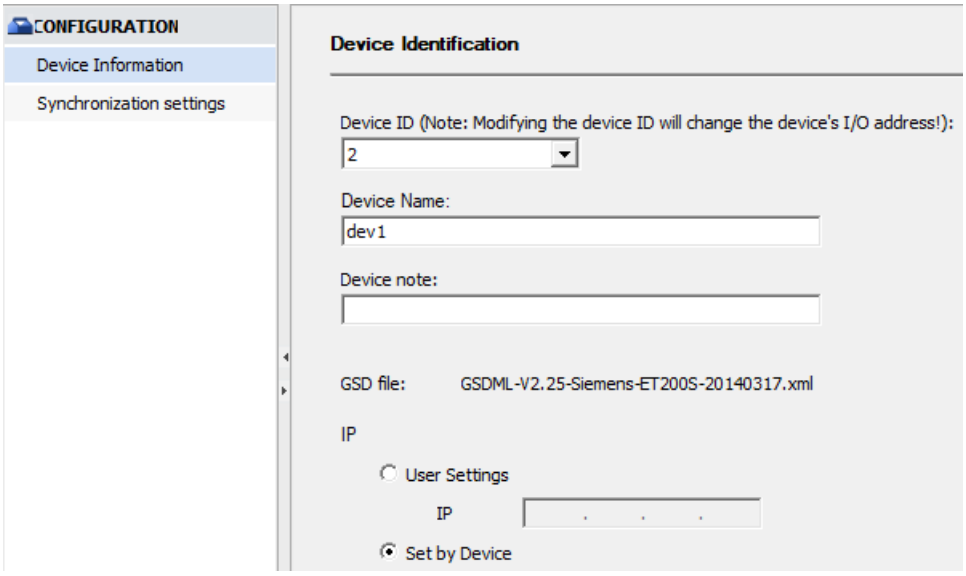
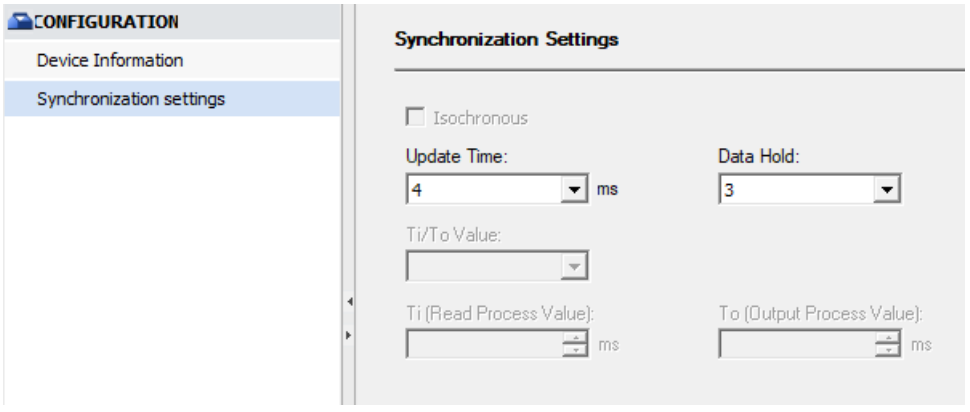


Table header	Meaning
device id	When the module connects to the PLC via PROFINET, the PROFINET device number is used.
Device name	The device name must match the name of the inserted module. Example: • If a module is inserted into a front-end module named "IODevice1", then it must be set to "IODevice1" here.
Device Notes	Add comments to the device.
IP Address	There are two methods for configuring IP addresses: • User settings: The controller assigns IP addresses to the devices. • Fixed IP

3. Click "Synchronization Settings" in the configuration window.



Update times: The data exchange cycle.
Data hold: The default value is 3.

Add/remove/move RT modules

Introduction

Once you have added devices from the hardware catalog to your configuration in Topology View, you can add modules to the devices. The graphic area and table area in the device view are linked. When you make changes in one region, the other region will automatically refresh to ensure that the data and configuration remain consistent.

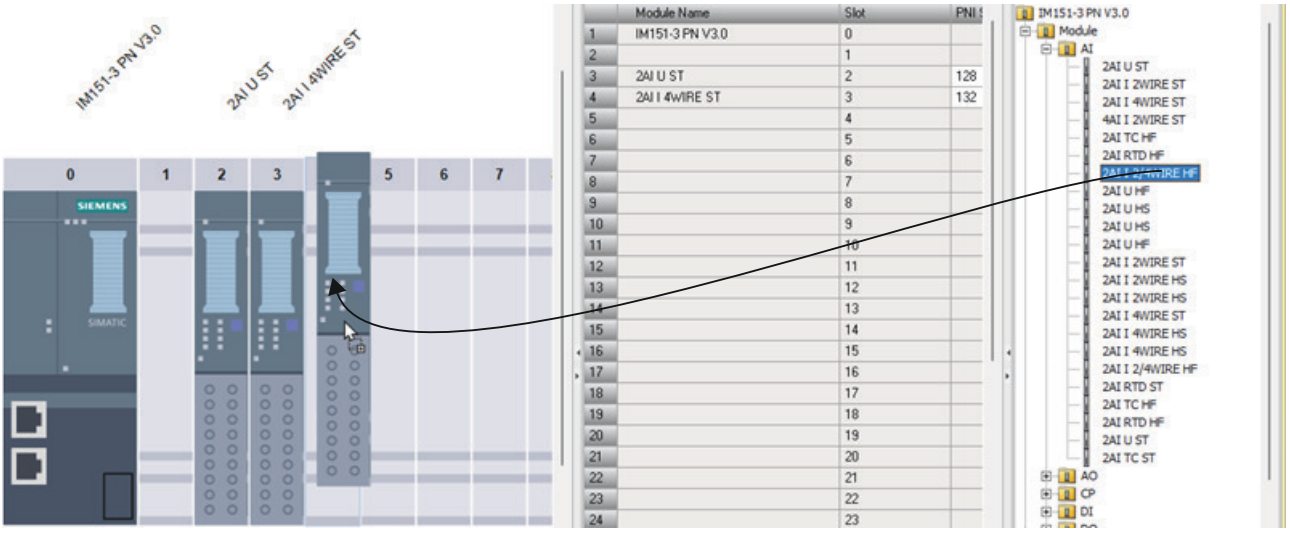
Note

No other modules may be dragged into or moved to SLOT-0.

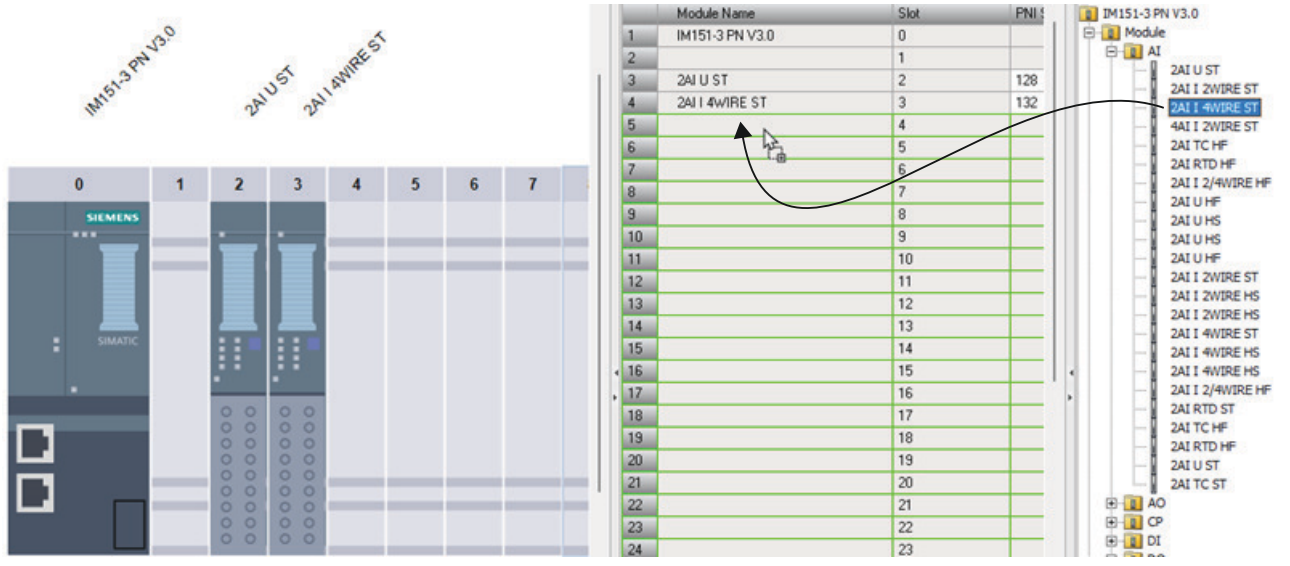
Adding a module

To add a module from the hardware catalog, follow these steps:

1. In the hardware catalog, select the required module.
2. If necessary, set the firmware version using the drop-down list in the hardware catalog.
3. Use drag-and-drop to move the module from the hardware catalog to an available valid slot in the graphic or table area:
   – Drag the module to an empty slot in the graphic area.



   – Drag the module to an empty slot in the table area.



After adding a module, STEP 7-Micro/WIN SMART V3 will add PROFINET device variables to the PROFINET device variable table.

Deleting modules

Delete the module using any of the following methods:

- Right-click the module in the graph or table area and select "Delete" from the context menu.
- Select the module in the graph or table area and press the Delete key.

Moving modules

To move a module, follow these steps:

1. Select the module you want to move.
2. Drag and drop the module to the location you want to move it to.

The selected module will be moved to the target location.

Configuring CPU as an I-Device without subordinates

Requirements

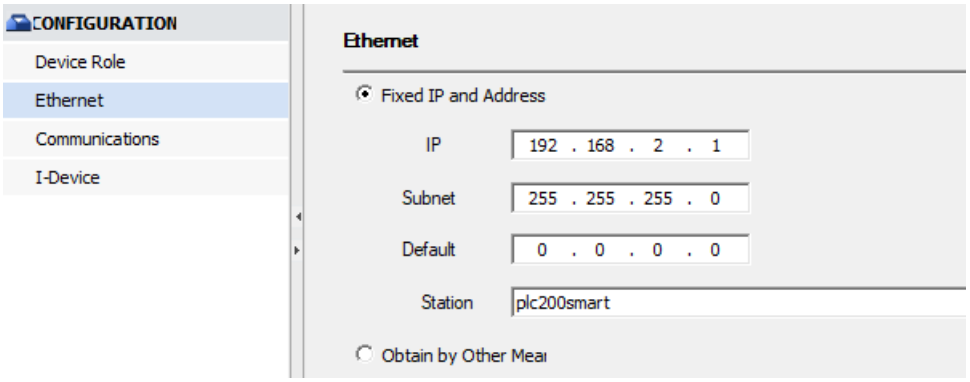
Ensure that the required CPU module is selected in the system block dialog.

Procedure

Please configure the I-Device following these steps:

1. Open the hardware network topology.
2. In the topology view, select PLC in the controller layer.
3. In the configuration window, check the checkbox before "I-Device".
   If you select "Parameter assignment of PROFINET interface by high-level IO controller", the device port configuration function will be available. Otherwise, the device port configuration function will be unavailable.

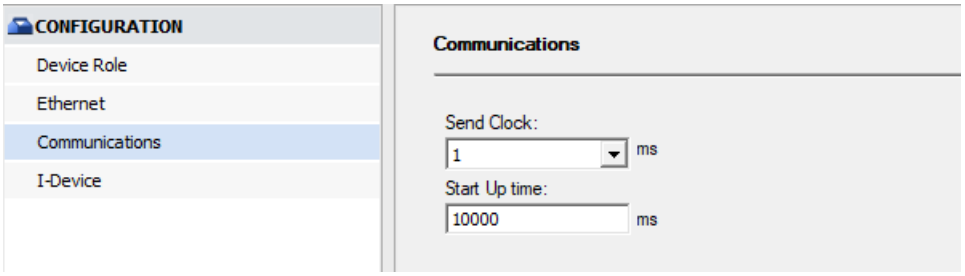
4. Configure the Ethernet and station name in the configuration window.



Ether-net

- Assign a fixed IP address: When configuring a device, the configured IP address and station name must be consistent with the actual IP address and station name of the I-Device.
- Obtaining an IP address through other means: When configuring the device, set the IP address to "User Settings" and fill in the specific IP address.

5. In the configuration window, click "Communication" to assign "Send Clock" and "Start Up time" for the controller.



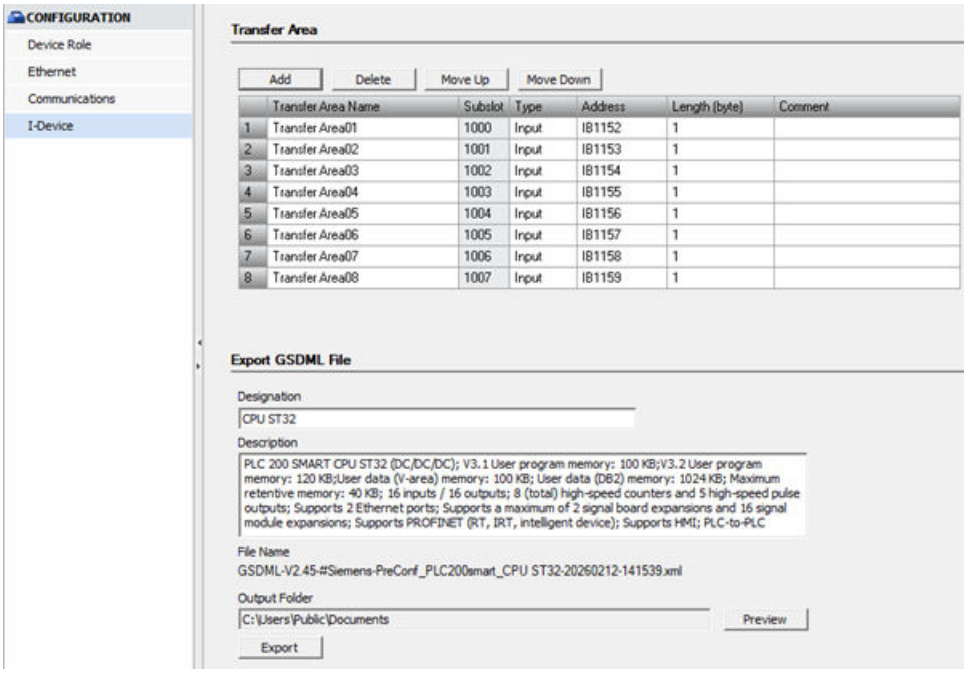
Your setting for the "Start Up time" impacts the amount of time the CPU takes to switch to RUN mode after it is powered on. The default value for "Start Up time" is 10 s, and the maximum value is one minute:

- 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" expires, the CPU switches to RUN mode at the end of the "Start Up time".

6. Click "I-Device" in the configuration window to access the configuration transfer area. 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 areas is 8.
- At least one transfer area must be configured.



The "Transfer Area" table lists the following items:

Project	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.
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:

Project	Description
Designation	The device family name is displayed by default. For example, CPU ST32. You can also enter any information as you need: • The Chinese, English and digit character are supported. • The characters '|', '\\', '/', ': ', '\', '', '<', '>' 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.

Configuring an I-device with a GSD file

If you use an I-device in another project, or if the I-device is used in another engineering system, then configure the higher-level IO controller and the I-device as described above.

Click on the "Export" button after configuring the transfer areas so a new GSD file is created from the I-device. This GSD file represents the configured I-device in other projects.

Import the GSD file into other projects.

Configuring an I-Device project with lower-level PROFINET IO system

Requirements

Ensure that the desired CPU module is selected in the "System Block" dialog box.

Procedure

To configure an I-Device project with lower-level PROFINET network, follow these steps:

1. Open the hardware network topology.
2. Add the devices you need (Page 512) within the hardware network topology.
3. Select the "I-Device" checkbox. "Controller" is selected by default.
   If you configure the I-Device with lower-level PROFINET IO system, the I-Device PROFINET interface parameters cannot be set by the higher-level controller.
4. Configure the CPU as an I-Device (Page 522) in the I-Device configuration page.
   In this step, you export one I-Device GSDML file.
5. Configure the CPU as a controller (Page 514) in the controller configuration page.
6. Download this I-Device project to a CPU.

Note

For an I-Device with lower-level PROFINET IO system, if there is an error with I-Device's lower-level IO device, this I-Device sends the following error to the higher-level controller: Device is running with error.

10.5 RS485

10.5.1 Communication selection

10.5.1.1 Overview

An RS485 network is a differential (multi-point) network and can have up to 126 addressable nodes per network and up to 32 devices per segment. Repeaters are used to segment the network. Repeaters are not addressable nodes; therefore, they are not included in the count of addressable nodes, but are counted in the devices per segment.

RS485 allows for data transfer at a high speed (from 100 m at 12 Mbit/s to 1 km at 187.5 Kbit/s).

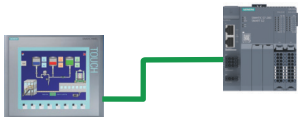
RS485 can operate with the PPI protocol and Freeport:

- PPI protocol: Can operate on RS485 or RS232 (half-duplex). Possible connections include:
  - PPI protocol devices
  - RS485 HMI displays
- Freeport: Can operate on RS485 or RS232 (half-duplex). Possible connections include:
  - RS485-compatible devices (for example, a bar code scanner)
  - Devices that have RS485 interfaces (for example, a control system)
  - Third-party devices using Freeport
  - Modems

10.5.1.2 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: 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.

PPI protocol and S7-200 SMART 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 and S7-200 SMART G2 CPU.

All S7-200 SMART V3 CPUs and S7-200 SMART G2 CPUs support both PPI and PPI Advanced protocols.

Table 10-2 Number of connections for the S7-200 SMART CPU

Module	Baud rate	Connection
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

10.5.2 Communication basics

10.5.2.1 Basics of S7-200 SMART network communications

The S7-200 SMART is designed to solve your communication and networking needs by supporting not only the simplest of networks but also supporting more complex networks. The S7-200 SMART also provides tools that allow you to communicate with other devices, such as printers and weigh scales which use their own communication protocols.

STEP 7-Micro/WIN SMART V3 makes setting up and configuring your network simple and straightforward.

Definition of baud rate and network address (Page 528)

Setting the baud rate and network address for the S7-200 SMART CPU (Page 529)

10.5.2.2 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 10-3 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 10-4 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

10.5.2.3 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 501), 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.

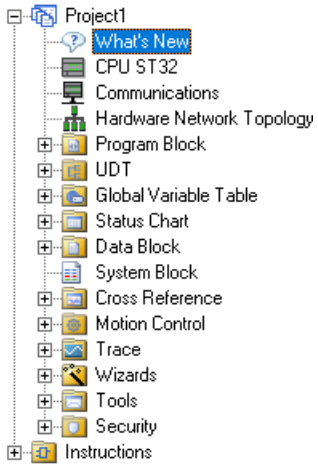
The system block of the CPU stores the RS485 port network address and baud rate. After you select the parameters for the CPU, you must download the project to the S7-200 SMART CPU.

Procedure

To access the "System Block" dialog, click one of the following:



System block button in the navigation bar



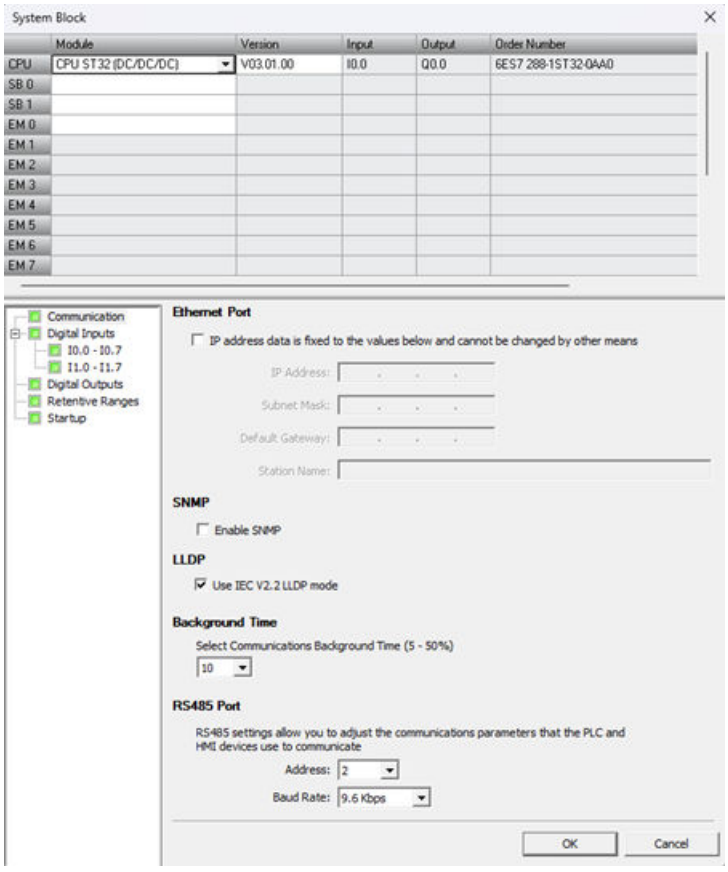
System block in the project tree



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.

10.5.3 Sample RS485 network configurations

10.5.3.1 Single-master PPI networks

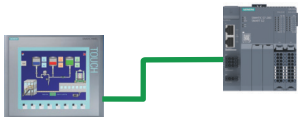
Introduction

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, TP or KP) is the network master:



HMI: Master S7-200 SMART: Slave

In the sample network, the CPU is a slave that responds to requests from the master.

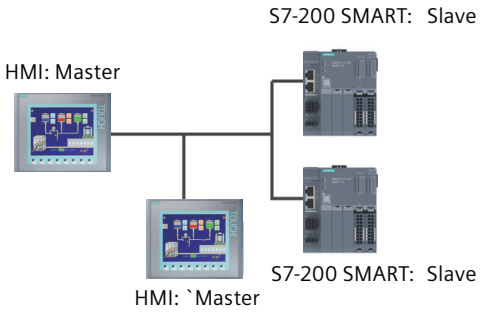
10.5.3.2 Multi-master and multi-slave PPI networks

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.



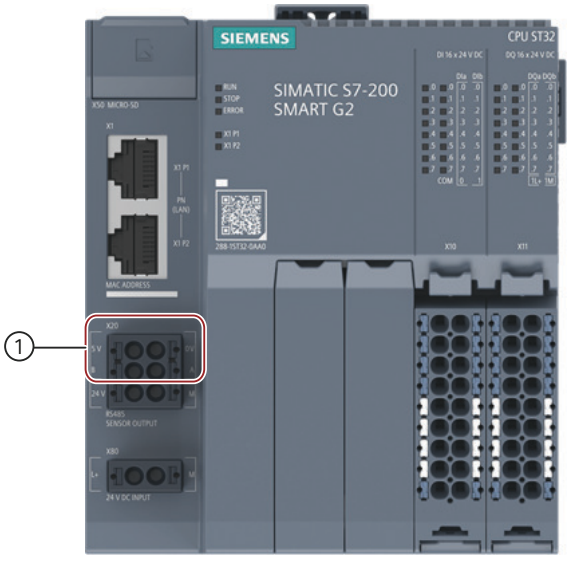
All devices (masters and slaves) have different network addresses. The S7-200 SMART CPUs are slaves.

10.5.4 Assigning RS485 addresses

10.5.4.1 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, the transmission rate is 19.2 Kbps).

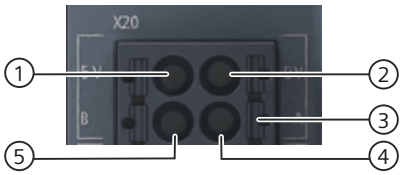


① RS485 port

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.

RS485(X20)

The following diagram shows the RS485 pin assignment and the location of the spring-loaded trip unit.

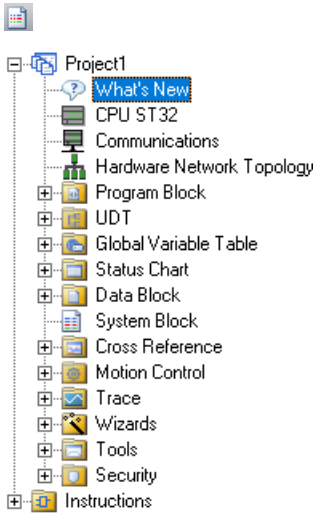


1. +5 V output with internal 100 Ω series resistor
2. 5 V return, logic common
3. Spring release
4. RS485 Signal A
5. RS485 Signal B

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.

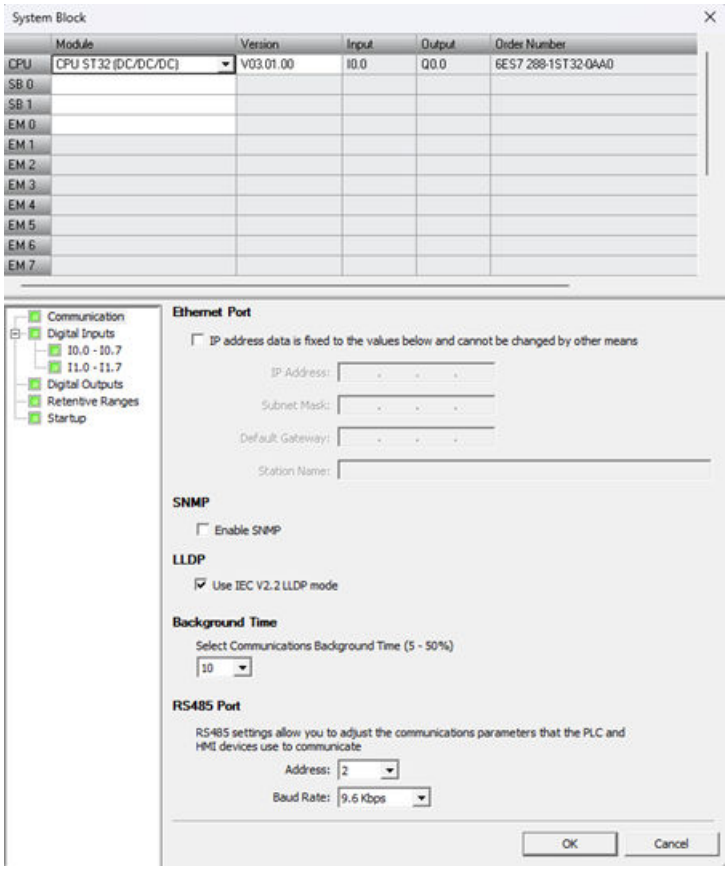
To access this dialog, perform one of the following:



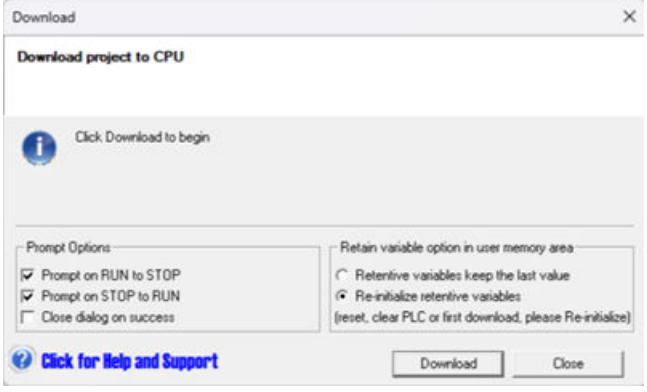
- In the navigation bar, click the "System Block" button.
- In the Project tree, select the "System Block" node, then press Enter; or double-click the "System Block" node.

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.

10.5.5 Building your network

10.5.5.1 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.

10.5.5.2 Building a network

Procedure

To build a network, proceed as follows:

1. Determining the distances, transmission rate, and cable for your network (Page 535)
2. Using repeaters on the network (Page 536)
3. Selecting the network cable (Page 537)
4. Connector pin assignments
5. Biasing and terminating the network cable
6. Creating user-defined protocols with Freeport mode (Page 539)
7. Using the RS232/PPI Multi-Master cable and Freeport mode with RS232 devices (Page 541)

10.5.5.3 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 10-5 Maximum length for a network cable

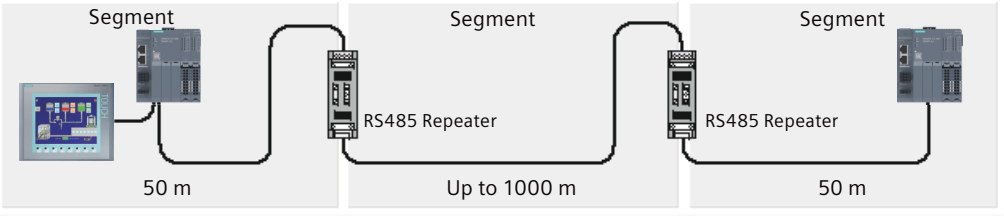
Baud rate	Non-isolated CPU port[1]	CPU port with repeater
9.6 Kbps to 187.5 Kbps	50 m	1,000 m
500 Kbps	Not supported	400 m
1 Mbps to 1.5 Mbps	Not supported	200 m
3 Mbps to 12 Mbps	Not supported	100 m
[1] The maximum distance allowed without using an isolator or repeater is 50 m. You measure this distance from the first node to the last node in the segment.		

10.5.5.4 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.



10.5.5.5 Specifications for RS485 cable

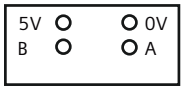
S7-200 SMART CPU networks use the RS485 standard on twisted pair cables. The following table lists the specifications for the network cable. You can connect up to 32 devices on a network segment.

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

10.5.5.6 connector pin assignments

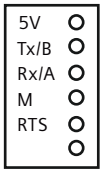
The following table shows the connector that provides the physical connection for the RS485 communication port on the S7-200 SMART G2 CPU and describes the communication port pin assignments.

Table 10-6 Pin assignments for the S7-200 SMART G2 CPU integrated RS485 port (Port 0)

Connector	Signals	Integrated RS485 port (Port 0)
X20 	5 V	+5 V output, 100 Ω series resistor
	0 V	Logic common
	B	RS485 Signal B
	A	RS485 Signal A

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.

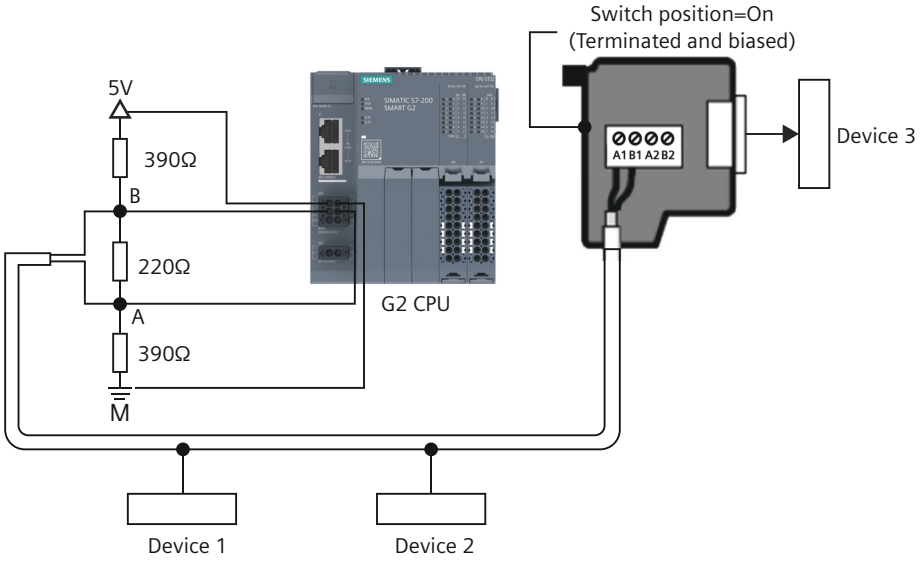
Table 10-7 Pin assignments for the S7-200 SMART G2 CM01 Signal Board (SB) port (Port 1)

Connector	Signals	CM01 Signal Board (SB) port (Port 1)
SB CM01 	+5 V DC	+5 V with internal 100 Ω series resistor
	Tx/B	RS232-Tx/RS485-B
	Rx/A	RS232-Rx/RS485-A
	M-ground	Logic common
	RTS (TTL)	Request to send

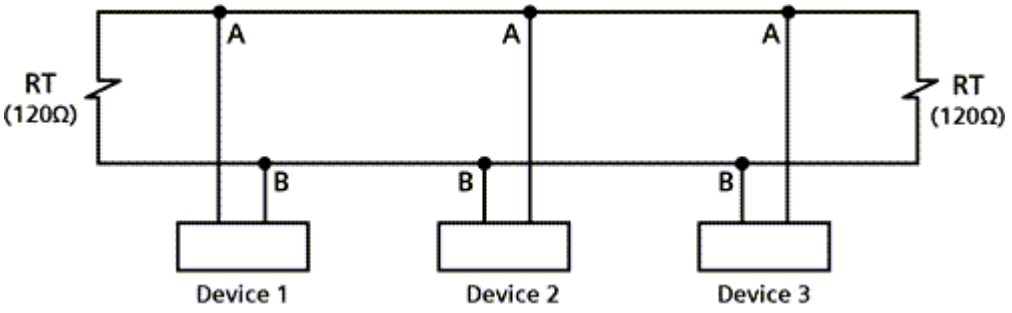
10.5.5.7 biasing and terminating the network cable

Siemens provides two types of network connectors that you can use to easily connect multiple devices to a network:

- Standard network connector



- To ensure the integrity and stability of the network signals, we strongly recommend that you install a 120 Ω terminating resistor at both ends of the cable when you do not use Siemens standard network connectors.



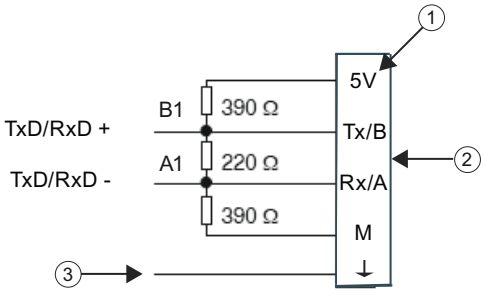
10.5.5.8 Biasing and terminating the CM01 signal board

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.

Note

The S7-200 SMART G2 CM01 does not require a connecting cable shield.



① Terminal name
② Terminal block
③ Cable shieding

10.5.6 Freeport mode

10.5.6.1 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

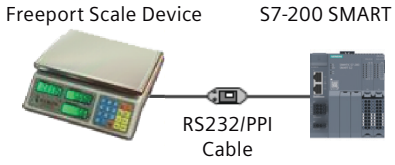
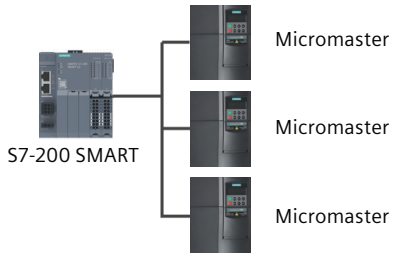
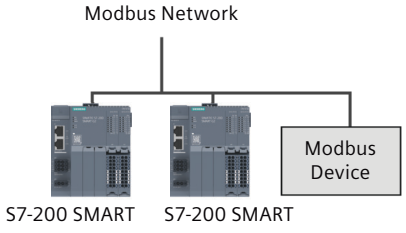
Attaching a USB-PPI cable to the CPU's RS485 port forces the CPU to exit Freeport mode and enable PPI mode. This allows STEP 7-Micro/WIN SMART to regain control of the CPU.

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.

Note

A direct connection to the USB-PPI cable is not supported by the S7-200 SMART G2 CPU. To connect an S7-200 SMART G2 CPU with this cable, you will need to make your own RS485 connection wire.

Table 10-8 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.

10.5.6.2 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 10-9 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.

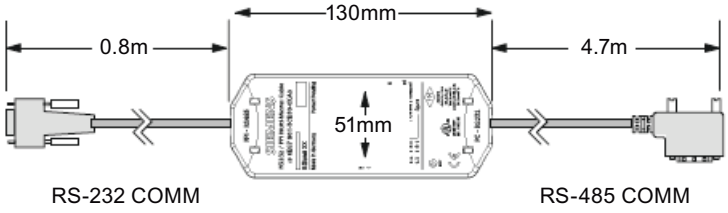
10.5.6.3 Switches of the RS232/PPI Multi-Master cable

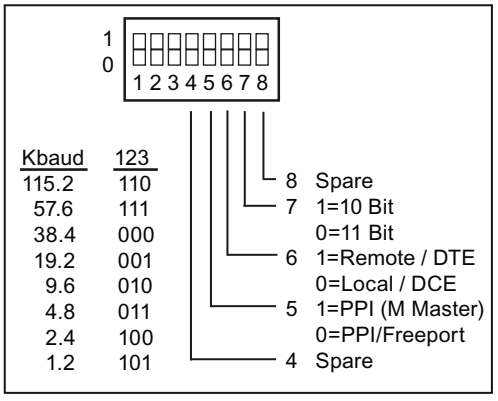
Introduction

You can use a PC/PPI cable to connect the RS232 communication port of a modem to an S7-200 SMART CPU RS485 port.

The CM01 signal board supports both RS232 half-duplex and RS485. Using the CM01 signal board, you can connect a modem directly to the RS232 port and without using a PC/PPI cable.

Switches 1, 2, and 3 on the PC/PPI cable set the baud rate. Switch 6 selects either the Data Communications Equipment (DCE) or Data Terminal Equipment (DTE) mode. Switch 5 selects the operation of the RTS signal on the RS232 port of the PC/PPI cable.





LED	Color	Description
Tx	Green	RS-232 transmit indicator
Rx	Green	RS-232 receive indicator
PPI	Green	RS-485 transmit indicator

Modems normally use the RS232 control signals (such as RTS, CTS, and DTR) to allow a computer to control the modem. When you use a modem with a PC/PPI cable, you must configure the modem to operate without these signals. To determine the commands required for configuring the modem, consult the documentation for your specific modem.

Switch 7 (only for V2.x CRs)

You use switch 7 on the S7-200 SMART Classic CPUs to select either 10- or 11-bit modem protocols. The RS232-PPI cable marking reflects this use.

You use switch 7 for a different purpose on the S7-200 SMART CPUs. Switch 7 selects the polarity of pin 9 on the RS485 D-connector. This switch has no effect on the SR or ST model CPUs. Switch 7 should be set to a "1" (labeled "10 Bit") when used with any of the S7-200 SMART CRs model CPUs to allow Freeport mode.

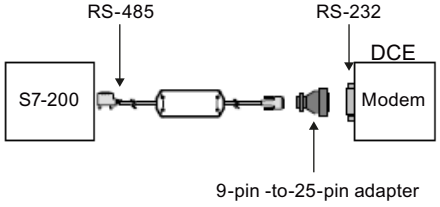
Switch 6

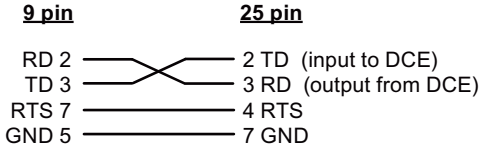
Switch 6 of the PC/PPI cable allows you to set the RS232 port of the cable to either DCE or DTE mode. If you are using the PC/PPI cable with STEP 7-Micro/WIN SMART or if the PC/PPI cable is connected to a computer, set the PC/PPI cable to DCE mode. If you are using the PC/PPI cable with a modem (which is a DCE device), set the PC/PPI cable to DTE mode. This eliminates the need to install a null modem adapter between the PC/PPI cable and the modem. Depending on the connector on the modem, you may still need to use a 9-pin-to-25-pin adapter.

Switch 5

Switch 5 of the PC/PPI cable selects the operation of the RTS signal on the RS232 connector. Selecting "RTS for XMT" will have the following effect: Causes the RTS signal to be active when the CPU is transmitting on the RS485 port, and the RTS signal to be inactive when the CPU is receiving data. Selecting "RTS Always" will have the following effect: Causes the RTS signal to always be active on the RS232 port of the PC/PPI cable, regardless of whether the CPU is transmitting or receiving. Switch 5 affects only the RTS signal when the PC/PPI cable is set for DTE mode.

The following figure shows the pin assignments for common modem adapters.





The following table shows the pin numbers and functions for the RS485 and RS232 ports of the PC/PPI cable in DTE mode. Table 7-10 shows the pin numbers and functions for the RS485 and RS232 ports of the PC/PPI cable in DCE mode. The PC/PPI cable supplies RTS only when it is in DTE mode.

RS-485 connector pin-out		**RS-232 DTE connector pin-out ****	
Pin	Signal Description	Pin	Signal Description
1	Ground (RS485 logic ground)	1	Not used: Data Carrier Detect (DCD)
2	24 V Return (RS485 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	Not used: Data Terminal Ready (DTR)
5	Ground (RS485 logic ground)	5	Ground (RS232 logic ground)
6	NC (No connect)	6	Not used: Data Set Ready (DSR)
7	24 V Supply	7	Request To Send (RTS) (output - switch selectable)

RS-485 connector pin-out		RS-232 DTE connector pin-out **	
8	Signal A (RxD/TxD-)	8	Not used: Clear To Send (CTS)
9	Protocol select	9	Not used: Ring Indicator (RI)

**A conversion from female to male, and a conversion from 9-pin to 25-pin may be required for modems

RS-485 connector pin-out		RS-232 DCE connector pin-out	
Pin	Signal Description	Pin	Signal Description
1	Ground (RS485 logic ground)	1	Not used: Data Carrier Detect (DCD)
2	24 V Return (RS485 logic ground)	2	Receive Data (RD) (output to PC/PPI cable)
3	Signal B (RxD/TxD+)	3	Transmit Data (TD) (input from PC/PPI cable)
4	RTS (TTL level)	4	Not used: Data Terminal Ready (DTR)
5	Ground (RS485 logic ground)	5	Ground (RS232 logic ground)
6	NC (No connect)	6	Not used: Data Set Ready (DSR)
7	24 V Supply	7	Not used: Request To Send (RTS)
8	Signal A (RxD/TxD-)	8	Not used: Clear To Send (CTS)
9	Protocol select	9	Not used: Ring Indicator (RI)

10.6 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

11

11.1 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 552), Modbus TCP (Page 572), Open User Communication (Page 589),PN Read Write Record library (Page 632), SINAMICS Library (Page 651) and USS protocol (Page 635))
- User-defined (Page 707) (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. 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. 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. 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. 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. 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 (https://www.siemens.com/global/en/products/automation/topic-areas/industrial-cybersecurity.html)

11.2 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.

11.2.1 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 11-1 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 CPU memory ranges and features (Page 1245).

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 565) instruction for more information on setting up the memory restrictions for the Modbus RTU slave.

See the description of the MBUS SERVER (Page 578) instruction for more information on setting up the memory restrictions for the Modbus TCP server.

11.2.2 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 11-2 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.

11.3 Modbus RTU library instructions

11.3.1 Modbus RTU communication overview

11.3.1.1 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.

11.3.1.2 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 557) 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 565) 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).

11.3.1.3 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.

11.3.2 Modbus RTU master

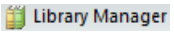
11.3.2.1 Using the Modbus RTU master instructions

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 557) and MBUS_MSG (Page 559). For a second Modbus RTU Master, use the instructions MB_CTRL2 (Page 557) and MB_MSG2 (Page 559). For a third Modbus RTU Master, use the instructions MB_CTRL3 (Page 557) and MB_MSG3 (Page 559).

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:

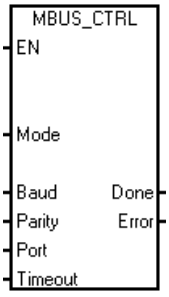
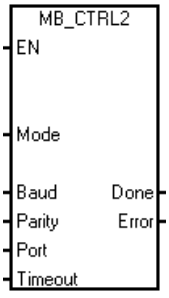
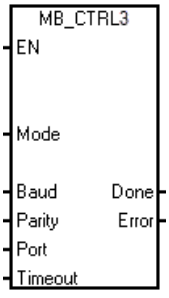
1. Insert the MBUS_CTRL / MB_CTRL2 / MB_CTRL3 instruction in your program and execute it on every scan. You can use the MBUS_CTRL / MB_CTRL2 / MB_CTRL3 instruction either to initiate or to change the Modbus communications parameters. When you insert the MBUS_CTRL / MB_CTRL2 / MB_CTRL3 instruction, STEP 7-Micro/WIN SMART adds several protected subroutines and interrupt routines to your program.
2. Click the "Manager" button  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.

11.3.2.2 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 11-3 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. **NOTICE**: Upon successful execution of the initialization instruction, the configuration states of the relevant parameters (including Baud, Parity, and Port) will be latched. That is, modifications to the aforementioned parameters will not take effect after the PLC enters RUN mode.

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 11-4 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.

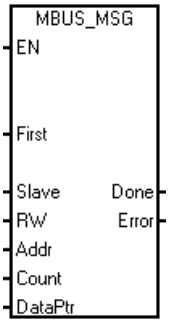
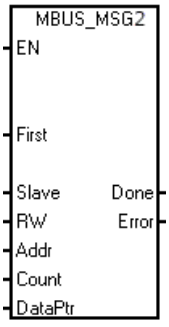
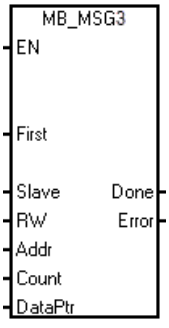
The Error output contains the result of executing the instruction.

See also Modbus RTU master execution error codes (Page 562)

11.3.2.3 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 11-5 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 11-6 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 568) 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.

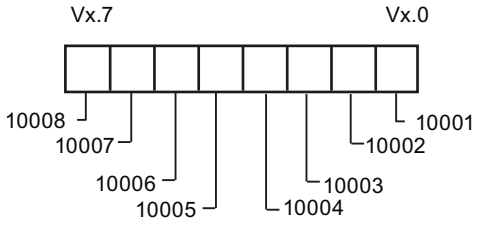
Memory layout

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 11-7 Modbus Holding Register

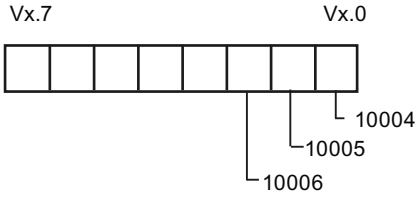
CPU memory byte address		CPU memory word address		Modbus holding register address	
Address	**Hex data**	**Address**	**Hex data**	**Address**	**Hex data**
VB200	12	VW200	12 34	40001	12 34
VB201	34				
VB202	56	VW202	56 78	40002	56 78
VB203	78				
VB204	9A	VW204	9A BC	40003	9A BC
VB205	BC				

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 562) is valid only when the Done output is TRUE.

11.3.2.4 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.

11.3.3 Modbus RTU slave

11.3.3.1 Using the Modbus RTU slave instructions

Step

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 "Manager" button Library Manager 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.

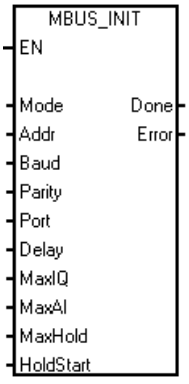
The Modbus slave instructions support the Modbus RTU protocol. These instructions use the Freeport feature of the S7-200 SMART CPU to support the most common Modbus functions. The following Modbus functions are supported:

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.

11.3.3.2 MBUS_INIT instruction (initialize slave)

Table 11-8 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 11-9 MBUS_INIT parameters

Input/output	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.

Note

To read addresses beyond I 1151.7, or to read/write addresses beyond Q 1151.7, you must first map these addresses to the V area and perform read/write operations via holding registers.

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 568) 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.

11.3.3.3 MBUS_SLAVE instruction

Table 11-10 MBUS_SLAVE instruction

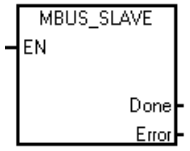
LAD / FBD	STL	Description
	**CALL MBUS_SLAVE, Done, Error**	The MBUS_SLAVE instruction is used to service a request from the Modbus master and must be executed every scan to allow it to check for and respond to Modbus requests. The instruction is executed on each scan when the EN input is ON. The MBUS_SLAVE instruction has no input parameters.

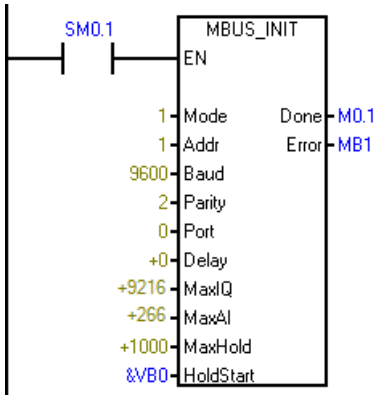
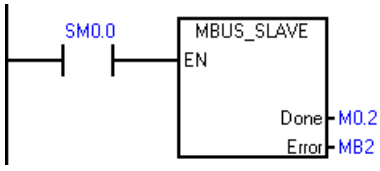
Table 11-11 Parameters for the MBUS_SLAVE Instruction

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 (Page 568) 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 11-12 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**

11.3.3.4 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

11.3.4 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.

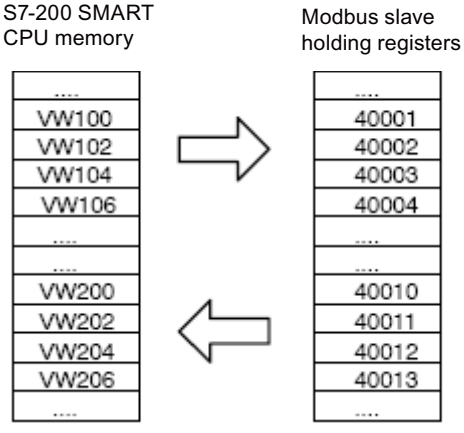
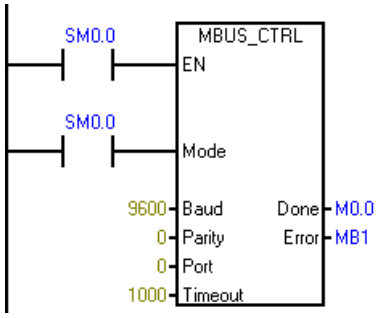
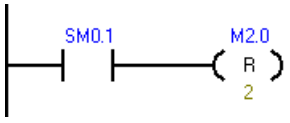
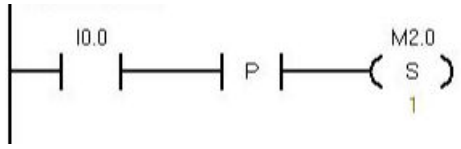
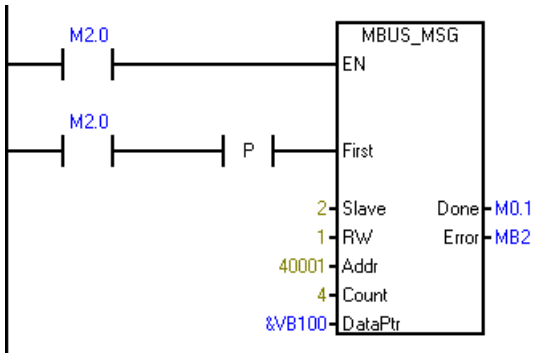
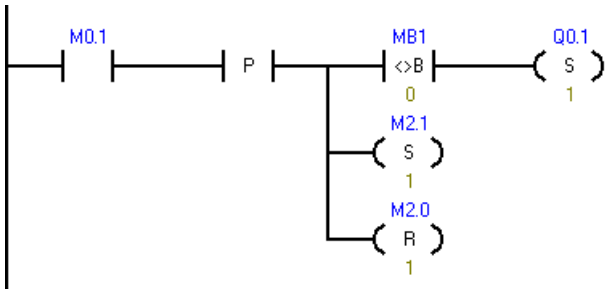
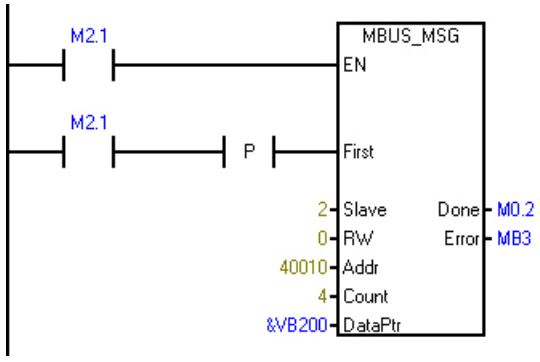
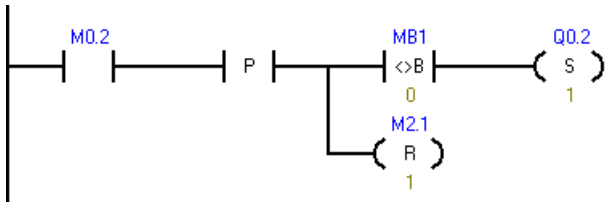


Figure 11-1 Example Program Data Transfers

The following program turns on outputs Q0.1 and Q0.2 if the MBUS_MSG instruction returns an error.

Table 11-13 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.
	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.

LAD	Description
	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.

11.3.5 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.

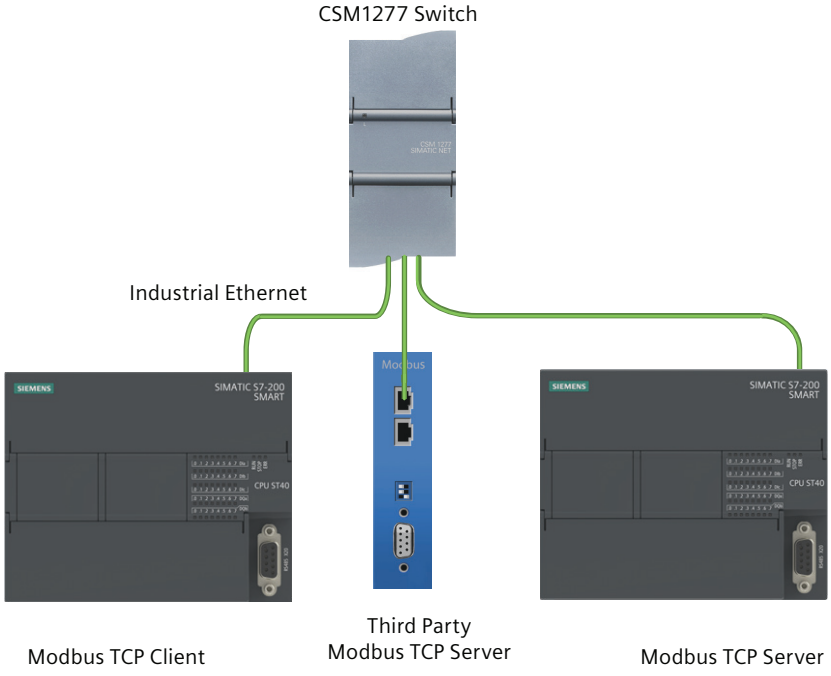
For example: to access holding register 16768, the **Addr** parameter of MBUS_MSG/MB_MSG2/MB_MSG3 should be set to 416768.

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.

11.4 Modbus TCP library

11.4.1 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 574)
- MBUS_SERVER (Page 578)

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

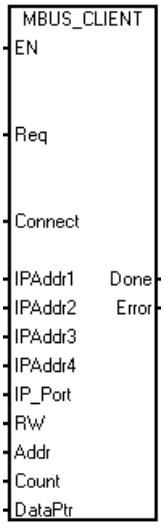
The Modbus server instruction (MBUS_SERVER) uses the following resources from the CPU:

- 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.

11.4.2 Modbus TCP client

11.4.2.1 MBUS_CLIENT instruction

Table 11-14 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 11-15 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. 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.

11.4.2.2 Modbus TCP client execution error codes

The MBUS_CLIENT instruction (Page 574) can return the following error codes:

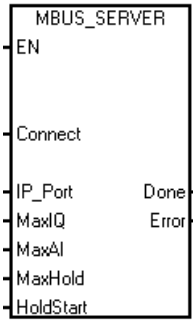
Error (decimal)	Description
0	No error
32	Unknown state Check network connections and check that the program is not modifying any library symbols that intefere with client/server communication.
33	Connection is busy with another request. A single connection can only service one Modbus request at a time.
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 587) and Open User Communication error codes (Page 612)

11.4.3 Modbus TCP server

11.4.3.1 MBUS_SERVER instruction

Table 11-16 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 11-17 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 value: 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. **Note**: To read addresses beyond I 1151.7, or to read/write addresses beyond Q 1151.7, you must first map these addresses to the V area and perform read/write operations via holding registers.

Parameter and type		Data type	Description
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.
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

Disconnect

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.

11.4.3.2 Modbus TCP server execution error codes

The MBUS_SERVER instruction (Page 578) 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

Error (decimal)	Description
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 587) and Open User Communication error codes (Page 612)

11.4.4 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.

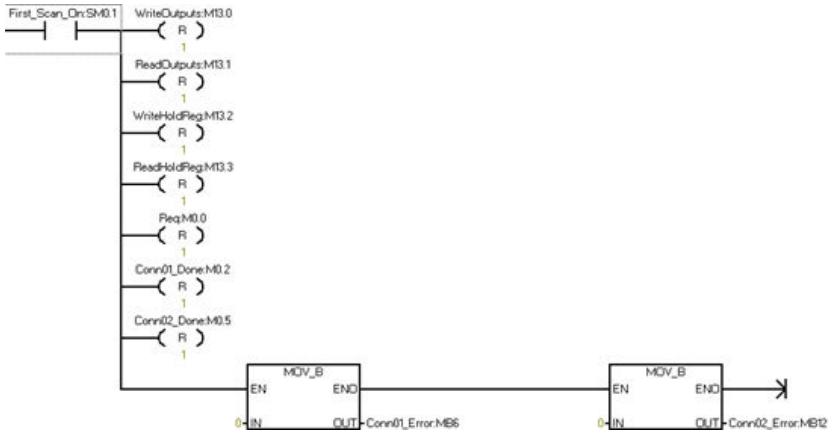
The basic description for this example:

Two Modbus clients establish connections with two Modbus server devices.

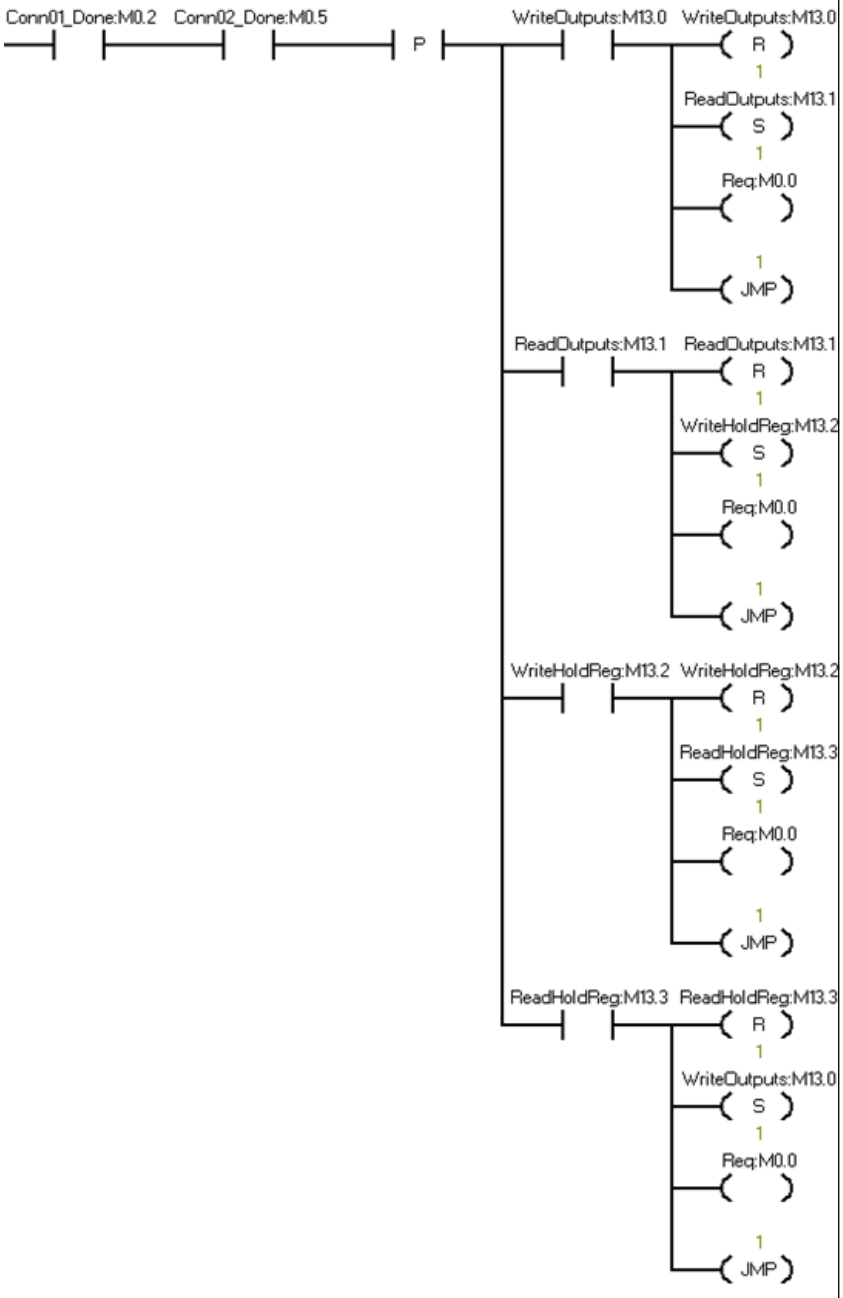
Modbus server 01: IP Address 192.168.2.10, Port 502

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	Description
	Network 1: On startup, clear all the flags and errors.

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

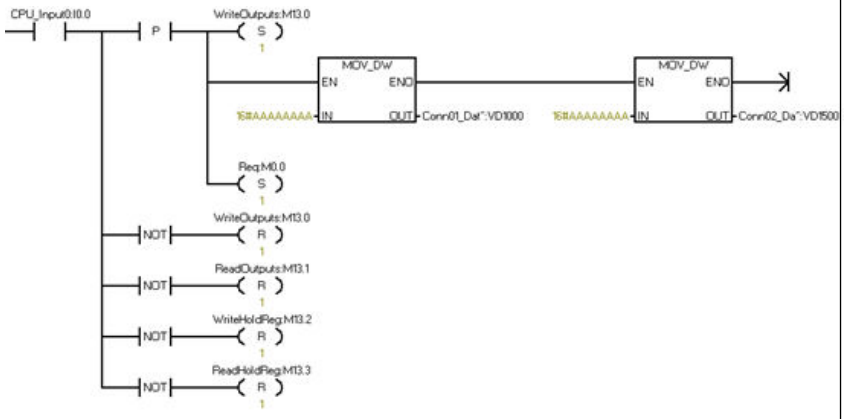
LAD	Description
	Network 2: When both clients complete the Modbus requests, start the next Modbus request.

LAD	Description

Symbol	Address	Comment
Conn01_Done	M0.2	Connection #1 has completed the current operation
Conn02_Done	M0.5	Connection #2 has completed the current operation
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



Network 3



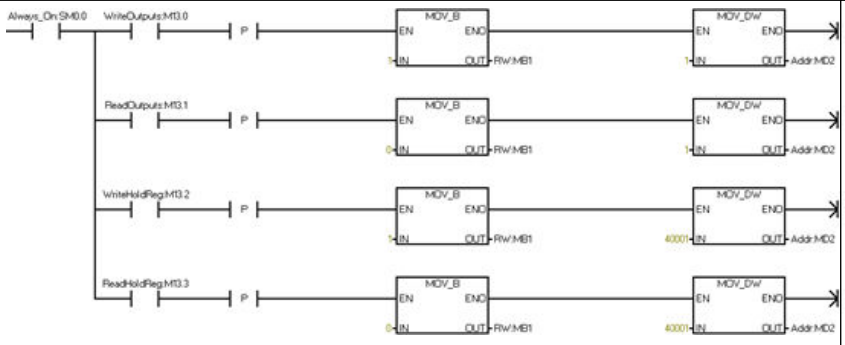
Network 4:

CPU_Input 0 rising edge triggers the Modbus request sequence to begin Write some data in V memory to send to the Modbus servers.

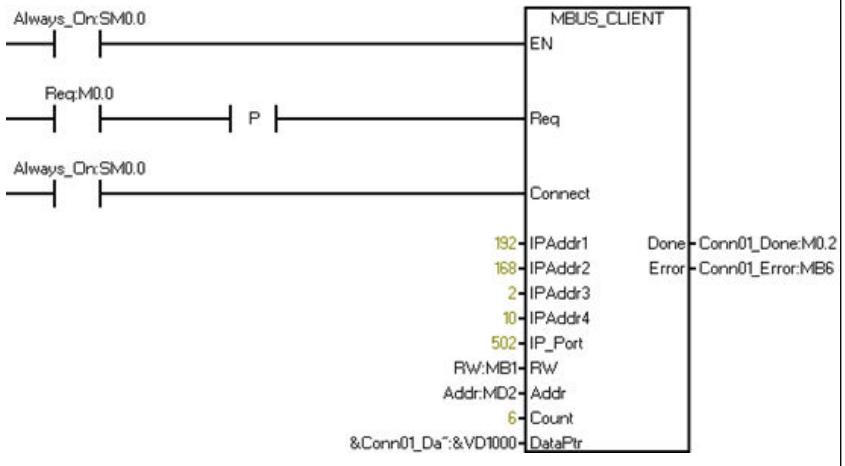
Set the Req input bit TRUE.

When the CPU_Input 0 is False, stop sending Modbus requests.

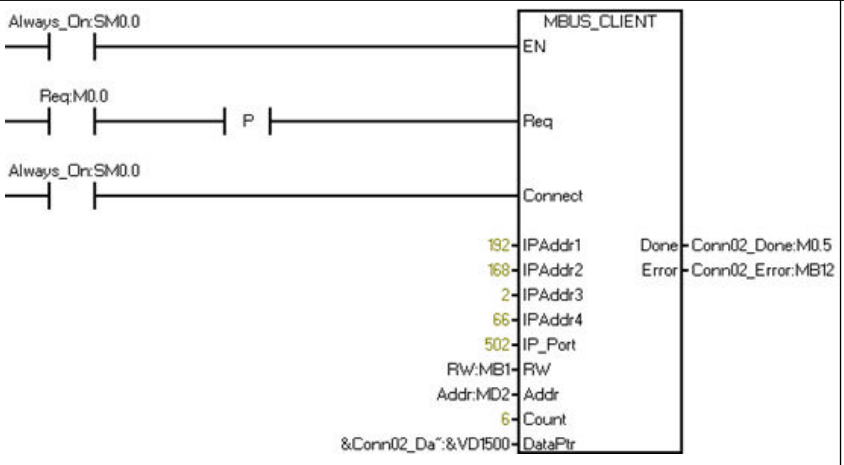
Symbol	Address	Comment
Conn01_Data	VD1000	Connection #1 Data Location
Conn02_Data	VD1500	Connection #2 Data Location
CPU_Input0	I0.0	
ReadHoldReg	M13.3	Initiate a Read Holding Register request
ReadOutputs	M13.1	Initiate a Read Output request
Req	M0.0	Initiate a Modbus request
WriteHoldReg	M13.2	Initiate a Write Holding Register request
WriteOutputs	M13.0	Initiate a Write Output request

LAD	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
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

LAD	Description
	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.

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

LAD	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.

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

11.4.5 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.

11.4.6 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)

11.4.7 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

11.5 Open user communication library

11.5.1 Overview

The STEP 7-Micro/WIN SMART Open User Communication (OUC) library instructions create the tables required by the OUC instructions (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.

11.5.2 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-".

11.5.3 Open user communication library instructions

11.5.3.1 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.

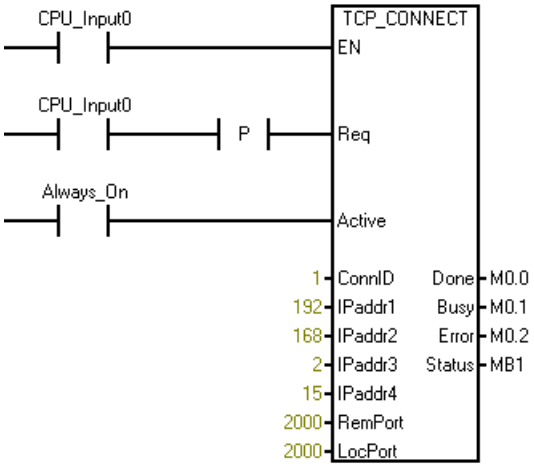
Table 11-18 Parameters of the TCP_CONNECT instruction

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 590).
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 612) 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.

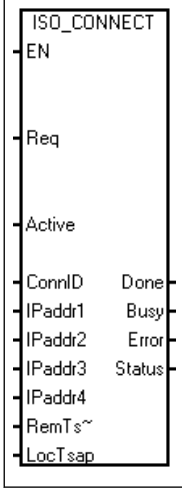
Example

This is an example usage of the TCP_CONNECT instruction:



11.5.3.2 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.

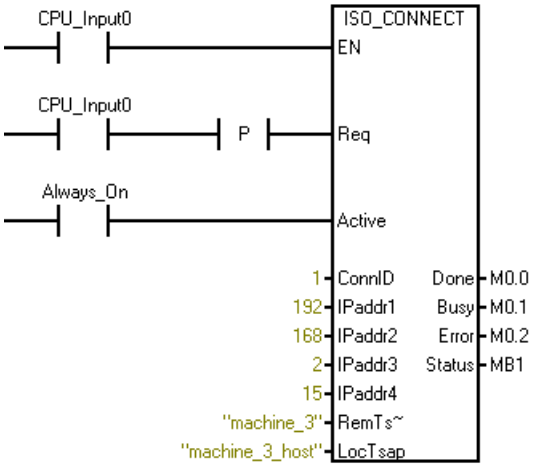
Table 11-19 Parameters of the ISO_CONNECT instruction

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 612) 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:

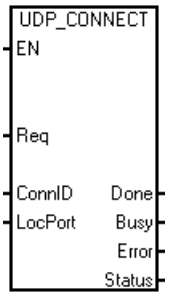


STEP7-Micro/WIN SMART always uses pointers to pass a string to the ISO_CONNECT instruction for the RemTsap and the LocTsap. If you use a constant string (as in the example above), STEP7-Micro/WIN SMART automatically creates the string and the pointer. If you wish to create strings in a Data Block and 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.

11.5.3.3 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:

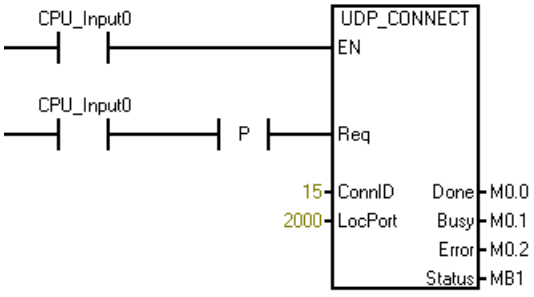
- The instruction sets the Done output if the connection is active and ready to send or receive. This only means that you can use the connection. This is not an indication that there is any remote device present.
- The instruction sets the Busy output if the connect is still processing.
- The instruction 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 11-20 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 590).
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 612) 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.

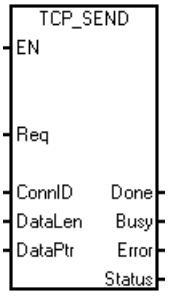
Example

This is an example usage of the UDP_CONNECT instruction:



11.5.3.4 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.

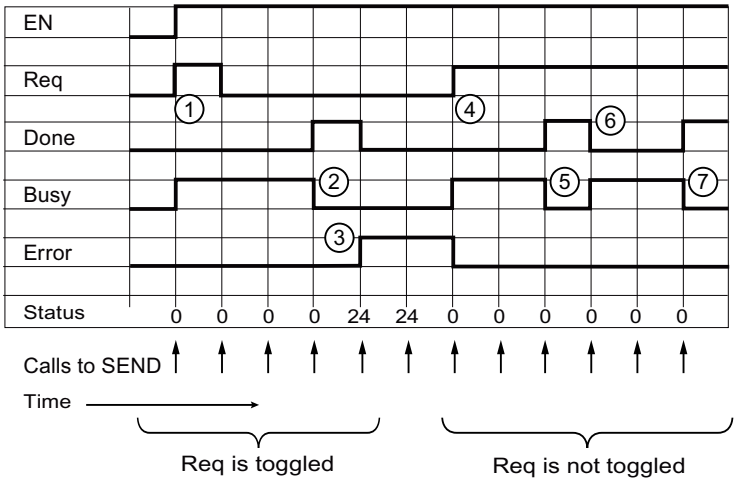
LAD/FBD	STL	Description
	**TCP_SEND Req, ConnID, DataLen, DataPtr, Done, Busy, Error, Status**	The TCP_SEND transmits the requested number of bytes from the requested buffer location over an existing connection.

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.

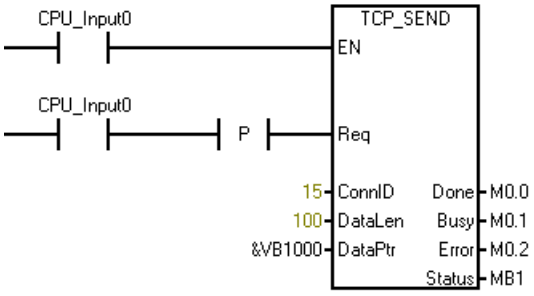
Table 11-21 Parameters of the TCP_SEND instruction

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 612) 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.

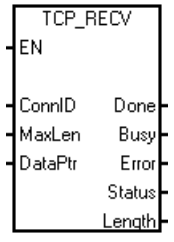
Example

This is an example usage of the TCP_SEND instruction:



11.5.3.5 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

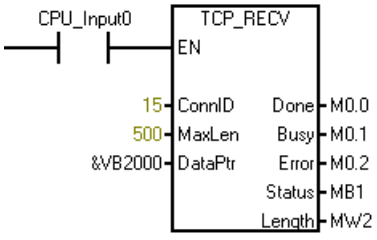
happens because ISO-on-TCP has start and end markers in the protocol to delineate the messages and separate them in the receiving device.

Table 11-22 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 612) 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.

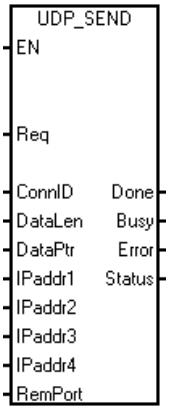
Example

This is an example usage of the TCP_RECV instruction:



11.5.3.6 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.

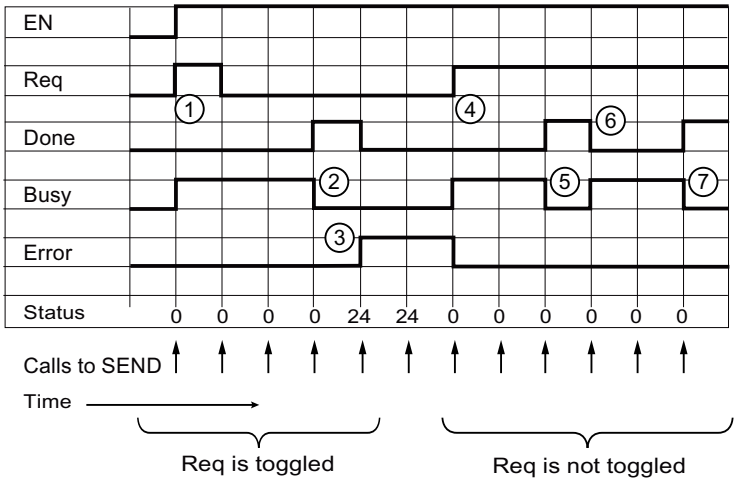
LAD/FBD	STL	Description
	**UDP_SEND Req, ConnID, DataLen, DataPtr, IPaddr1, IPaddr2, IPaddr3, IPaddr4, RemPort, Done, Busy, Error, Status**	The UDP_SEND instruction transmits the requested number of bytes from the requested buffer location to the device specified by the IP address and port.

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 590) 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.

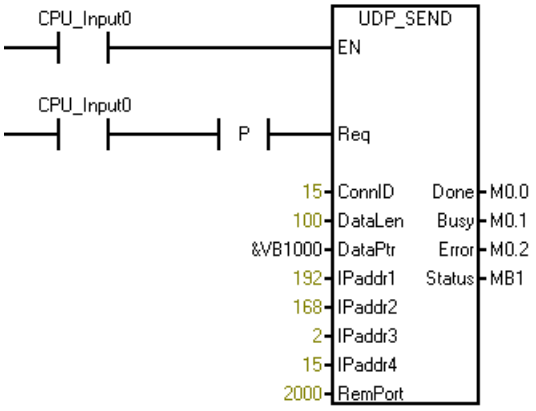
Table 11-23 Parameters of the UDP_SEND instruction

Parameter	Declara-tion	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 612) 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.

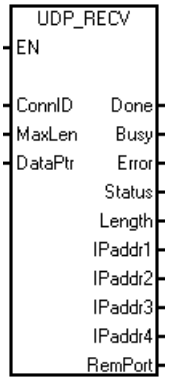
Example

This is an example usage of the UDP_SEND instruction:



11.5.3.7 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.

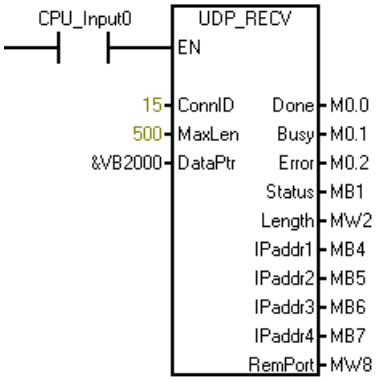
The UDP_RECV instruction also returns the IP address and port number of the remote device. This allows your program to respond to the remote device with a UDP_SEND where the remote IP address and port are required parameters.

Table 11-24 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 612) 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.

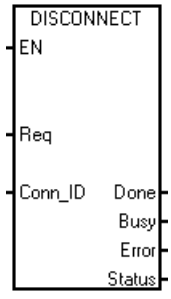
Example

This is an example usage of the UDP_RECV instruction:



11.5.3.8 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.

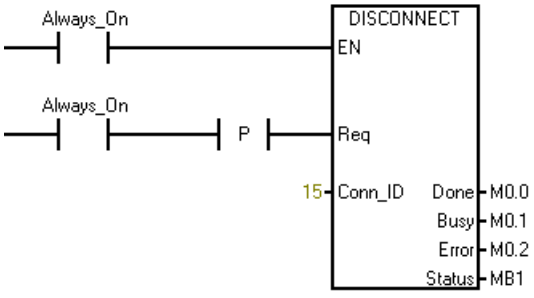
After the DISCONNECT instruction completes a disconnect, if the program calls the DISCONNECT instruction with Req set FALSE, the instruction returns error code 24, which means no operation pending.

Table 11-25 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 612) 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:



11.5.4 Open user communication library instruction error codes

The following table lists the Open User Communication (OUC) library (Page 589) 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

11.5.5 Open user communication library example

11.5.5.1 Active partner (client)

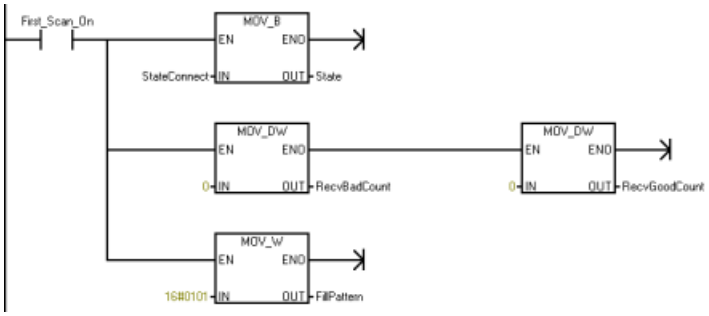
This program implements a simple state machine to manage the creating of a connection, cyclically sending/receiving a message, and handling errors.

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 623) to see the symbol table for this program.

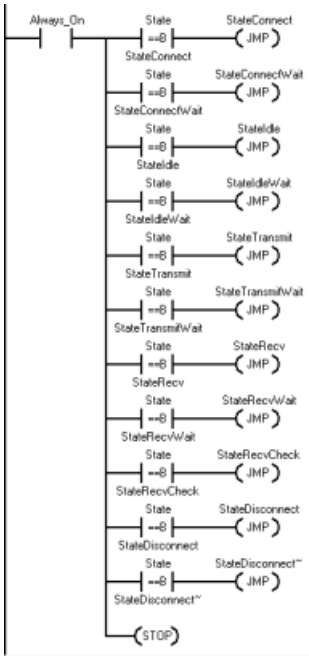
Network 1: On the first scan....

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.



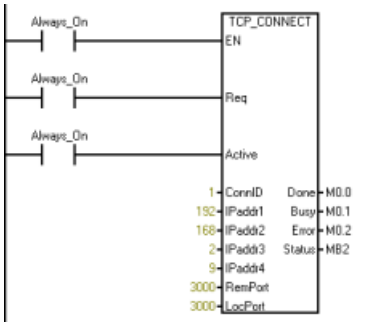
Network 3: State "Connect"...

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.

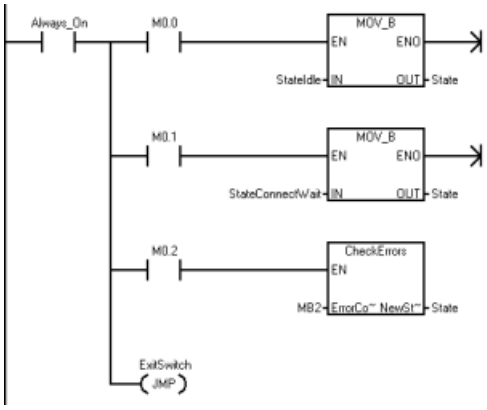


Network 5: If Done is TRUE, the CPU established the connection to go to the "Idle" state.

If Busy is TRUE, go to the "Connect Wait" state to wait for the connection to be established.

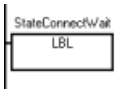
If Error is TRUE, there is probably an error with the input parameters, so check to see what state to go to next.

In all cases, exit the state machine for this scan. The program continues to the next state on the next scan.

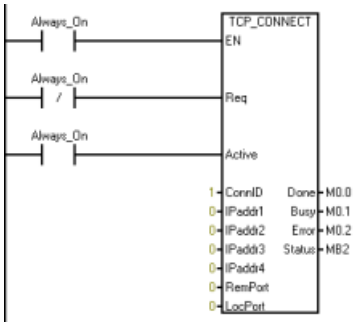


Network 6: State "Connect Wait"...

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.

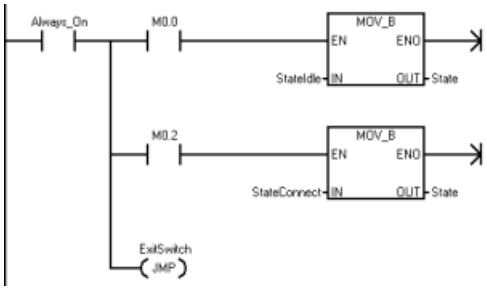


Network 8: If Done is TRUE, this means that the CPU established the connection, so continue in the "Idle" state.

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.

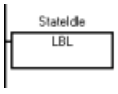
If Error is TRUE, there is a problem connecting to the passive device. In this case, just try again by going back to the "Connect" state. Note that if the passive device is present but it rejects the connection request, the connection errors very quickly and utilizes a great amount of bandwidth as the 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.

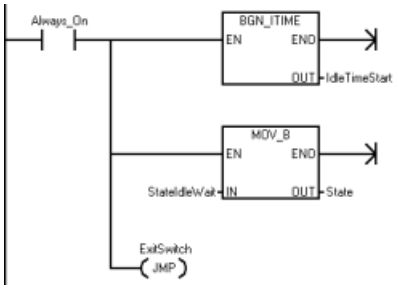


Network 9: State "Idle"...

This state puts a time delay between messages so that you do not flood the network. The symbol "IdleTimeDelay" specifies the delay time.

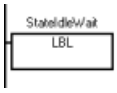


Network 10: Capture the interval timer and then change to the "Idle Wait" state to delay until it is time to transmit again.

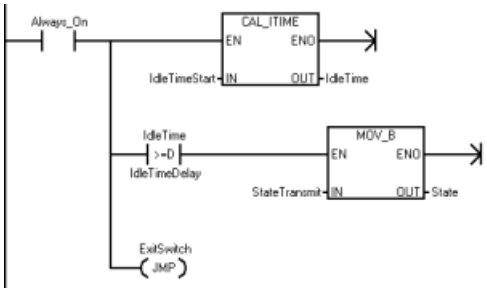


Network 11: State "Idle Wait"...

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.

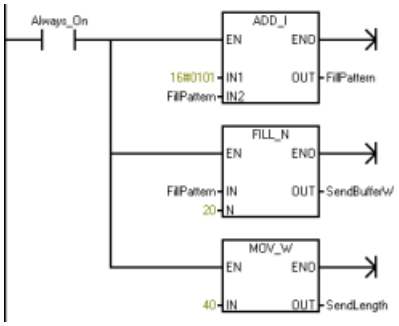


Network 13: State 'Transmit"...

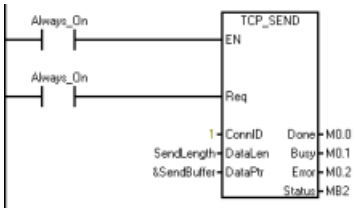


Network 14: Create the message to send to the passive device.

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.



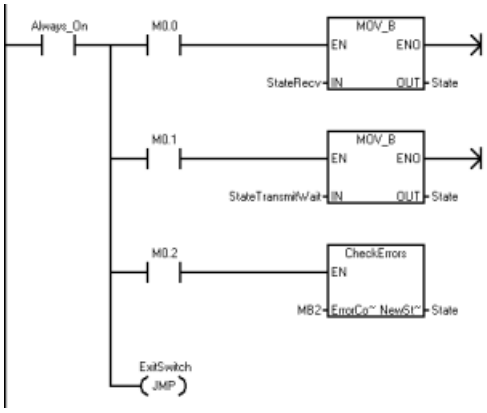
Network 15: Send the new message to the passive device. Set Req to TRUE to initiate the new send operation.



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.

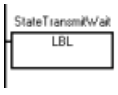
If Busy is TRUE (this is the normal situation), go to the "Transmit Wait" state to wait for the transmit to finish.

If Error is TRUE, check the reason, and you may need to change state if there is a connection issue.

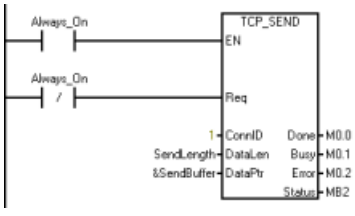


Network 17: State "Transmit Wait"...

Wait in this state until the transmit is complete.



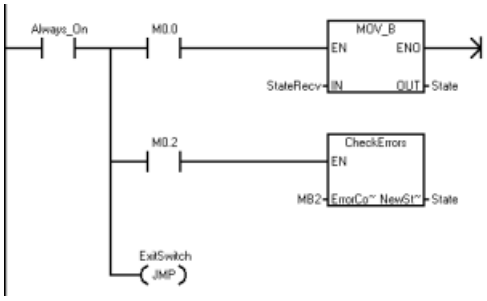
Network 18: Call the TCP_SEND instruction with Req = FALSE to determine if the send is complete. Be sure to use the same send length and buffer pointer that you used to initialize the send request.



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.



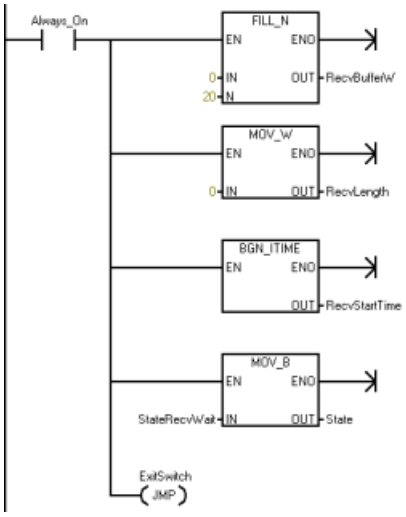
Network 20: State "Receive"...

Clean up the receive data area and prepare to receive the response.



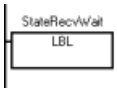
Network 21: Clear the receive buffer and the "RecvLength" so that there is no data left from the last received message.

Capture the current interval time value (in "RecvStartTime") to support a receive timeout and then go to the "Receive Wait" state.

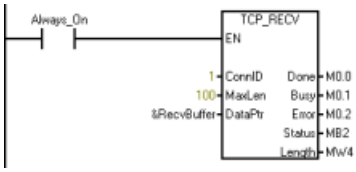


Network 22: State "Receive Wait"...

Stay in this state until you either receive some data or time out.



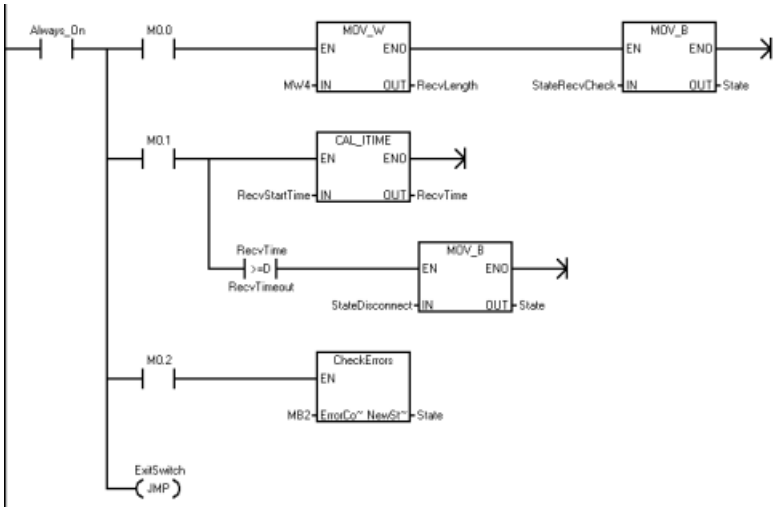
Network 23: Call the TCP_RECV instruction to get any messages that the CPU has received.



Network 24: If Done is TRUE, the instruction received new data, so go to the "Receive Check" state.

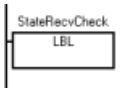
If Busy is TRUE, stay in the "Receive" state until you reach the receive timeout value. If you time out in this state, disconnect the device and then reconnect it.

If Error is TRUE, check the error code to determine what to do next.

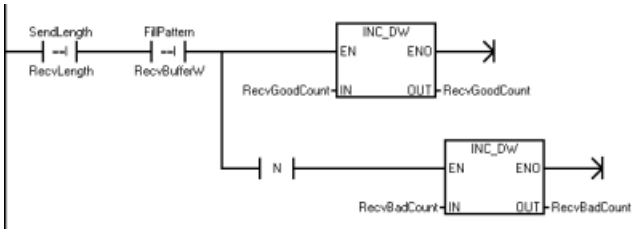


Network 25: State "RecvCheck"...

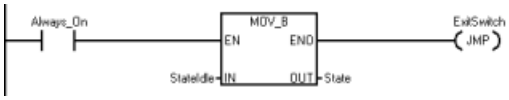
Process the data returned from the passive device.



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.

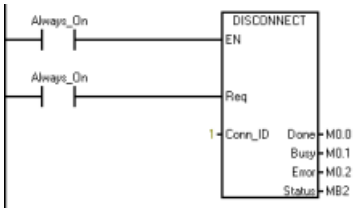


Network 28: State "Disconnect"...

Initiate a disconnect.



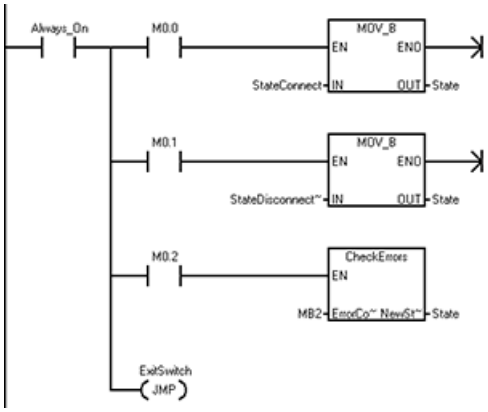
Network 29: Initiate the disconnect with the ConnID by calling the DISCONNECT instruction with Req = TRUE.



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.

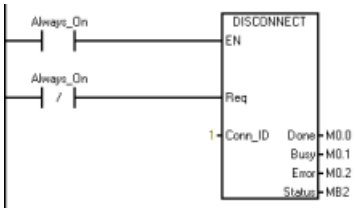


Network 31: State "Disconnect Wait"...

Wait in this state for the disconnect operation to complete.



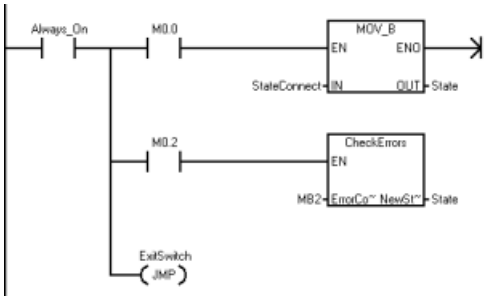
Network 32: Call the DISCONNECT instruction with Req = FALSE to check the status of the disconnect operation.



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.



11.5.5.2 CheckErrors subroutine

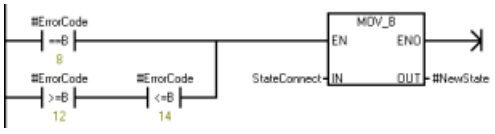
The CheckErrors subroutine checks the Open User Communication error codes and determines if the program requires a state change. You use the same CheckErrors subroutine for both the Active partner (client) and Passive partner (server).

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".

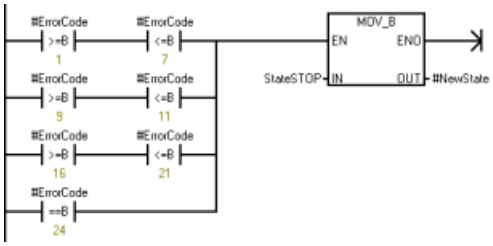


Network 3: If the error is a parameter error (error codes 1 - 7), stop the program because there is a configuration error with one of inputs to the function. Correct this in the program.

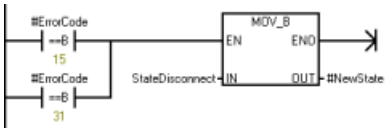
If the error code is 9 (connect in progress), 10 (disconnect in progress), or 11 (send in progress), stop because the state machine is broken. The state machine is programmed to stay in the associated waiting state until the operation is complete, and these errors should never occur.

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.

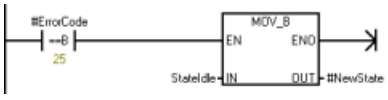
If the SEND and DISCONNECT instructions return error 24, there is no operation currently pending. This probably means that the operation is complete; however, error 24 should never appear based upon the state machine. Consider this an error, and go to stop.



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.



11.5.5.3 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

11.5.5.4 Passive partner (server)

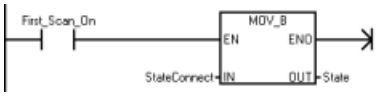
This program implements a simple state machine to manage the opening of a connection, receiving a message, sending a response, and handling errors.

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 631) to see the symbol table for this program.

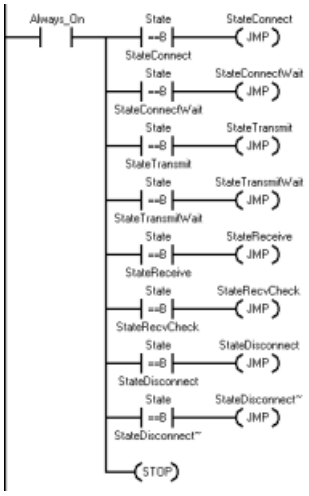
Network 1: On the first scan....

Initialize the State variable to initiate a connection.



Network 2: Process the state machine...

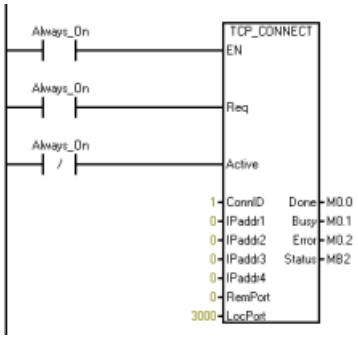
Determine the current state of the state machine and jump to the label for the state handler. If the state is ever illegal, the CPU goes to the STOP mode.



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.

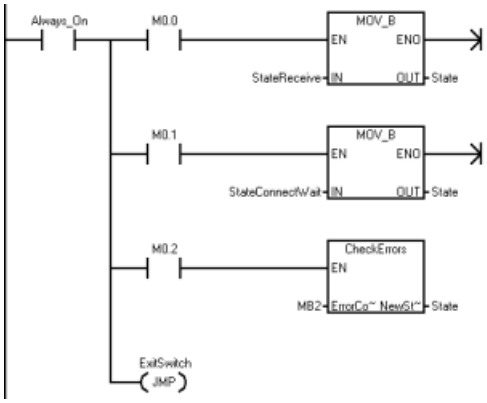


Network 5: If Done is TRUE, the CPU established the connection so go to the "Idle" state.

If Busy is TRUE, go to the "Connect Wait" state to wait for the connection to be established.

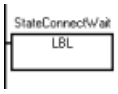
If Error is TRUE, there is probably an error with the input parameters, so check to see what state to go to next.

In all cases, exit the state machine for this scan. The program continues to the next state on the next scan.

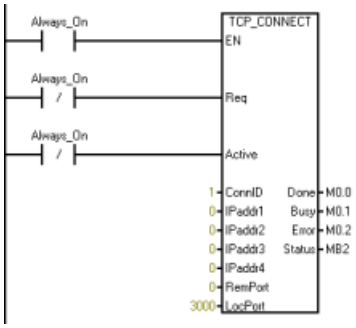


Network 6: State "Connect Wait"...

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.

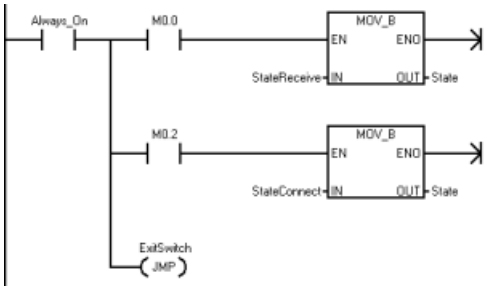


Network 8: If Done is TRUE, this means that the CPU established the connection so continue in the "Idle" state.

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.

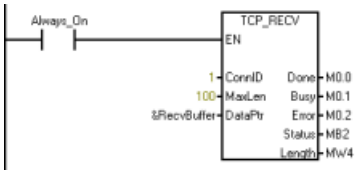


Network 9: State Receive...

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.

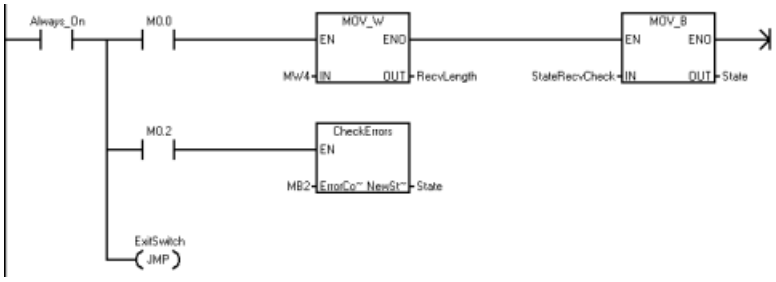


Network 11: If Done is TRUE, the CPU received new data so go to the "Receive Check" state.

If Busy is TRUE, stay in the "Receive" state until you receive some data.

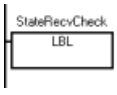
If Error is TRUE, check the error code to determine what to do next.

In all cases, exit the state machine for this scan. The program continues to the next state on the next scan.



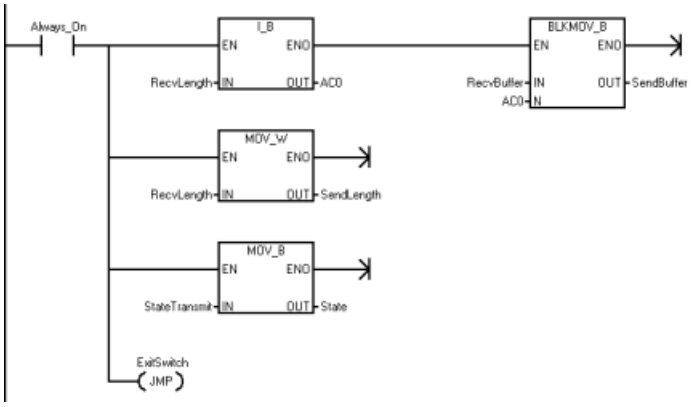
Network 12: State "Receive Check"...

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.

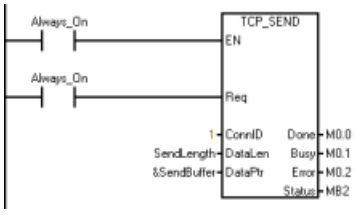
Change the state to "Transmit", and exit until the next scan.



Network 14: State "Transmit"...



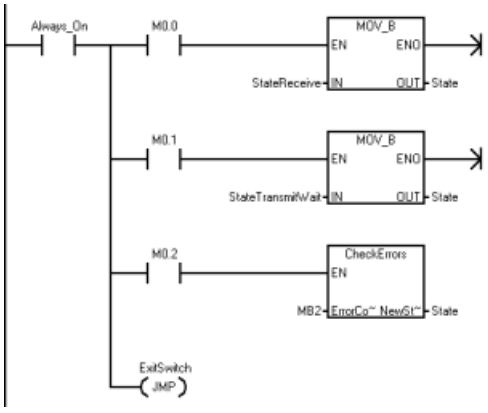
Network 15: Send the data back to the partner.



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.

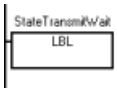
If Busy is TRUE (this is the normal situation), go to the "Transmit Wait" state to wait for the transmit to finish.

If Error is TRUE, check the reason, and you may need to change state if there is a connection issue.

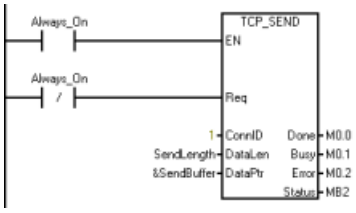


Network 17: State "Transmit Wait"...

Wait in this state until the transmit is complete.



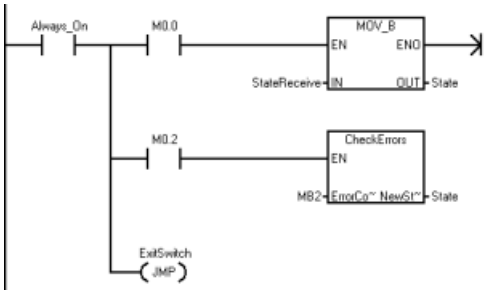
Network 18: Call the TCP_SEND instruction with Req = FALSE to determine when the send is complete. Be sure to use the same send length and buffer pointer that you used to initialize the send request.



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.

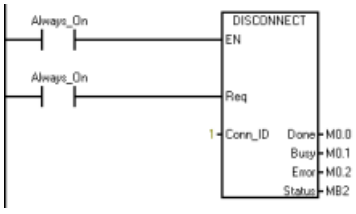


Network 20: State "Disconnect"...

Initiate a disconnect.



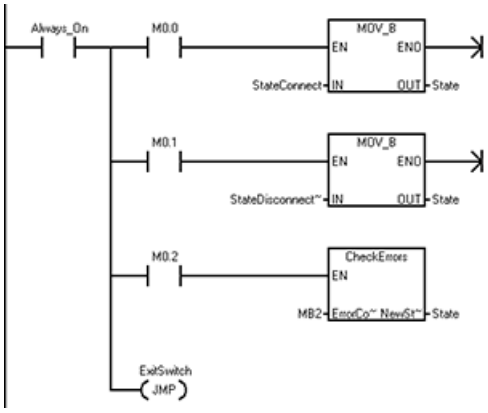
Network 21: Initiate the disconnect with the ConnID by calling the DISCONNECT instruction with Req = TRUE.



Network 22: If Done is TRUE, this means that the disconnect is complete (probably will never happen), so try to reconnect.

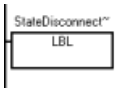
If Busy is TRUE (this would be normal), go to the "Disconnect Wait" state.

If Error is TRUE, check the reason, and you may need to change state if there is a connection issue.

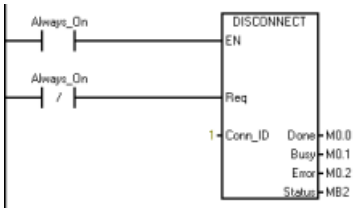


Network 23: State "Disconnect Wait"...

Wait in this state for the disconnect operation to complete.



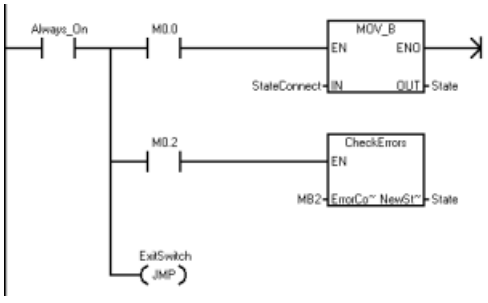
Network 24: Call the DISCONNECT instruction with Req = FALSE to check the status of the disconnect operation.



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.



11.5.5.5 CheckErrors subroutine

The CheckErrors subroutine (Page 622) 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).

11.5.5.6 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	

11.6 PN Read Write Record library

11.6.1 PN Read Write Record features

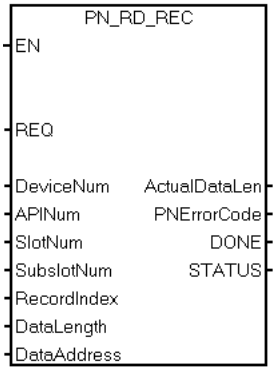
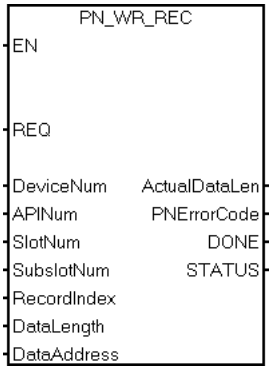
PN Read Write Record library includes the following two instructions:

- PN_RD_REC: Read a data record from any connected PROFINET device
- PN_WR_REC: Write a data record to any connected PROFINET device.

11.6.2 Input and output interface of PN Read Write Record library

The PN_RD_REC and PN_WR_REC instructions are as follows:

Table 11-26 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 11-27 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 633).

11.6.3 Definition of parameters for input signal "STATUS"

The following table lists the parameter information of "STATUS":

Table 11-28 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 634).

11.6.4 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.

11.7 USS Protocol library instructions

11.7.1 USS communication overview

11.7.1.1 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 636)
- Calculating the time required for communicating with the drive (Page 636)

Refer to "Using the USS protocol instructions (Page 637)" 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.

11.7.1.2 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.

11.7.1.3 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 11-29 Communications times

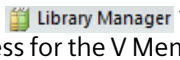
Baud rate	Time between polls of active drives (with no parameter access instructions active)
1200	240 ms (maximum) times the number of drives
2400	130 ms (maximum) times the number of drives
4800	75 ms (maximum) times the number of drives
9600	50 ms (maximum) times the number of drives
19200	35 ms (maximum) 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

11.7.2 USS program instructions

11.7.2.1 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 

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.

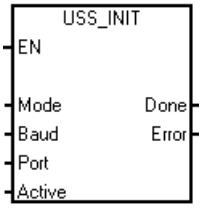
The USS protocol instructions consist of the following:

- USS_INIT (Page 638)
- USS_CTRL (Page 640)
- USS_RPM_X (Page 643)
- USS_WPM_x (Page 646)

USS protocol program examples (Page 648) and a listing of USS protocol error codes (Page 650) are also discussed in this section.

11.7.2.2 USS_INIT instruction

Table 11-30 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.

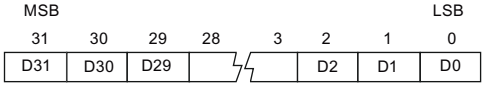
Table 11-31 Parameters for the USS_INIT instruction

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 11-32 USS_INIT parameter descriptions

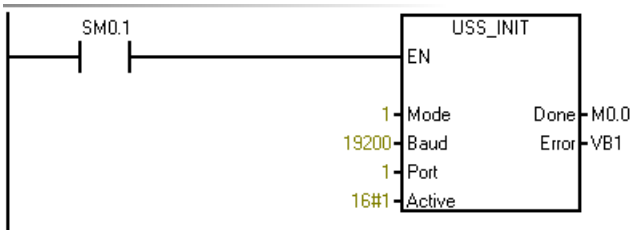
Parameter	Description
Mode	This value selects the communications protocol: • An input value of 1 assigns the port to USS protocol and enables the protocol. • An input value of 0 assigns the port to PPI and disables the USS protocol.
Baud	Sets the baud rate at 1200, 2400, 4800, 9600, 19200, 38400, 57600, 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 650) define the error conditions that could result from executing the instruction.

Table 11-33 Format for the Active drive parameter

	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 650) to compute the time between status polls and the error conditions that could result from executing the instruction.

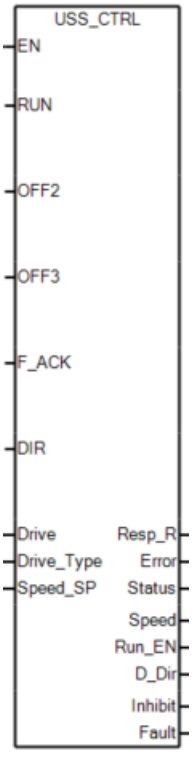
Table 11-34 USS_INIT example program

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 637) for and a listing of USS protocol instructions and error codes and example programs.

11.7.2.3 USS_CTRL instruction

Table 11-35 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 11-36 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.

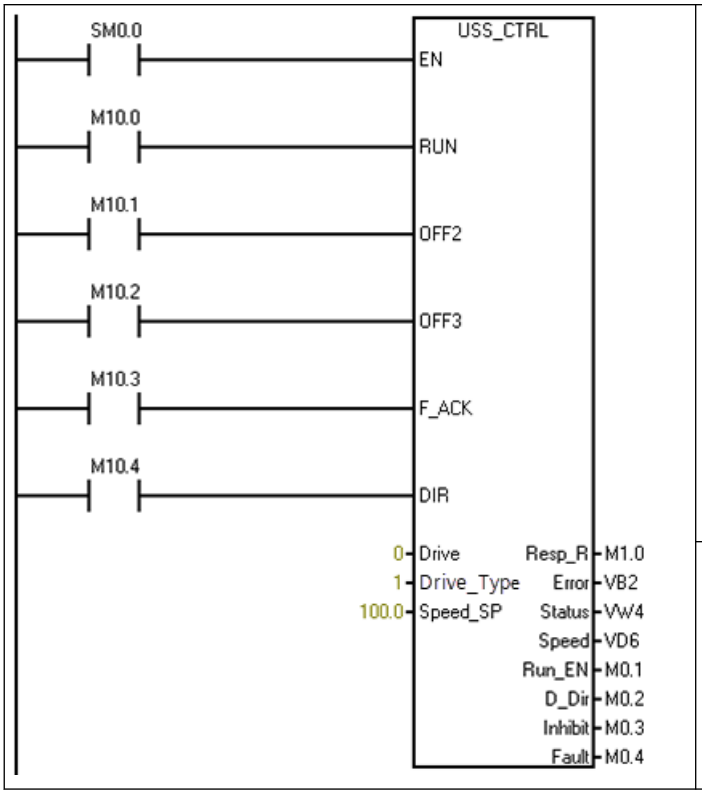
Resp_R parameter

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 650) 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 11-37 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
```

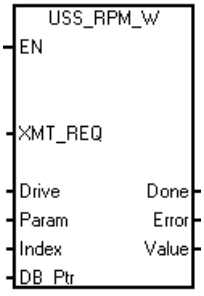
To display in STL only:

```
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 637) for a listing of USS protocol instructions and error codes and example programs.

11.7.2.4 USS_RPM_x instruction

Table 11-38 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 11-39 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.

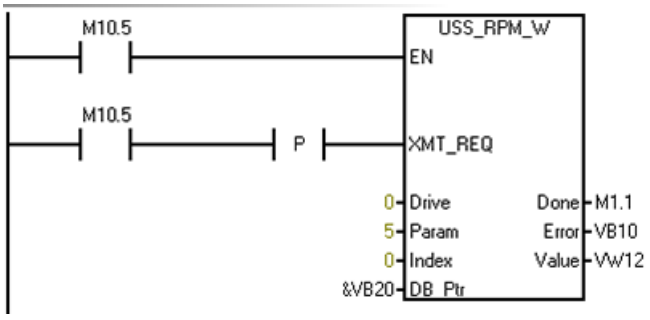
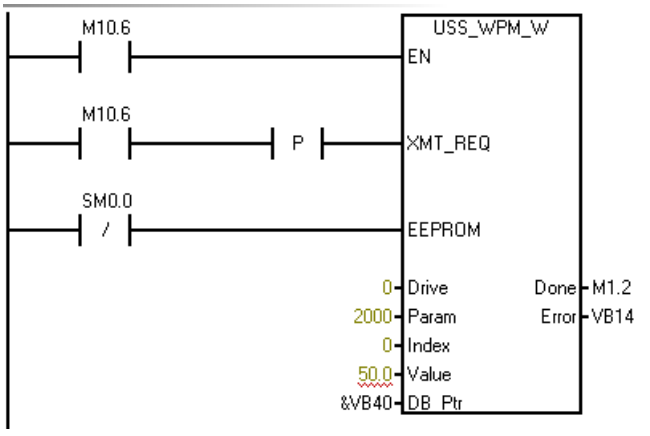
Table 11-40 USS_RPM_x parameter descriptions

Parameter	Description
XMT_REQ (transmit request)	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 650) 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

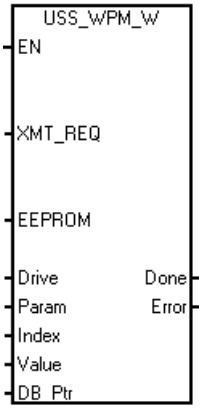
Table 11-41 USS_RPM_x and USS_WPM_x example program

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 637) for and a listing of USS protocol instructions and error codes and example programs.

11.7.2.5 USS_WPM_x instruction

Table 11-42 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 11-43 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 11-44 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 650) 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.

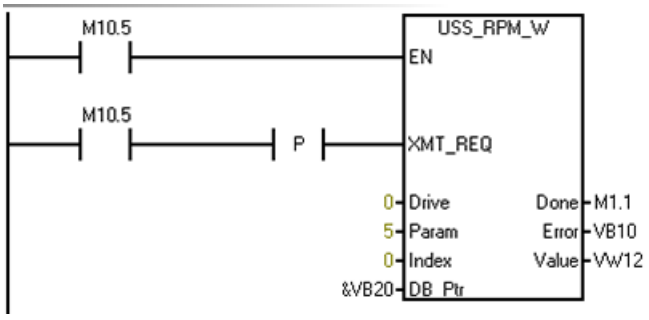
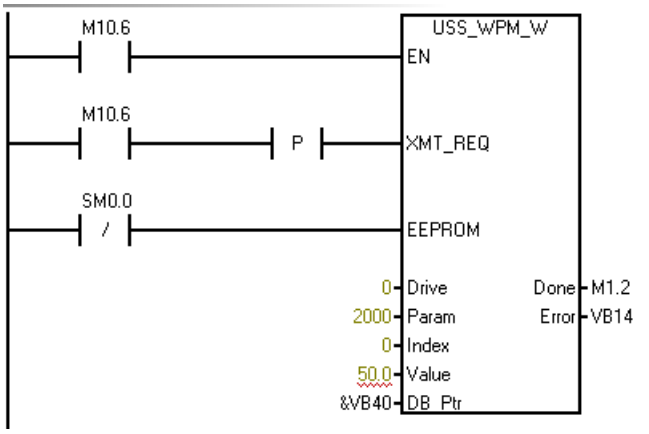
EEPROM

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 11-45 USS_RPM_x and USS_WPM_x example program

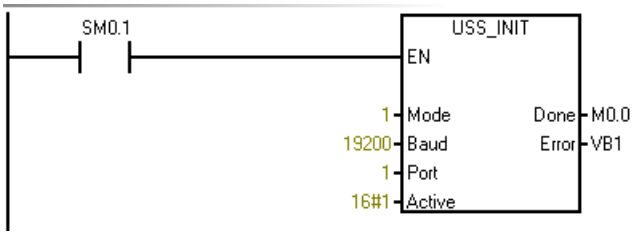
Network 1	Network 1
	**LD M10.5** **= L60.0** **LD M10.5** **EU** **= L63.7** **LD L60.0** **CALL USS_RPM_W, L63.7, 0, 5, 0, &VB20, M1.1, VB10, VW12**
Network 2	**Network 2**
	**LD M10.6** **= L60.0** **LD M10.6** **EU** **= L63.7** **LDN SM0.0** **= L63.6** **LD L60.0** **CALL USS_WPM_W, L63.7, L63.6, 0, 2000, 0, 50.0, &VB40, M1.2, VB14**

Refer to "Using the USS protocol instructions" (Page 637) for and a listing of USS protocol instructions and error codes and example programs.

11.7.2.6 USS protocol example program

Table 11-46 Sample USS program

Network 1

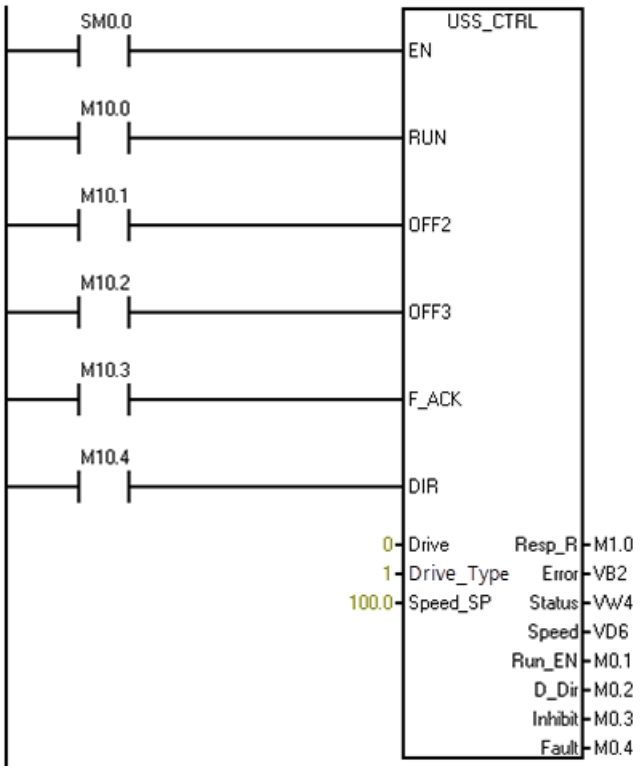


Network 1:
Initialize USS protocol: On the first scan, enable USS protocol for port 1 at 19200 with drive address "0" active.

```
LD SM0.1
CALL USS_INIT, 1, 19200, 16#00000001,
Q0.0, VB1
```

Network 2

Network 2:
Control parameters for Drive 0

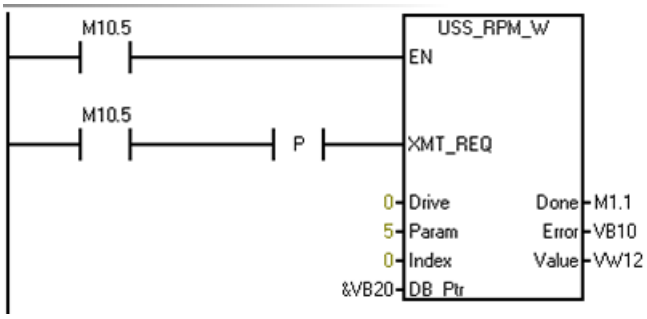


Network 3

```
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:
Read a Word parameter from Drive 0.
Read parameter 5, index 0:

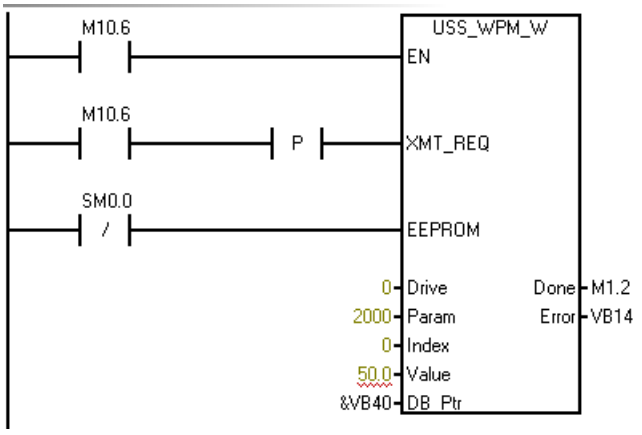
1. Save the state of M10.5 to a temporary location so that this network displays in LAD.
2. Save the rising edge pulse of I0.5 to a temporary L location so that it can be passed to the subroutine.



Network 4

```
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:
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 637) for and a listing of USS protocol instructions and error codes and example programs.

11.7.3 USS protocol execution error codes

Table 11-47 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 637) for and a listing of USS protocol instructions and error codes and example programs.

11.8 SINAMICS Library

11.8.1 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 652): Control the drive position with 8 different operating modes
  - SINA_SPEED (Page 692): Control the drive speed
- SINAMICS_Parameter:
  SINA_PARA_S (Page 698): 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

11.8.2 SINA_POS instruction

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.

11.8.2.1 Prerequisite of using the SINA_POS instruction

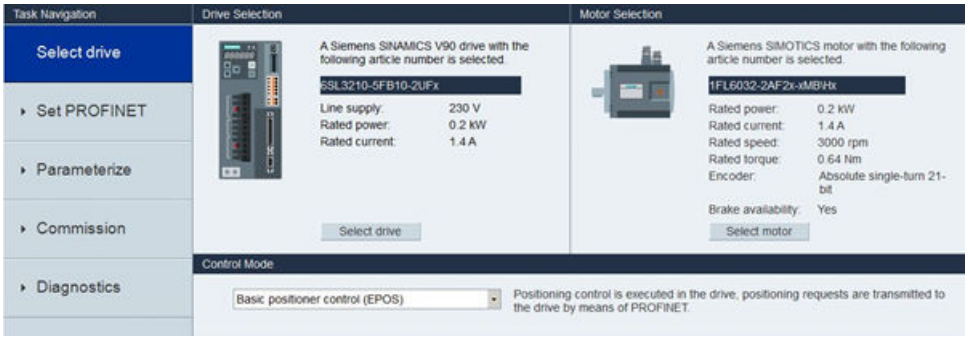
The prerequisite for using the SINA_POS instruction is as follows:

- 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.

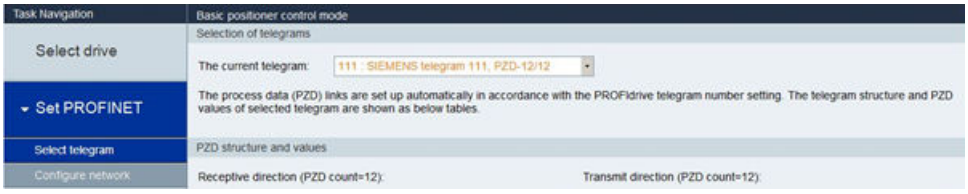
Procedure of configuring SINAMICS V90 PN parameters with V-assistant

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.

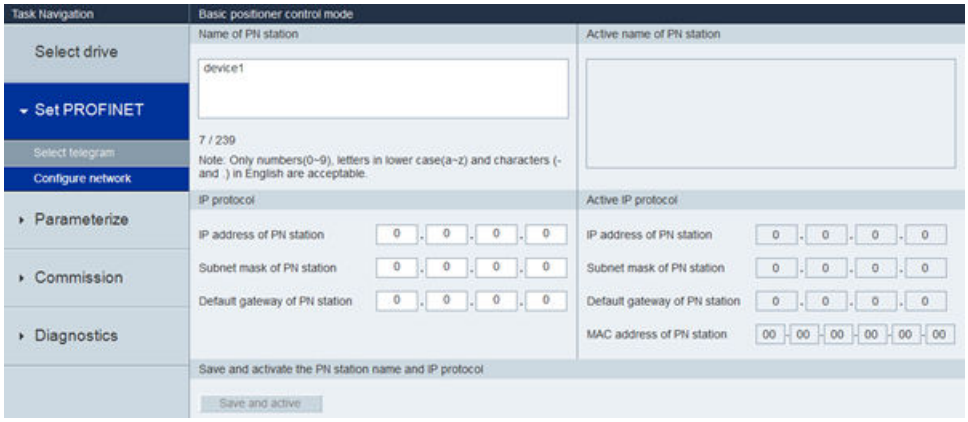


Note

When configuring the PROFINET network in the hardware network topology, ensure that the telegrams are consistent with those 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 configuring the PROFINET network in the hardware network topology, ensure that the device name matches 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.

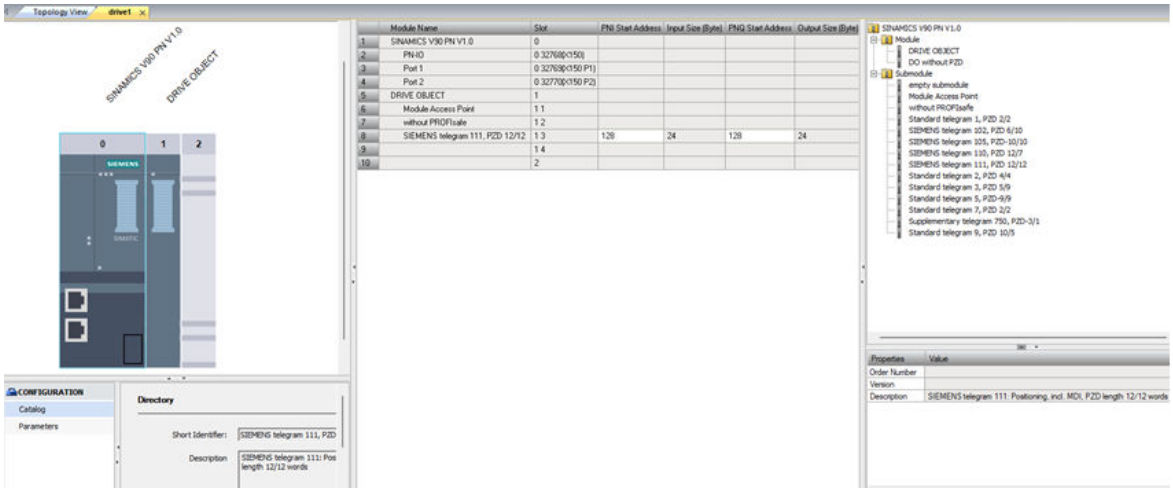


Figure 11-2 Example of PROFINET network configuration in hardware network topology

11.8.2.2 Input and output interface of SINA_POS instruction

The SINA_POS instruction can control and set the position for drives.

Table 11-48 SINA_POS Instruction

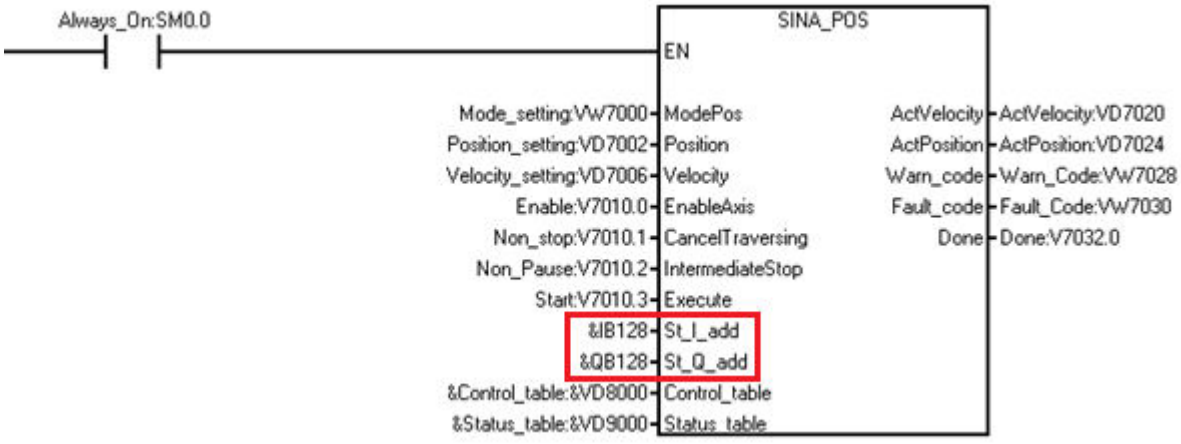
LAD/FBD	STL	Description
SINA_POS EN ModePos — ActVelocity Position — ActPosition Velocity — Warn_code EnableAxis — Fault_code CancelTraversing — Done IntermediateStop Execute St_I_add St_Q_add Control_table Status_table	CALL SINA_POS, ModePos, Position, Velocity, EnableAxis, CancelTraversing, IntermediateStop, 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:



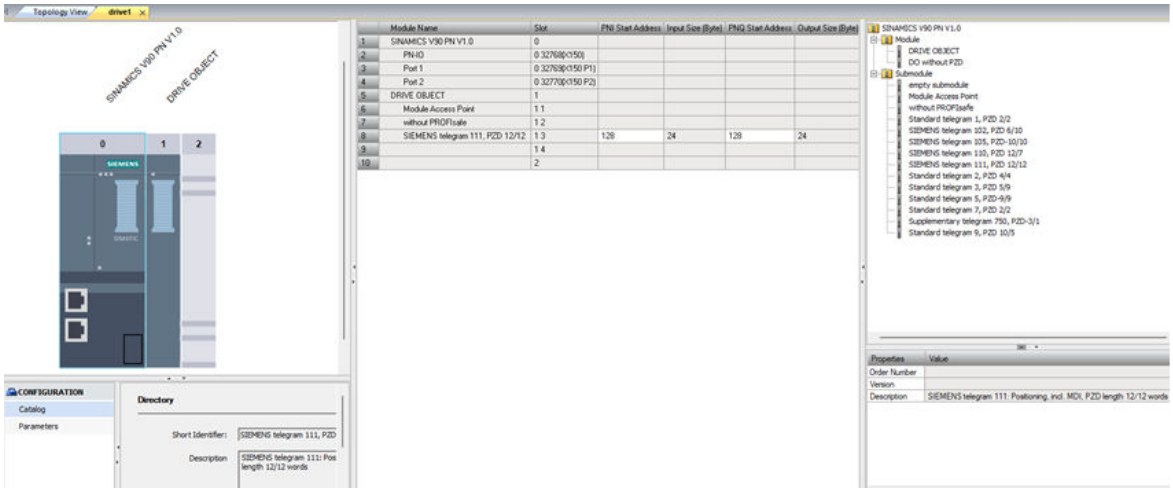


Table 11-49 Parameters of SINA_POS instruction

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 value = 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)
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.

Parameter and type		Data type	Description
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. For example, &VD8000.
Status_table	IN	DWORD	Pointer of the start address of Status_table. 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 11-50 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								
8	ConfigEpos [10]							
9								
10								
11								

Table 11-51 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 659).

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.%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 11-52 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 660).

[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 660).

[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 661).(Default = 0)

[11] Epos_zsw2: Status of EPos_zsw2 (bit-granular). For the detailed information, refer to Assignment of "Epos_zsw2" (Page 661).(Default = 0)

Error codes for the "Status_table" parameter

The following table lists the error code of the "Status_table" parameter:

Table 11-53 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.

Error Code	Description
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 11-54 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 11-55 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		

Bit	Abbr.	Designation	Drive parameter	Function chart
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

11.8.2.3 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.

11.8.2.4 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.

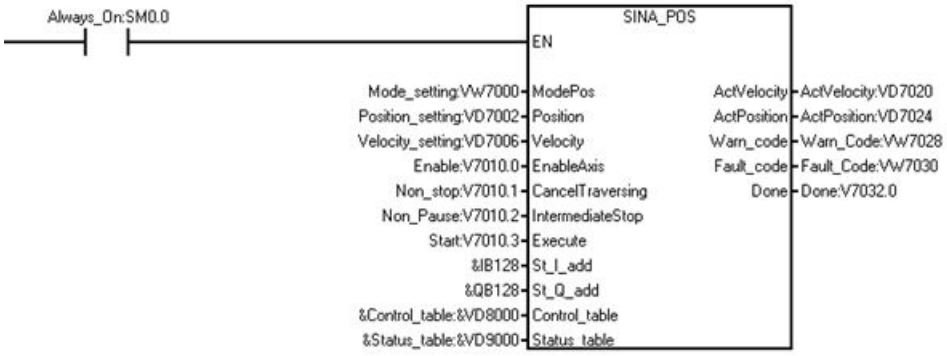
Traversing motion is started with a positive edge at "Execute". You can track the current state of the active command with "EPos_zsw1 / EPos_zsw2".

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 657).

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".

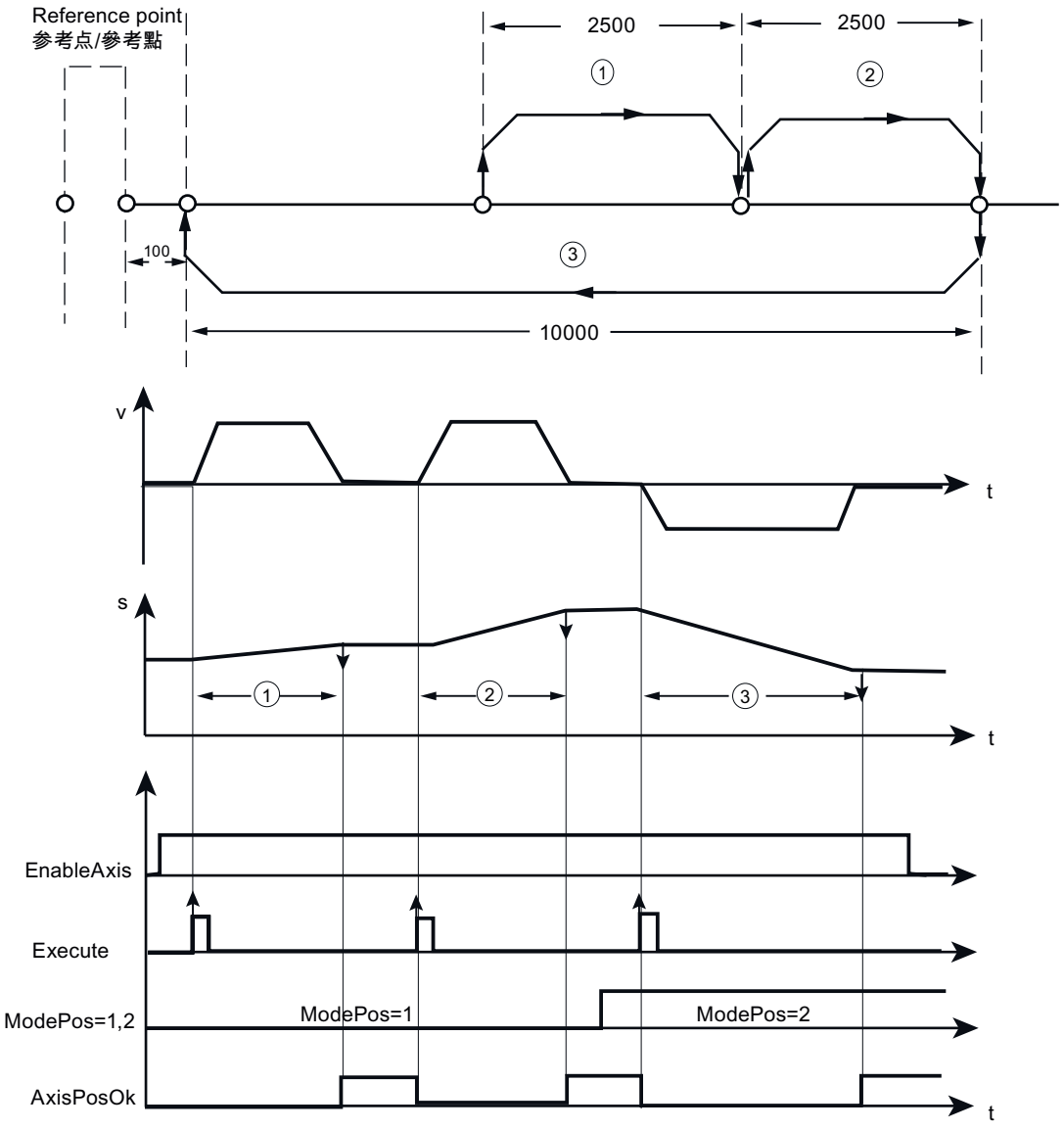
Note

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.

11.8.2.5 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

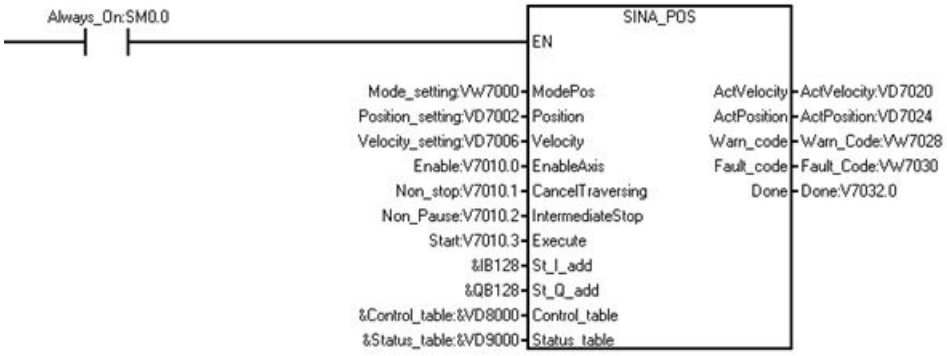
Traversing motion is started with a positive edge at "Execute". You can track the current state of the active command with "EPoszsw1 / EPoszsw2".

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.

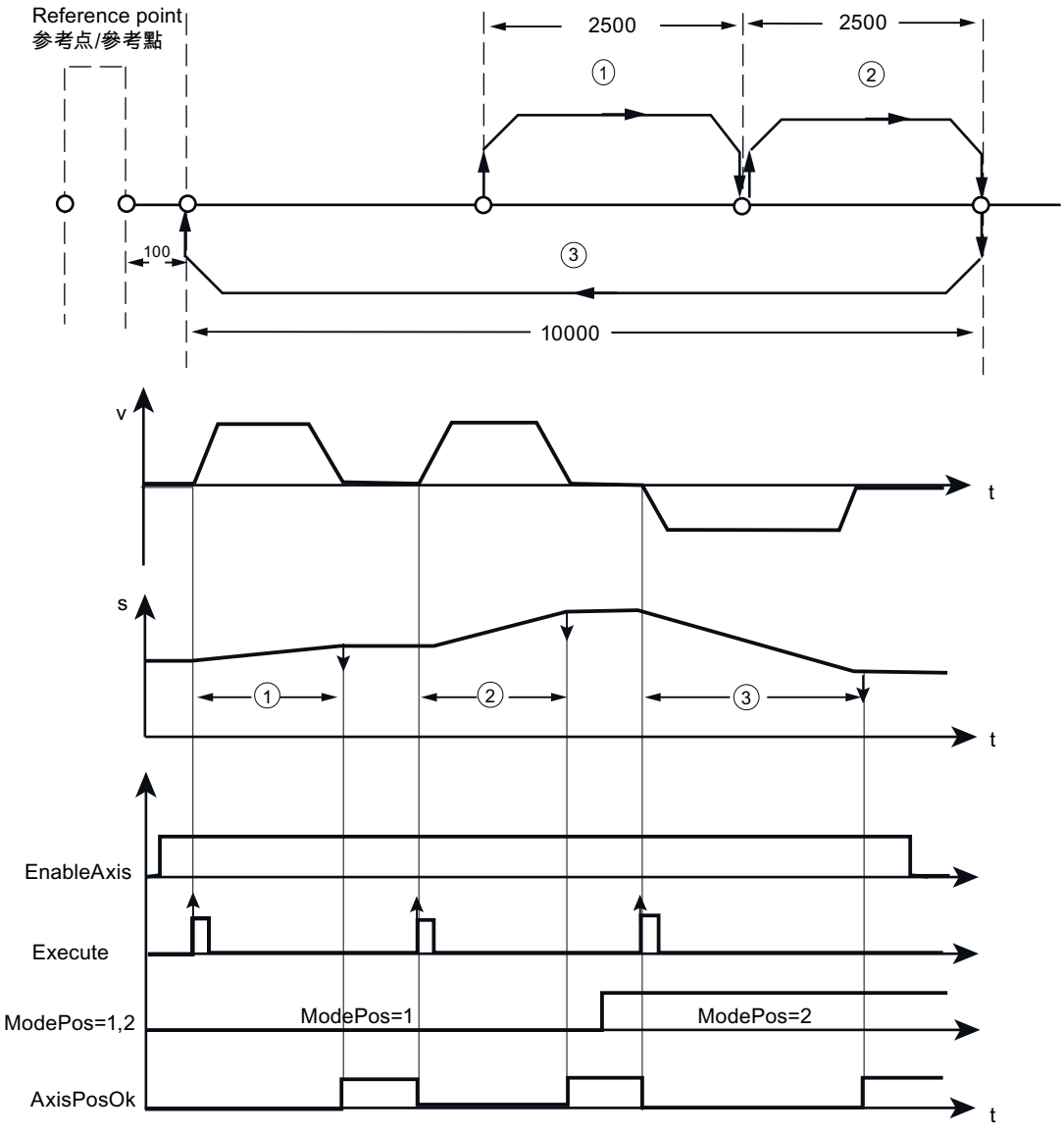
For the input operand "St_I_add" and "St_Q_add", keep the offset consistent with that in PROFINET wizard.

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.

11.8.2.6 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".

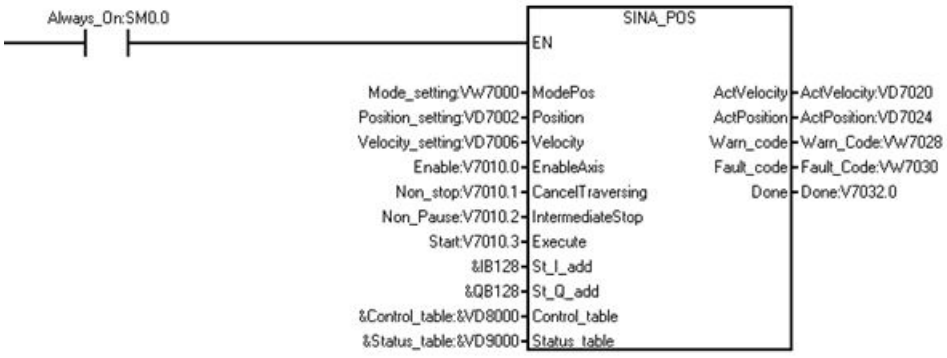
The output signal "AxisPosOk" is set when the setup mode is terminated with reject traversing task and the axis has stopped.

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

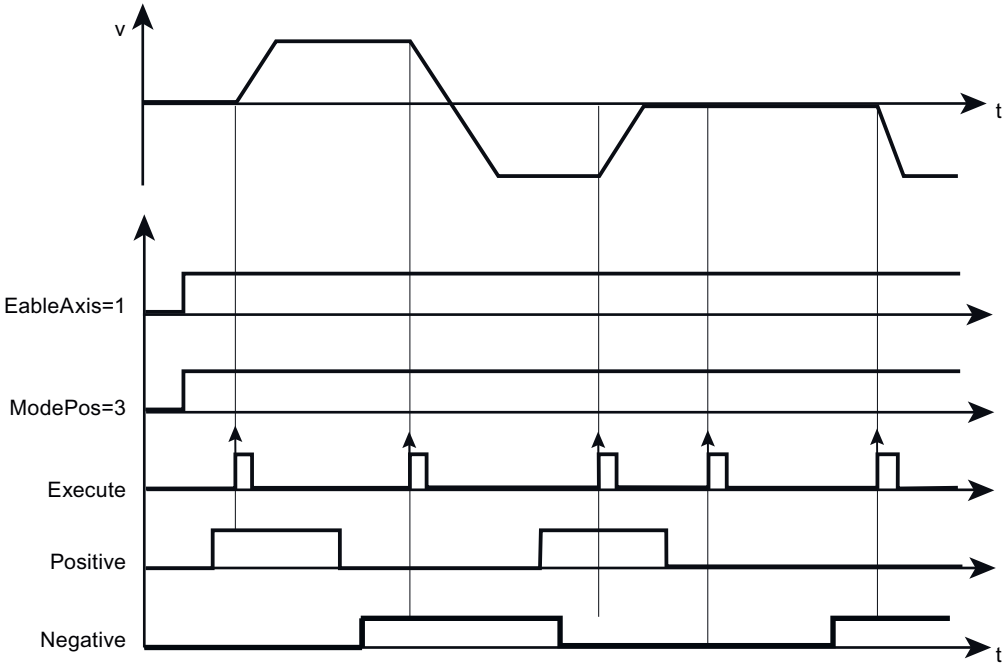
For the four inputs "St_I_add", "St_Q_add", "Control_table", and "Status_table", the mode of addressing instruction operands is the indirect addressing. You must enter an ampersand (&) at the beginning of the input operand.

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.

11.8.2.7 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.

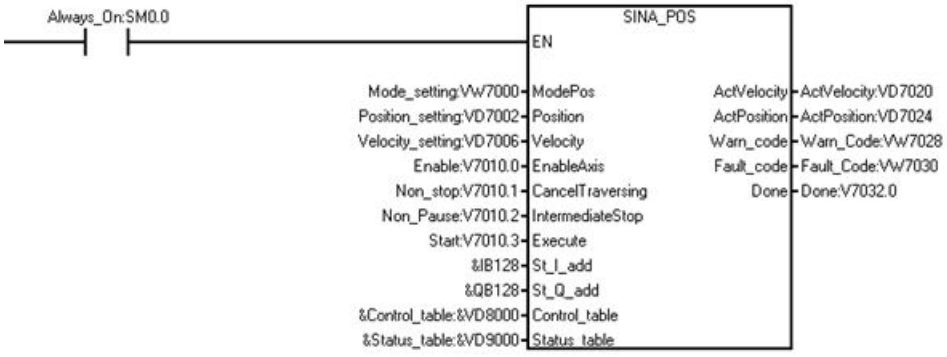
The reference point approach is started with a positive edge at "Execute".

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.

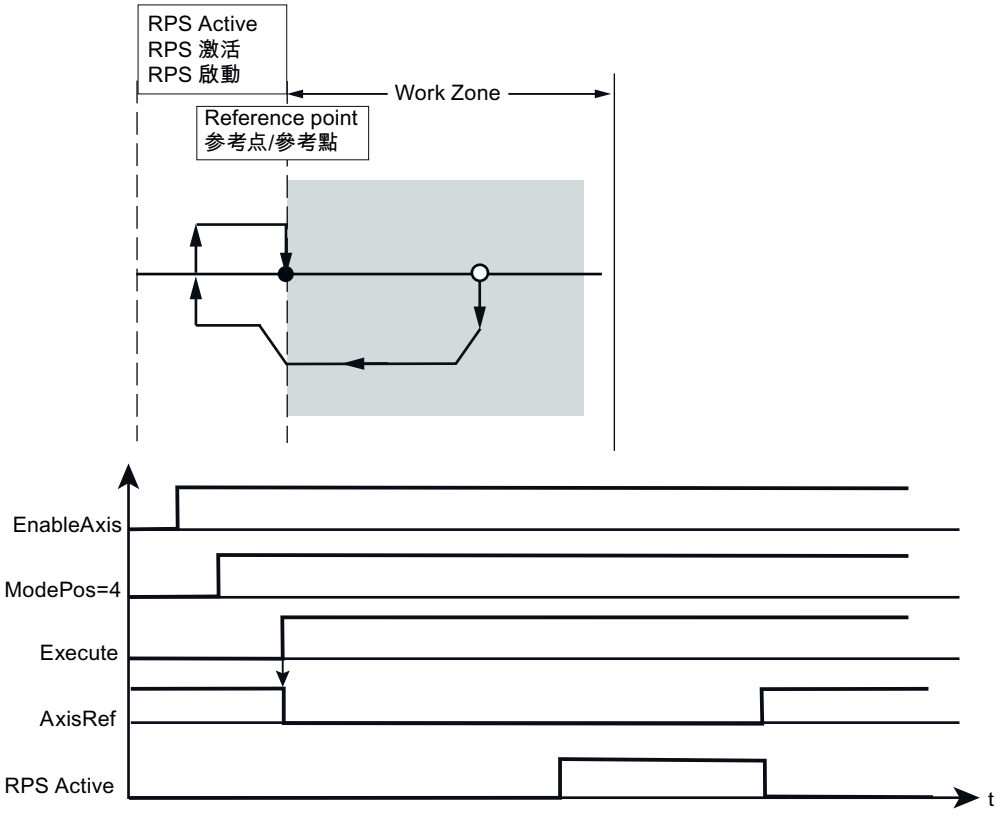
For the input operands "St_I_add" and "St_Q_add", keep the offset consistent with that in the PROFINET wizard.

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.

11.8.2.8 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

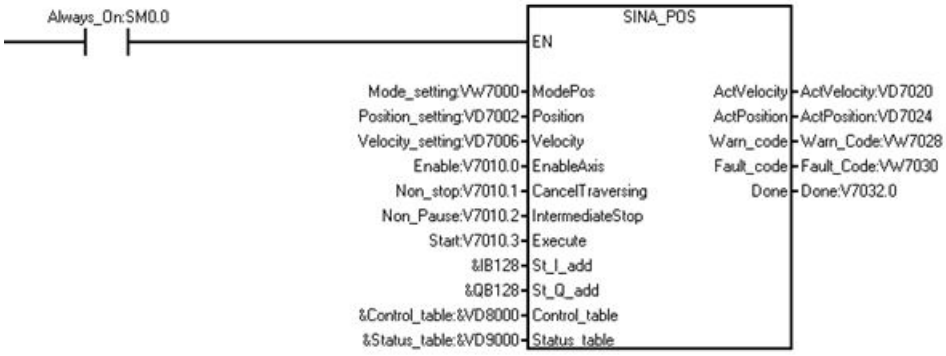
- The mode is selected with "ModePos"=5.
- The axis can be in closed-loop control, but must be at a standstill.

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	Config-Epos	3
Status_table	VD9000	The status parameters.	Status_table	DWORD	V9000.2	Axis-Ref	1

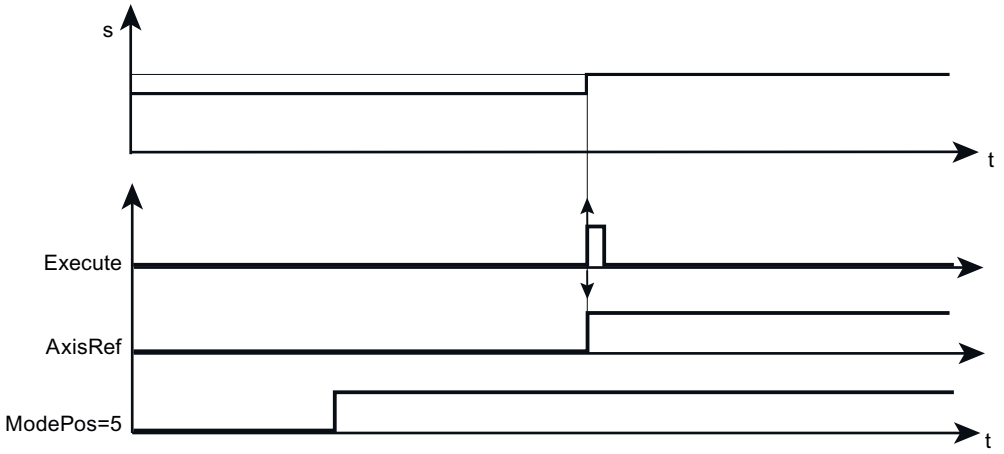
2. Enter the data in the input "St_I_add" and "St_Q_add".

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.

11.8.2.9 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 660)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.

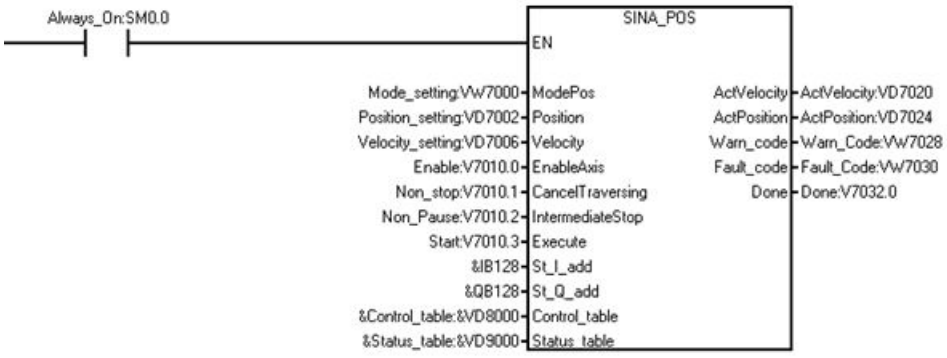
Traversing motion is started with a positive edge at "ExecuteMode". You can track the current state of the active command with "EPos_zsw1 / EPos_zsw2".

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.

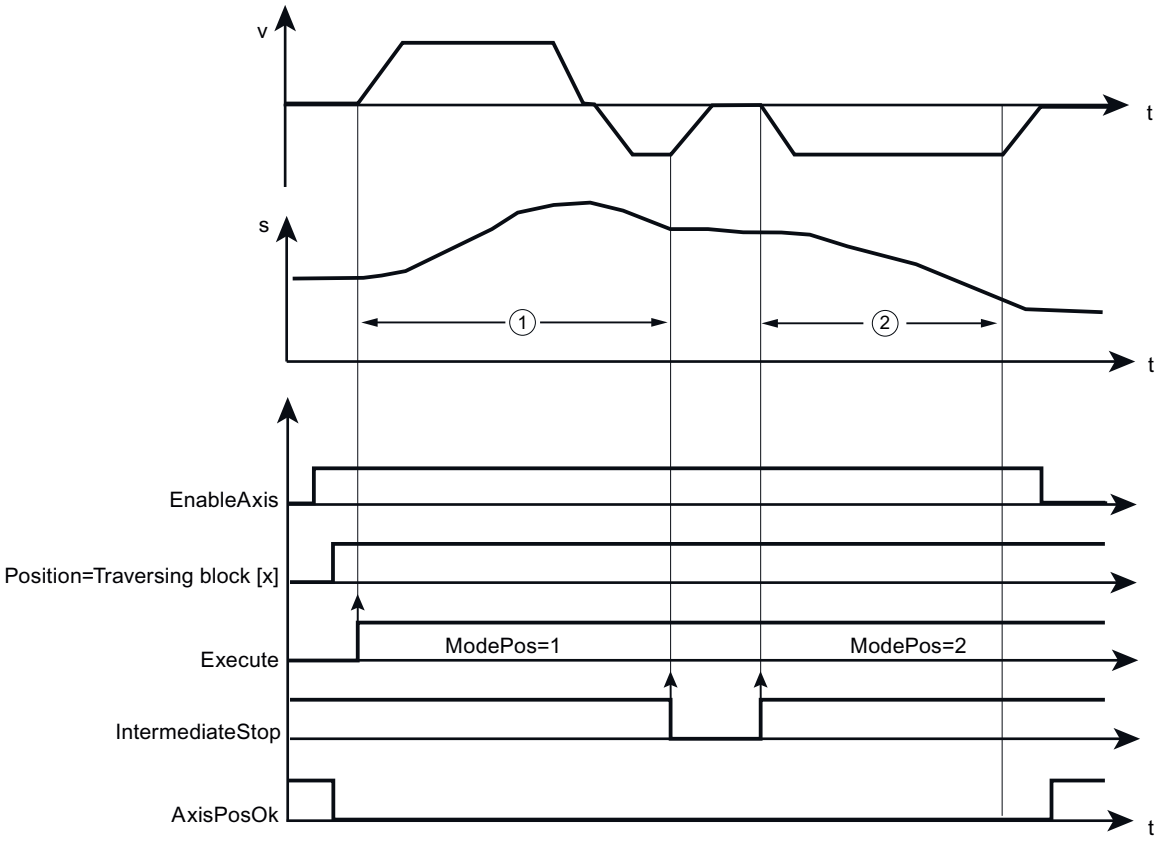
2. Enter the data in the input "St_I_add" and "St_Q_add".

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.

11.8.2.10 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".

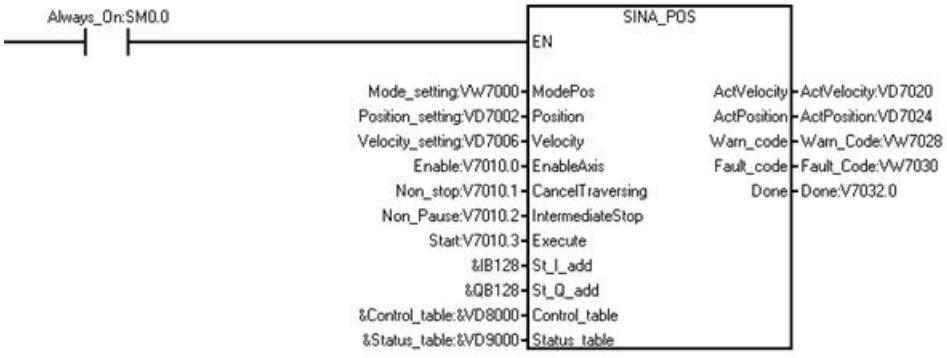
You can track the current state of the active command with "EPos_zsw1 / EPos_zsw2".

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	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".

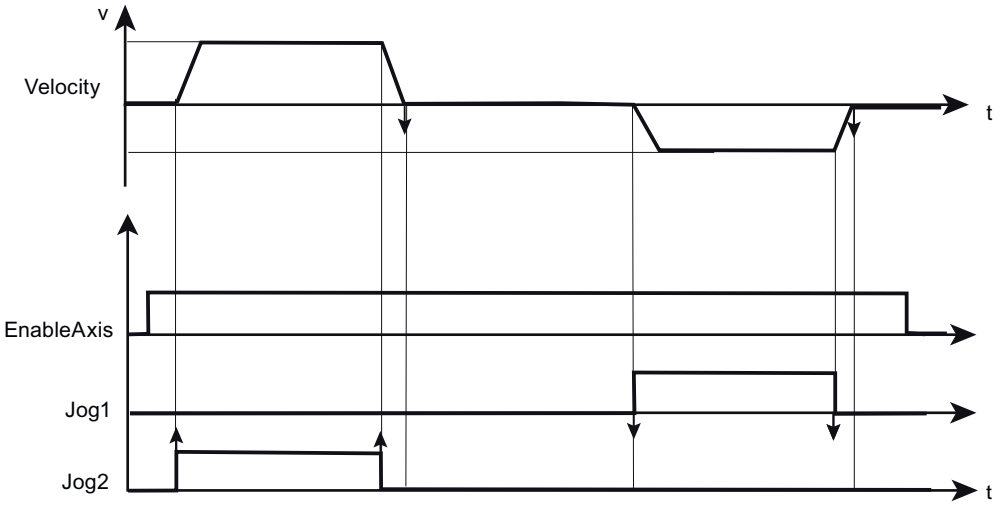
Note

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 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.

11.8.2.11 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.

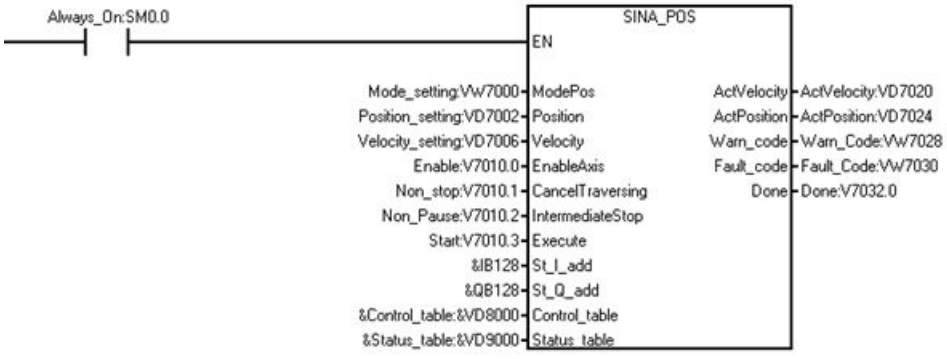
The current state of the active command can be tracked via "EPosZSW1 / EPosZSW2".

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.

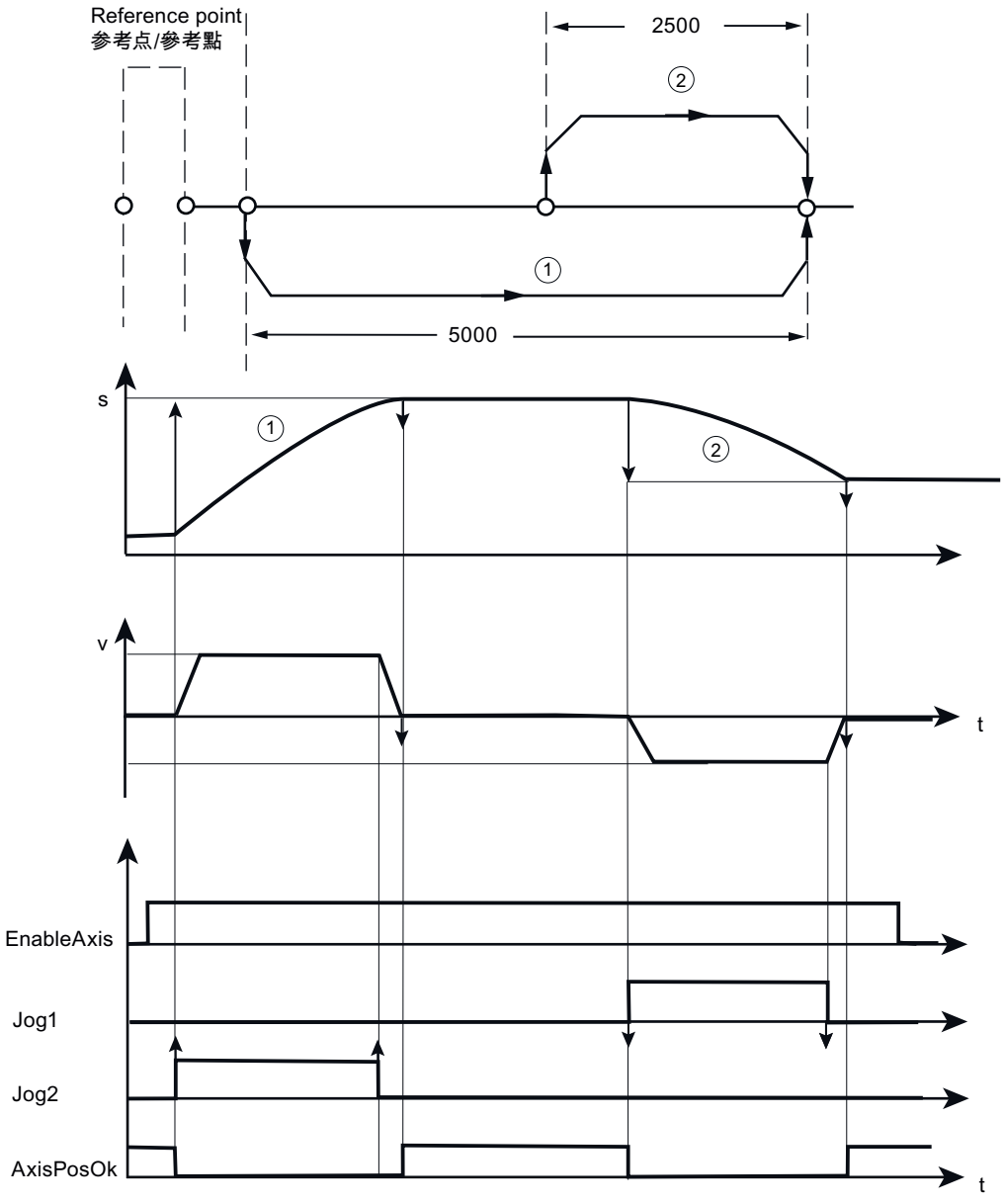
For the input operand "St_I_add" and "St_Q_add", keep the offset consistent with that in PROFINET wizard.

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.

11.8.2.12 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

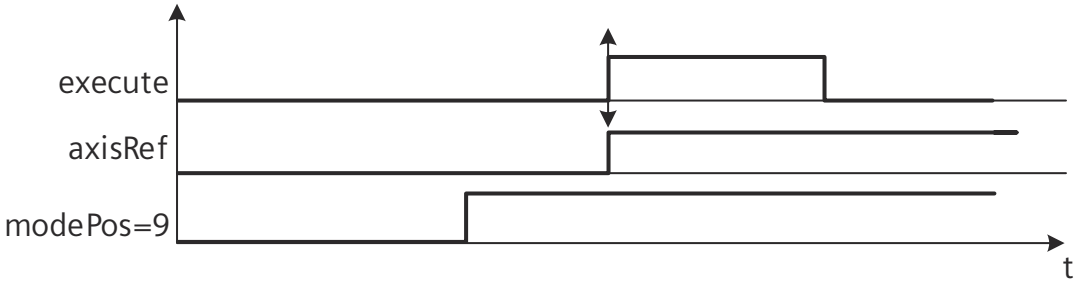
After the absolute encoder is adjusted, the offset should be saved into the drive.

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.



11.8.3 SINA_SPEED instruction

11.8.3.1 Prerequisite of using the SINA_SPEED instruction

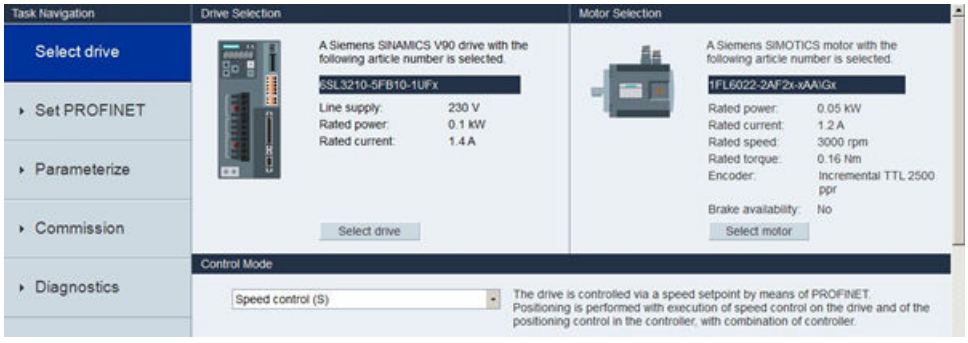
The prerequisite for using the Speed_control instruction is as follows:

- 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".

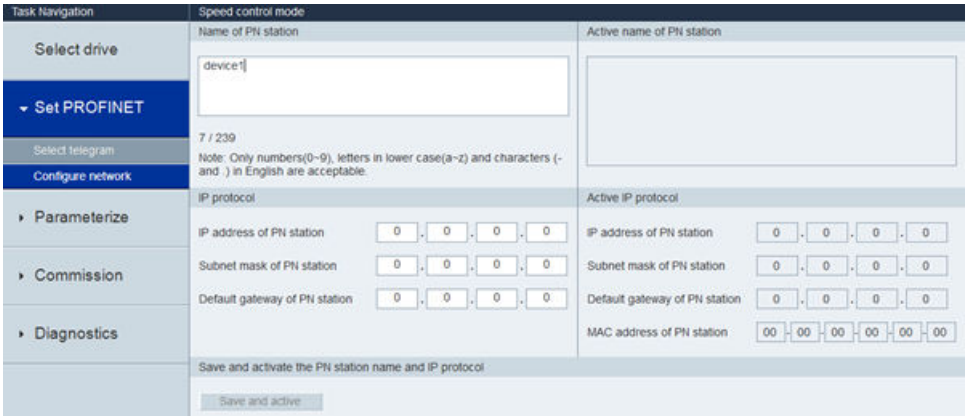
6. In the "Selection of telegram" field, select "Standard telegram 1" as the current telegram. The SINA_SPEED instruction only supports Standard telegram 1.



Note

When configuring the PROFINET network in the hardware network topology, ensure that the telegrams are consistent with those 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 configuring the PROFINET network in the hardware network topology, ensure that the device name matches the PN station name in this step.

9. Click the "Save and active" button.

Result: The drive restarts and the configured parameters take effect.

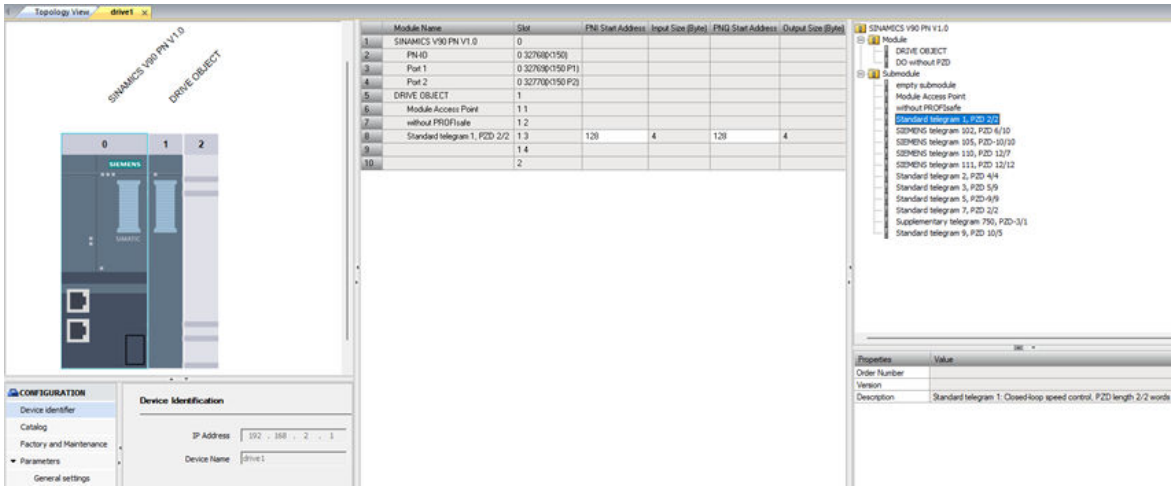


Figure 11-3 Example of PROFINET network configuration in hardware network topology

11.8.3.2 Input and output interface of SINA_SPEED instruction

The SINA_SPEED instruction is used for setting the drive speed.

Table 11-56 SINA_SPEED instruction

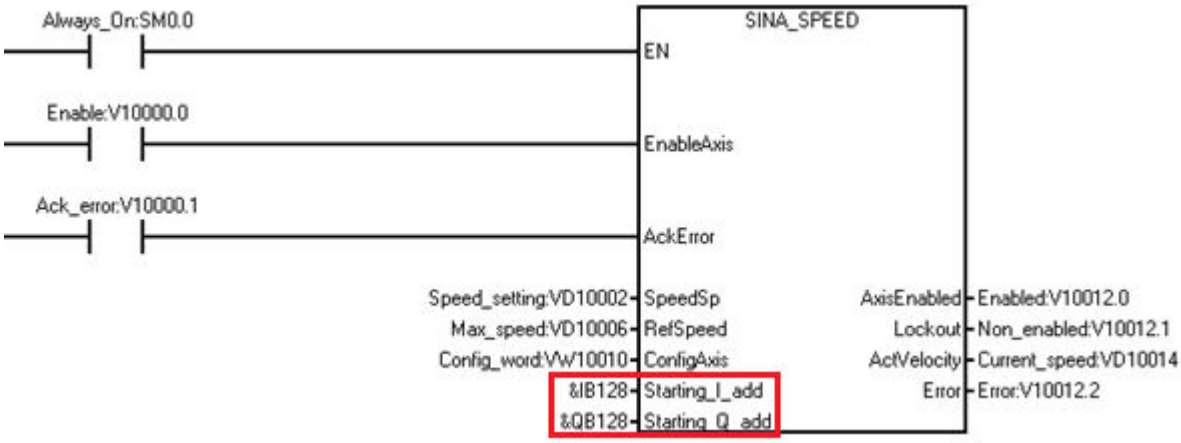
LAD/FBD	STL	Description
SINA_SPEED EN EnableAxis AckError SpeedSp — AxisEnabled RefSpeed — Lockout ConfigAxis — ActVelocity Starting_I_add — Error Starting_Q_add	CALL SINA_SPEED,EnableAxis, AckError, SpeedSp, RefSpeed, ConfigAxis Starting_I_add,Starting_Q_add, AxisEnabled, Lockout, ActVelocity, Error	The SINA_SPEED instruction enables the speed control of drive movement.

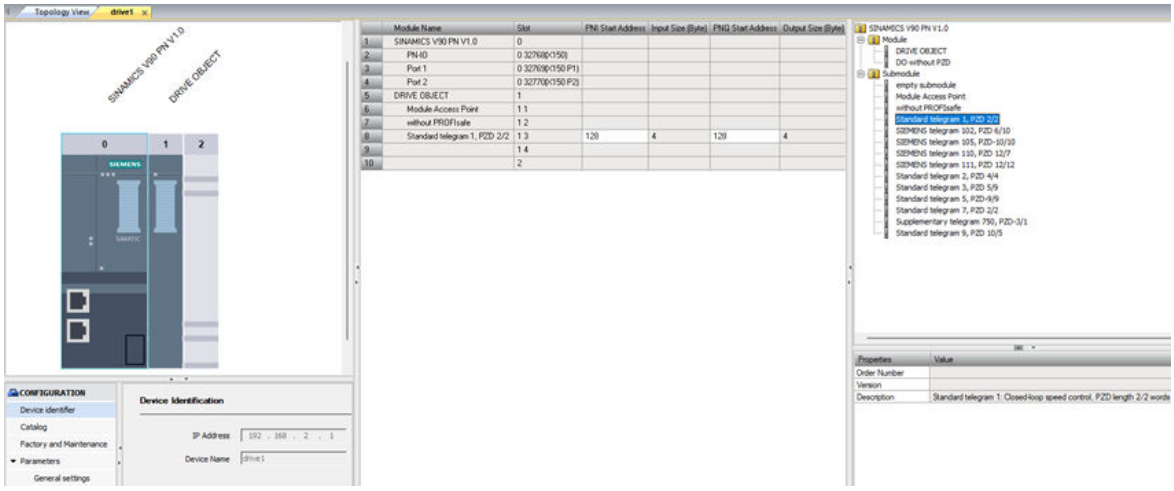
Note

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 11-57 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	Velocity setpoint. The value of SpeedSp varies 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 more information, refer to "Definition of "ConfigAxis" parameters (Page 696)". (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.

Parameter and type		Data type	Description
ActVelocity	OUT	REAL	Actual velocity ->dependent on scaling factor RefSpeed.
Error	OUT	BOOL	1 = group fault active.

11.8.3.3 Definition of "ConfigAxis" parameters

The following table lists the definition of "ConfigAxis" parameters:

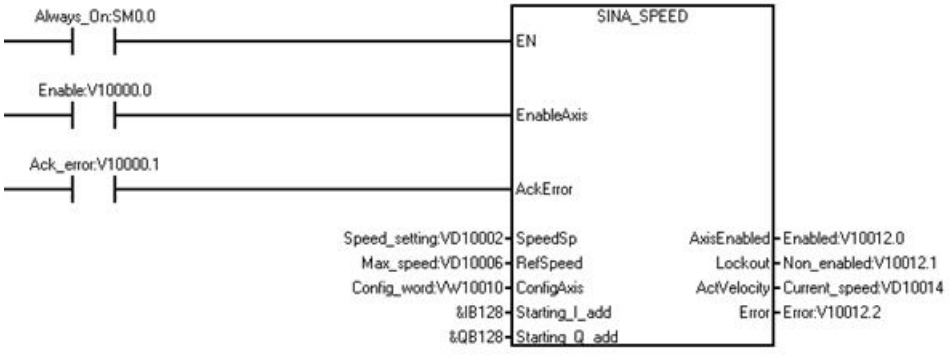
Table 11-58 ConfigAxis

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

11.8.3.4 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.

11.8.4 SINA_PARA_S instruction

11.8.4.1 Prerequisite of using the SINA_PARA_S instruction

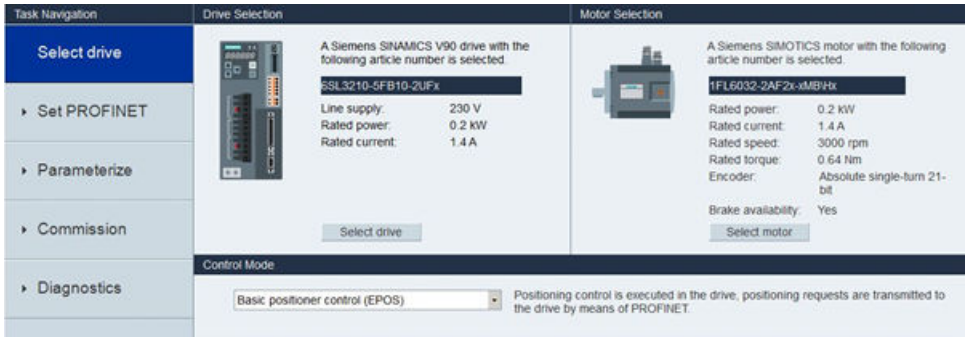
The prerequisite for using the SINA_PARA_S instruction is as follows:

- 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

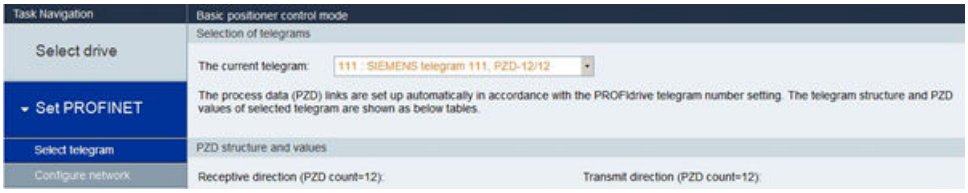
Configure the V90 PN parameters with V-assistant as follows:

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".

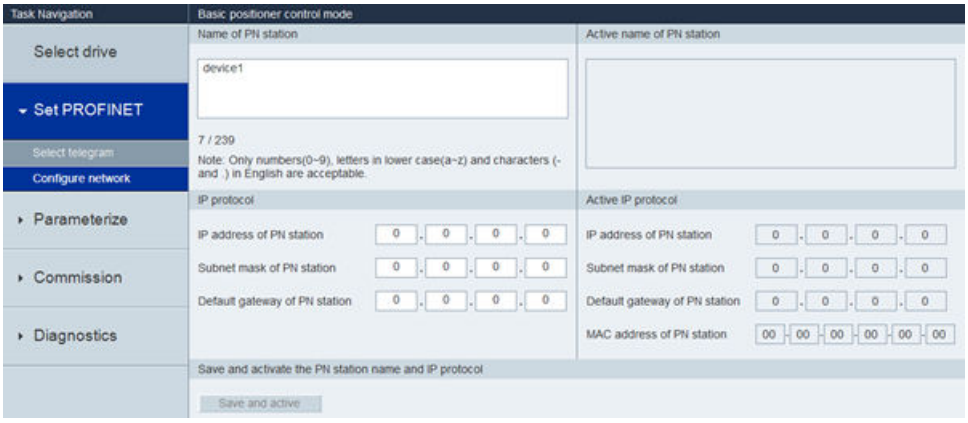
6. In the "Selection of telegram" field, select your desired telegram as the current telegram. The SINA_PARA_S instruction only supports SIEMENS telegram 111 or Standard telegram 1.



Note

When configuring the PROFINET network in the hardware network topology, ensure that the telegrams are consistent with those 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 configuring the PROFINET network in the hardware network topology, ensure that the device name matches the PN station name in this step.

9. Click the "Save and active" button.

Result: The drive restarts and the configured parameters take effect.

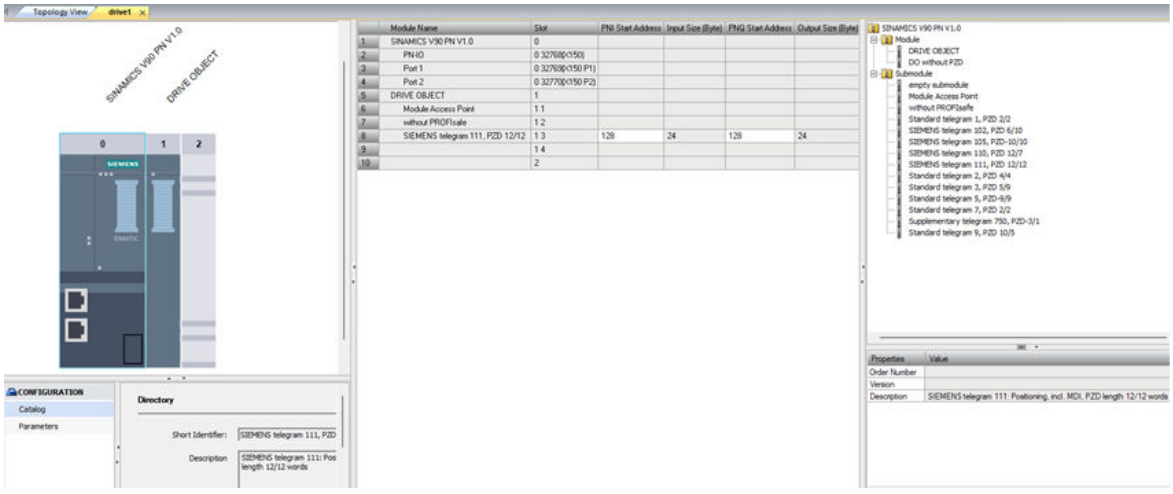


Figure 11-4 Example of PROFINET network configuration in hardware network topology

11.8.4.2 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 11-59 SINA_PARA_S instruction

LAD/FBD	STL	Description
SINA_PARA_S EN Start ReadWrite Parameter — ValueRead1 Index — ValueRead2 ValueWrite1 — Format ValueWrite2 — ErrorNo Device_Number — ErrorId Device_Parameter — PN_Error_Code Status Status_Bit	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 11-60 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_Parame-ter	IN	DWORD	Pointer of the start address of "Device_Parameter". "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 702)
ErrorNo	OUT	WORD	Error number according to PROFIdrive profile (Page 702)
ErrorID	OUT	DWORD	Error ID (Page 703). 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 704).
Status_bit	OUT	BYTE	Status table (Page 704)

Definition of "Device_parameter"

The following table list the bit definitions of Device_parameters:

Table 11-61 Device_parameters

Byte offset	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	Drive type: Select 2 for the V90 drive. Select 1 for the S200 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 information on API number, slot number, and subslot number, go to the Hardware network topology to retrieve the information.

Definition of "Format" parameters

The format of parameter read from the drive is as follows:

Table 11-62 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 11-63 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)

Error ID	Description
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)
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 11-64 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

ID	Description
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 11-65 ErrorID[2]

ID	Description
0×01	Unknown error

Definition of "STATUS" parameter

The parameters of STATUS are as follows:

Table 11-66 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 483).

Definition of "Status_bit" parameters

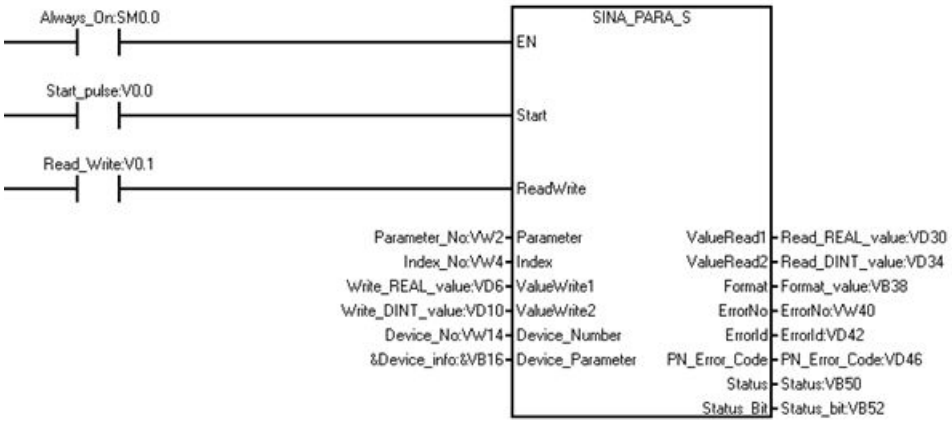
Table 11-67 Status_bit

Byte Offset	Bit 3	Bit 2	Bit 1	Bit 0
0	Error	Done	Busy	Ready

11.8.4.3 Example of the SINA_PARA_S instruction

The SINA_PARA_S instruction can read and write the drive parameters in acyclic communication.

Example


Always_On:SM0.0
Start_pulse:V0.0
Read_Write:V0.1
SINA_PARA_S
EN
Start
ReadWrite
Parameter_No:VW2
Index_No:VW4
Write_REAL_value:VD6
Write_DINT_value:VD10
Device_No:VW14
&Device_info:&VB16
Parameter
Index
ValueWrite1
ValueWrite2
Device_Number
Device_Parameter
ValueRead1
ValueRead2
Format
ErrorNo
ErrorId
PN_Error_Code
Status
Status_Bit
Read_REAL_value:VD30
Read_DINT_value:VD34
Format_value:VB38
ErrorNo:VW40
ErrorId:VD42
PN_Error_Code:VD46
Status:VB50
Status_bit:VB52

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	Values		
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	Drive type	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.

11.9 User-defined library programming tips

Introduction

The creation of the instruction library starts with a STEP 7-Micro/WIN SMART V3 project. You can choose to use any existing project that contains subroutines (SBRs) or function blocks (FBs) to generate the instruction library, but to simplify management, we recommend creating a new project dedicated to the library source code. If you need to build an instruction library using existing subroutines or function blocks scattered across different projects, you should first consolidate all related components into a unified project using the import/export function or by simply copying and pasting. After all the required subroutines and function blocks have been assembled, add the code that calls these components to the main program block. This step is intended to validate the subroutines/function blocks that will become instructions in the library and allows for adequate testing and debugging before formally creating the instruction library. Please note that this project is available both as a downloadable S7-200 SMART program and as the source for an instruction library.

You must have Administrator privileges to create a user-defined library. If you start STEP 7-Micro/WIN SMART with a "Run as administrator" command, this provides the sufficient privileges.

Data type and storage area usage rules

When creating an instruction library, only variables in the V area (variable area) and DB2 area (user variable area) are allowed to use complex data types (such as structures, arrays, etc.). Other user address areas only support the use of basic data types.

Note

When creating a library, be aware of the following limitations. The library creation process will not complete successfully if your POU contains any of the following:

- The POU contains global variables automatically generated by motion control
- The POU references a UDT or array type variable bound to the I, Q, M, or S area
- Absolute address programming in V area is used in POU

Function block instances in the library can access all basic type variables in the user global variable table. In addition, they can access complex data type variables defined in the V area and user variable area (DB2).

Because the library POU (Program Organization Unit) can only recognize variables with the library's own variable table as the scope, motion control-related variables (such as axes, axis groups, CamTable objects, etc.) are automatically generated by the motion control configuration and cannot be directly added to the library's variable table. Therefore, these motion control objects must be passed via IN_OUT parameters of the subroutine or function block. This approach ensures that the library can operate motion control objects safely and flexibly, avoiding conflicts that may be caused by directly referencing global variables.

Global memory usage

To eliminate the possibility of conflicting memory usage between the library and your project, avoid using global memory in your user-defined library. Here are some ways to minimize global memory usage:

- It is recommended to use subroutine/function block parameters instead of global memory whenever possible
  Minimize your library's dependence of global memory by using subroutine/function block parameters. You can specify the memory that your library instruction uses.
  For example, you could have a subroutine that calculates a value, and stores this output in a V memory location. The rest of your program then reads this V memory location to determine the results of the calculation. If you wish to put this subroutine in a library, consider adding an output parameter to the subroutine, and storing the results of the calculation in this parameter. This eliminates the need for a V memory location, and allows you to determine where to store the result.
- Use temp variables for calculations
  Minimize your library's usage of global memory by using temporary variables for calculations and temporary results. The temporary variables defined in your subroutine/function block variable table reside in local memory and will not conflict with the project.
  Local memory is not persistent, and does not retain data between instruction calls. Local memory only persists during the execution of the subroutine/function block that declares it.

- When you need V Memory
  When you must use global V memory in a subroutine/function block in an instruction library, declare variables for all V memories and use these variables in the program. When you compile the library, the compiler uses these variables to determine how much memory your library requires.
  As a general rule, group the V memory used by the subroutines/function blocks that you will include in your library in contiguous locations. For instance, assume that a subroutine uses the following variables:

Name	**Address**
Output1	VB0
Output2	VB10
Output3	VB20

  The compiler interprets this to indicate that the library requires 21 bytes of V memory. However, if we reassign the variables as follows, the variables use consecutive bytes of V memory:

Name	**Address**
Output1	VB0
Output2	VB1
Output3	VB2

  The compiler interprets this to indicate that the library requires only 3 bytes of V memory.

  Note

  Be cautious when reassigning variableic addresses to consolidate memory use. Sometimes a subroutine/function block has an inherent relation between addresses that you must preserve. For example, one variable might represent the start of a table while another variable represents the third element of that table. Only consolidate addresses for unrelated variables.

- When you need I/O addresses
  Sometimes you must use physical I/O addresses in the subroutines/function blocks that you plan to include in your user-defined library. When necessary, declare variables for all I/O addresses use the variables in your subroutine/function block.

- Memory initialization
  Instruction libraries do not include a data block. Therefore, if your subroutine/function block requires V memory with specific initial values, the subroutine/function block should set these values.
- When your subroutine/function block uses accumulators
  Using accumulators for calculations can often reduce the amount of code that a given solution needs since accumulators can be addressed as either byte, word, or dword addresses. Accumulators, however, are a global resource, and if you use them in a subroutine/function block that you intend to include in a library, you must take care that the subroutine/function block does not destroy the contents of the accumulators. Follow these steps to use accumulators safely in a library subroutine:
  - At the beginning of the subroutine/function block, use move instructions to copy the current contents of the accumulators into local memory. This preserves the contents so that the subroutine/function block can restore them later.
  - Your subroutine/function block is now free to use the accumulators and assign values as necessary.
  - At the end of the subroutine/function block, use move instructions to restore the saved values in local memory back into the accumulators.

Instruction and parameter naming

When designing subroutines/function blocks intended to become library instructions, exercise extra caution when naming the subroutines/function blocks and their parameters.

Consider the following when naming the subroutines/function blocks that become instructions in your library:

- Choose names that accurately describe the purpose of the instruction.
  Remember that the person using your library might not have detailed knowledge about each instruction. Choose names that clearly describe the operations of the instructions.
- Avoid generic names.
  By avoiding generic names, you minimize the possibility that the name is a duplicate of a subroutine/function block that the user already has. The user cannot place a call to your instruction if a POU with the same name already exists in their project.
- Be aware of the visual space limitations for LAD and FBD box titles.
  In the LAD and FBD editors, STEP 7-Micro/WIN SMART V3 imposes a limitation as to how many characters it displays for an instruction name as the title for box instructions. Exactly how many characters are visible depends on the font size and grid selection that the user has selected.
- Think about localization.
  If you intend to distribute your instruction library internationally, consider instruction names that are easily identifiable in different languages. Avoid using colloquialisms, slang, or words with differing interpretations.

Many of these same naming guidelines also apply to instruction parameters. You define the names of your instruction parameters in the variable table for the subroutine/function block. Keep in mind that the names for input and output parameters might be the only documentation that a user has to determine how to use an instruction. Define names that help the user determine what each parameter represents.

Signature conflicts and their handling

When importing an external instruction library into an existing project, the system performs an integrity check to identify and resolve signature conflicts. A signature conflict occurs when the name and internal structure of an SBR, FB, or UDT in the library completely match an existing element of the same type in the project. In this case, the system will notify you of the conflict and suggest assigning a new namespace to the imported library element. Once you successfully assign a namespace and complete the initial import, the system will automatically apply the previously specified namespace when importing other elements from the same library in subsequent operations.

Using subroutine/function block comments for program documentation

If you choose to password-protect your library instructions, the users of your library cannot look at the source code in your instructions. They can, however, see the comment for each subroutine/function block in your library.

The subroutine/function block comment is a good place to document the functionality of each instruction. Consider using the subroutine/function block comment to describe the following information:

- What, specifically, does the instruction do?
- How should a program call the instruction?
  For example, if only the first scan should call the instruction, or if the instruction is dependent on the successful completion of another instruction, use the subroutine/function block comment to document these details of instruction usage.
- What global memory is used by the instruction?
  If an instruction directly references global memory, the subroutine/function block comment can warn the user of this usage, so they can design their program to avoid conflicting memory access.
- How does the instruction indicate an error?
  Many instructions might indicate an error condition either through an output parameter or in a global memory location. Use the subroutine/function block comment to tell the user about possible errors, and what to do about them.

Creating a library

Click the "Create library" button from the Libraries section of the File menu ribbon strip to create the instruction library from the subroutines/function blocks in your project.

Debugging library instructions

After you create an instruction library, create some test projects that use the new library to test the functionality of the library instructions before distributing your library to other users. You might discover dependencies between the instructions and the original source project that you did not realize. When you test your instruction library, try to avoid any assumptions about how someone will use the library. Remember that the person using your library does not have the intimate knowledge about its operation that you do.

Using the library in your project

Once you have created the user-defined library, other projects can use it. You must close the project from which you created the library and open a new or another project in order to use your library. When you open a new project, the Libraries folder in the project tree shows the libraries that are in the folder that you configured in the project options for user-defined libraries.

You can also open the folder that you configured in the project options for user-defined libraries and add libraries that you have obtained from other sources. You can also remove libraries from the folder. All of the libraries in the folder are available for new projects. You can select the "Refresh libraries" command from the Libraries node of the project tree to update the set of libraries available for your project. Note that "Refresh libraries" does not include libraries that you created from the currently-open project.

See the "Library Manager" topic for allocating memory for a user-defined library that requires V memory.

Calling library instructions

Once you have included a library to your project, you add the instruction and edit the parameters just as you would any other instruction. You provide the instruction's parameters, and the logic to control its execution, just as you would for a subroutine/function block call or a box instruction.

Distributing your library instructions

When you create a library, the output is a single file (the .smartlibV3 file) that contains everything STEP 7-Micro/WIN SMART V3 needs to recognize and use the instruction library. If you wish to distribute your instruction library to other STEP 7-Micro/WIN SMART V3 users, you can give them a copy of this file.

Conversely, to use an instruction library that someone has given you, copy the .smartlibV3 file to the user-defined libraries folder on your computer, and include the library in your project.

Making changes to subroutines/function blocks included in your library

When you wish to make changes to the code in your library, open the project that serves as the source for the library, and edit the POUs included in the library. When you create the library, the .smartlibV3 file includes the new changes.

When you change your library, update the version and the name (if necessary) from the "Create library" dialog.

Note

If you do not change the location where STEP 7-Micro/WIN SMART V3 stores the library file or the name of the library file, then STEP 7-Micro/WIN SMART V3 writes the new version of the library over the existing version, essentially destroying the existing version. When you change the version of a library, change the name of the library file, or change the folder where the library is stored.

Using the version check feature

When you distribute an updated version of your library, and your users include this new version for use, existing projects do not "automatically" update to the new version.

To aid users in converting between different versions of an instruction library, STEP 7-Micro/WIN SMART V3 provides a version check feature. Right-click a library and select "Check version" from the context menu to perform a version check.

The version check feature looks at each of the library instructions that the current project uses. If the included library has a different version of the instruction, then one of the following two actions will be taken:

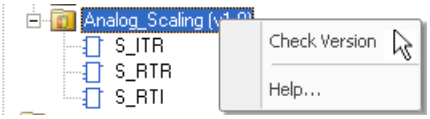
- If the signatures of both versions of the instruction are identical, STEP 7-Micro/WIN V3 will update the subroutines/function blocks that make up the current instruction using the new version of the library. When this is the case, STEP 7-Micro/WIN SMART V3 updates the project without any manual effort on the part of the user.
- If the signatures for the two versions of the instruction do not match, then no automatic update is possible. In this case, STEP 7-Micro/WIN SMART V3 will remove the subroutine/function block from the project. When this occurs, the user must replace the instruction call with a call to the new version of the instruction. This will add the correct version of the subroutine/function block to the project.

If the new version requires a different memory allocation, STEP 7-Micro/WIN SMART V3 allocates the required memory from the existing starting V memory address. You can always change the library memory allocation if you need to.

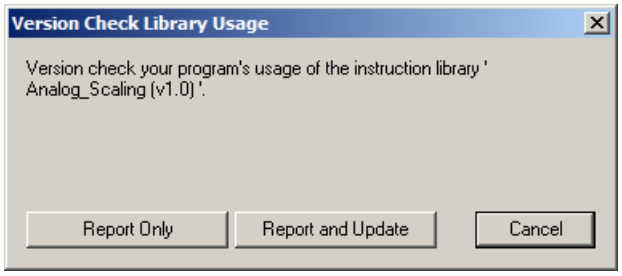
Updating a project to use the latest version of a library

To update an existing project to use the latest version of the library:

1. After adding the new version of the library, right-click the library name in the "Libraries" folder in the "Instructions" folder of the project tree.
2. From the context menu, click "Check version".



The "Check Version" context menu command is only available if the current project uses the library.
Selecting this option loads the following dialog:



3. From this dialog, you have the option of executing a version update on the current project or simply scanning the current project to determine what actions need to be taken.
   - **Report only**: The "Report only" option scans your program to determine what changes you need to make if you decide to update your project to the latest version of the library. STEP 7-Micro/WIN SMART V3 displays the report in the output window and makes no changes to your project.
   - **Report and update**: The "Report and update" option scans your program to determine what changes need to be made, and executes those changes. Before executing an update, you first select "Report only", and carefully review the changes that STEP 7-Micro/WIN SMART V3 will make to your project. Also save a backup copy of your project before executing an update.

11.10 How to reload libraries

To reload libraries from the configured library folder to the project, right-click the Libraries node of the project tree and select "Refresh Libraries" from the shortcut menu.

STEP 7-Micro/WIN SMART V3 looks for and loads any library files (.smartlibV3 files) from the library folder and its subfolders. You can then see them listed under the Libraries node of the project tree.

Note

If you create libraries from the currently-open project, STEP 7-Micro/WIN SMART V3 does not include them in the Libraries node of the project tree. The Refresh Libraries command loads libraries from the library folder except for any that correspond to the current project instance.

See also

Application user reference

Getting started

User-defined library example (Page 722)

11.11 User-defined library example

11.11.1 Example library for scaling analog values

You can create a library to save program logic that you might want to use to perform a common function in multiple projects. This example shows the use of library to store subroutines that scale analog values.

Library subroutines for scaling analog values

Before you create a library, you create a project with subroutines that perform the common functions that you eventually want to include in a library. This example uses three subroutines that perform analog value scaling routines. Each subroutine, in turn, implements a different type of scaling:

- Scale_I_to_R (S_ITR) (Page 716): This subroutine scales an analog value from an integer range to a floating point range. The initial (integer) value is an input to the subroutine. The scaled floating point value is an output from the subroutine. The integer high and low range of the input, and the floating-point high and low range for the output are all inputs to the subroutine.
- Scale_R_to_I (S_RTI) (Page 718): This subroutine scales an analog value from a floating point range to an integer range. The initial (floating point) value is an input to the subroutine. The scaled integer value is an output from the subroutine. The floating-point high and low range of the input, and the integer high and low range for the output are all inputs to the subroutine.
- Scale_R_to_R (S_RTI) (Page 721): This subroutine scales an analog value from one floating point range to another floating point range. The initial value (conforming to the input floating point range) is an input to the subroutine. The output of the subroutine is the analog value scaled to the output range. The floating-point high and low range of the input, and the floating-point high and low range for the output are all inputs to the subroutine.

11.11.2 Subroutine S_ITR: integer to real

Scale integer to real

The purpose of this subroutine is to scale an integer number from the input range to a real number in the output range.

This subroutine first reads the values passed from the calling POU: the input to be scaled, the input scale high limit (ISH), the input scale lower limit (ISL), the output scale high limit (OSH), and the output scale low limit (OSL).

The next step is to subtract OSL from OSH to determine the output scale range, stored back in OSH. Next, the subroutine subtracts ISL from the input and converts it first to a double word and then to a real number. The next instruction then multiplies the real number by the output range.

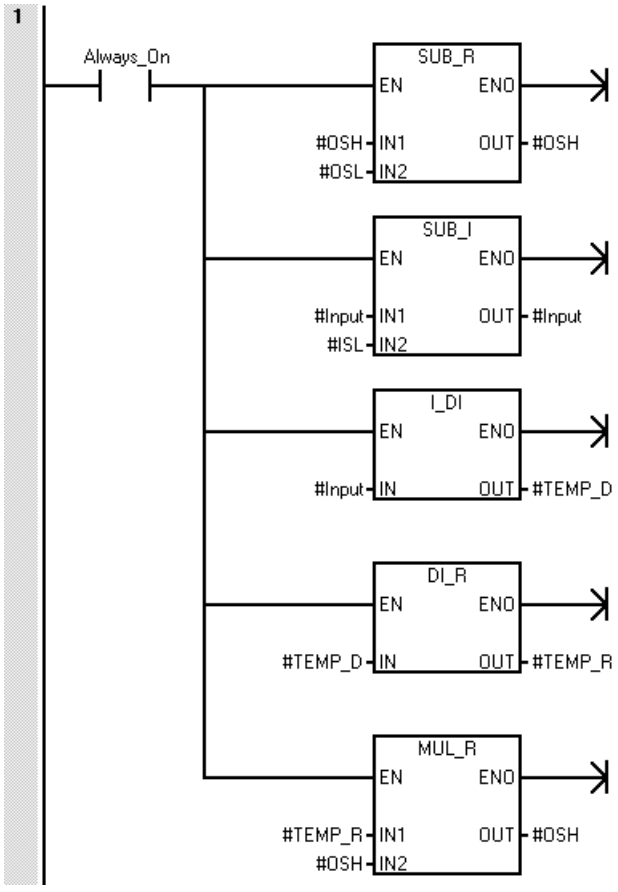
The next step subtracts ISL from ISH to calculate the input scale range, converts this calculated range to a double word, and then to a real number.

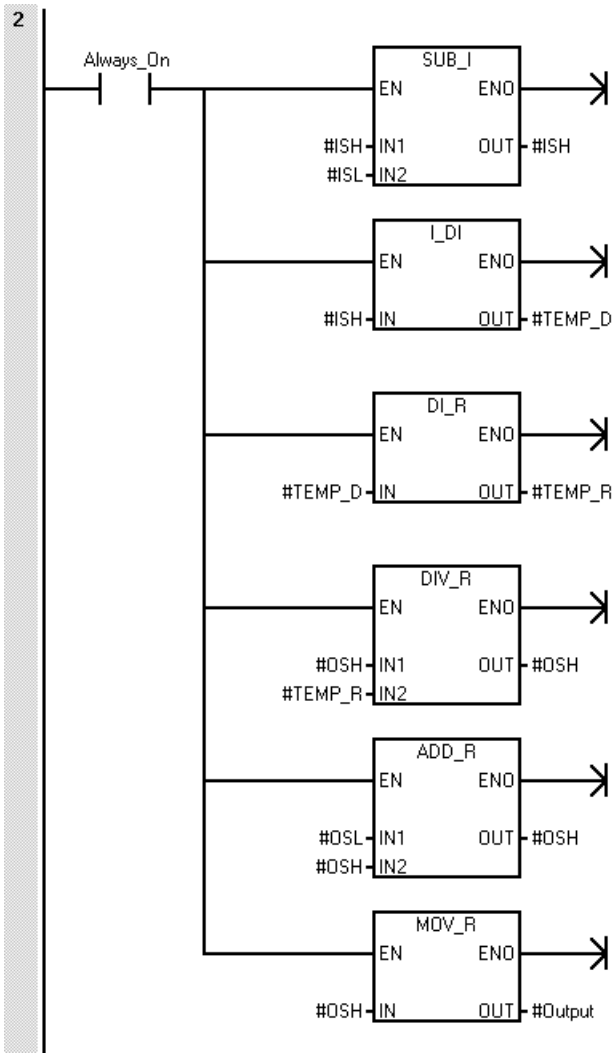
Next, the subroutine divides the output scale range by the calculated real number and adds the result back to the output scale low value (OSL) to determine the final output value. Finally, the subroutine moves this value to the output of the subroutine to be used by the calling POU.

The formula is as follows:

Output = [(OSH - OSL) * (Input - ISL) / (ISH - ISL)] + OSL

Variable	Variable type	Data type	Definition
Output	OUT	REAL	scaled output value
Input	IN	INT	analog input value
OSH	IN	REAL	high limit of the scale for the scaled output value
OSL	IN	REAL	low limit of the scale for the scaled output value
ISH	IN	INT	high limit of the scale for the analog input value
ISL	IN	INT	low limit of the scale for the analog input value


1
Always_On
SUB_R
EN
ENO
#OSH
IN1
OUT
#OSH
#OSL
IN2
SUB_I
EN
ENO
#Input
IN1
OUT
#Input
#ISL
IN2
I_DI
EN
ENO
#Input
IN
OUT
#TEMP_D
DI_R
EN
ENO
#TEMP_D
IN
OUT
#TEMP_R
MUL_R
EN
ENO
#TEMP_R
IN1
OUT
#OSH
#OSH
IN2



11.11.3 Subroutine S_RTI: real to integer

Scale real to integer

The purpose of this subroutine is to scale a real number from the input range to an integer in the output range.

This subroutine first reads the values passed from the calling POU: the input to be scaled, the input scale high limit (ISH), the input scale lower limit (ISL), the output scale high limit (OSH), and the output scale low limit (OSL).

The next step is to subtract OSL from OSH to determine the output scale range, stored back in OSH. Next, the subroutine subtracts ISL from the input and converts it first to a double word and then to a real number.

The next step subtracts ISL from the input and then multiplies the result by calculated real number output scale range.

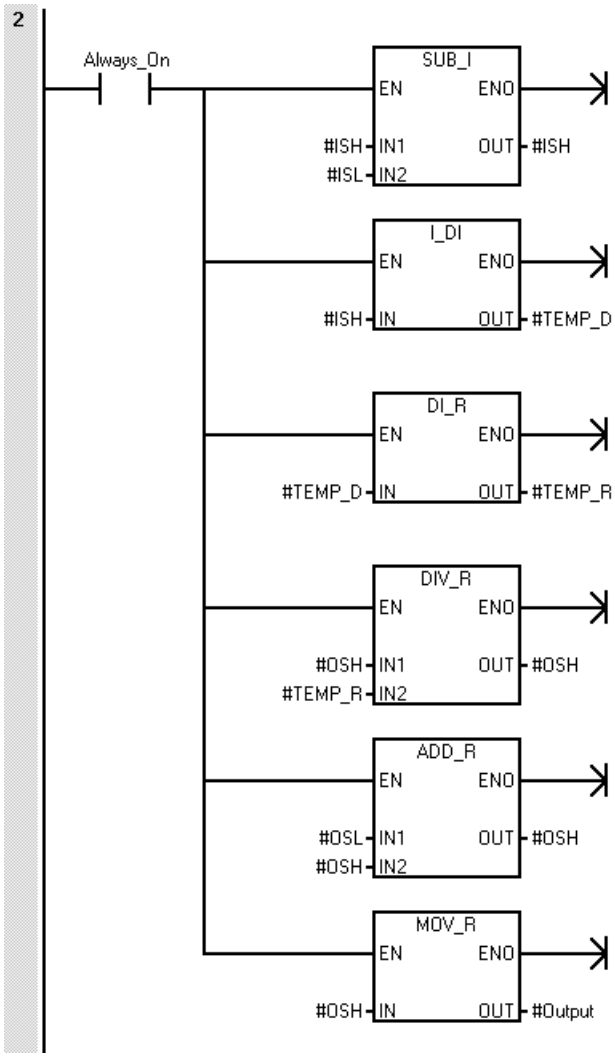
The next step subtracts ISL from ISH to calculate the input scale range, divides the output range by the calculated input scale range, and rounds the result into a double word.

The subroutine then checks the rounded output value against the boundary constraints for that data type to ensure accuracy of the final value. After the boundary check, the subroutine converts the double word value to an integer and adds it to the output scale lower limit (OSL), and moves the value to the subroutine output.

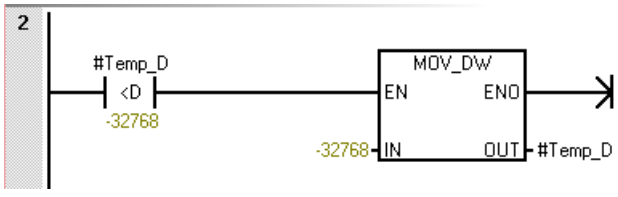
The formula is as follows:

Output = [(OSH - OSL) * (Input - ISL) / (ISL - ISL)] + OSL

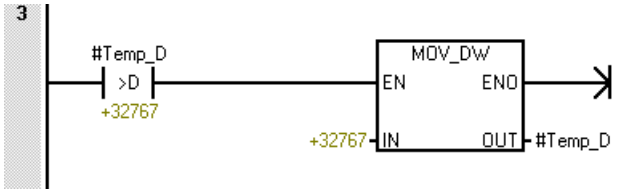
Variable	Variable type	Data type	Definition
Output	OUT	INT	scaled output value
Input	IN	REAL	analog input value
OSH	IN	INT	high limit of the scale for the scaled output value
OSL	IN	INT	low limit of the scale for the scaled output value
ISH	IN	REAL	high limit of the scale for the analog input value
ISL	IN	REAL	low limit of the scale for the analog input value



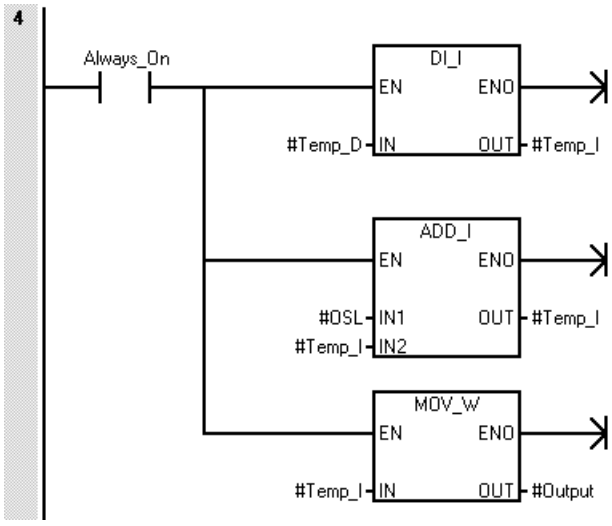
Network 2 checks whether Temp_D < -32768



Network 3 checks whether Temp_D > 32767



Network 4 converts to an integer value and adds the low limit offset



11.11.4 Subroutine S_RTR: real to real

Scale real to real

The purpose of this subroutine is to scale the normalized analog input from the input range to the output range.

This subroutine first reads the values passed from the calling POU: the input to be scaled, the input scale high limit (ISH), the input scale lower limit (ISL), the output scale high limit (OSH), and the output scale low limit (OSL). All inputs are real numbers.

The next step is to subtract OSL from OSH to determine the output scale range. The subroutine then subtracts ISL from the input and multiplies the result by the output range.

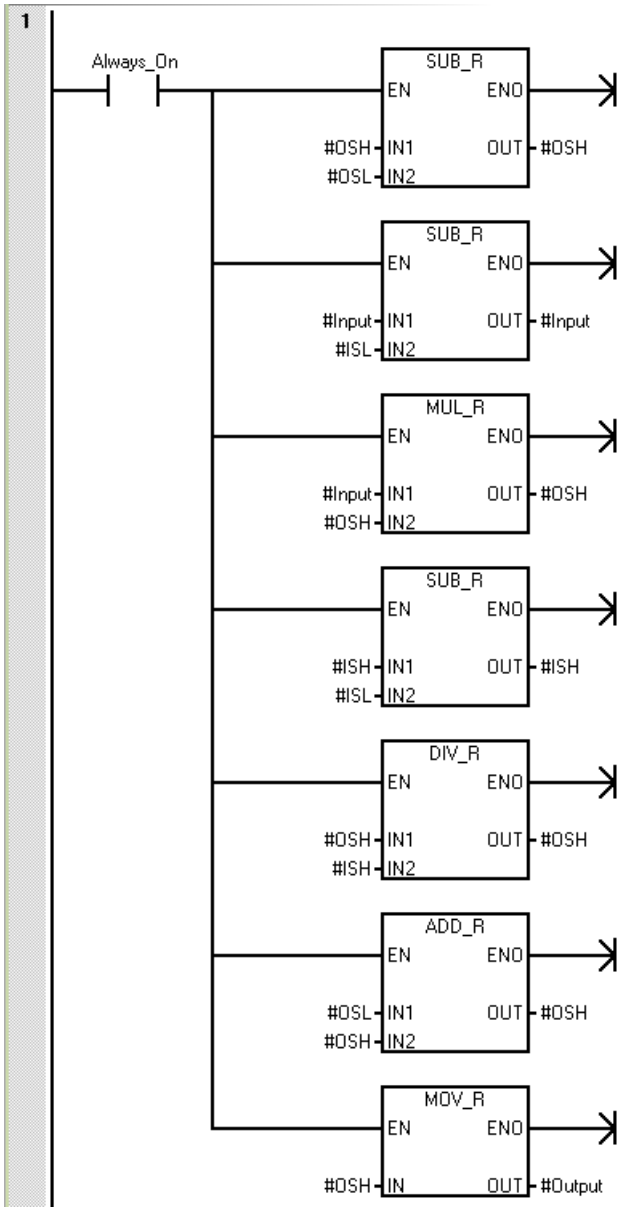
The next step is to subtract ISL from ISH to determine the input scale range.

At this point, the subroutine divides the output range by the de-scaled input to determine the final output value, and moves this value to the subroutine output to be used by the calling POU.

The formula is as follows:

Output = [(OSH - OSH) * (Input - ISH) / (ISH - ISL)] + OSL

Variable	Variable type	Data type	Definition
Output	OUT	REAL	scaled output value
Input	IN	REAL	analog input value
OSH	IN	REAL	high limit of the scale for the scaled output value
OSL	IN	REAL	low limit of the scale for the scaled output value
ISH	IN	REAL	high limit of the scale for the analog input value
ISL	IN	REAL	low limit of the scale for the analog input value



11.11.5 Creating the user-defined library

From the example analog scaling program, you can now create a library that contains the three scaling subroutines S_ITR (Page 716), S_RTI (Page 718), and S_RTR (Page 721).

Creating an instruction library from your project

To create a library from your analog scaling subroutines, follow these steps:

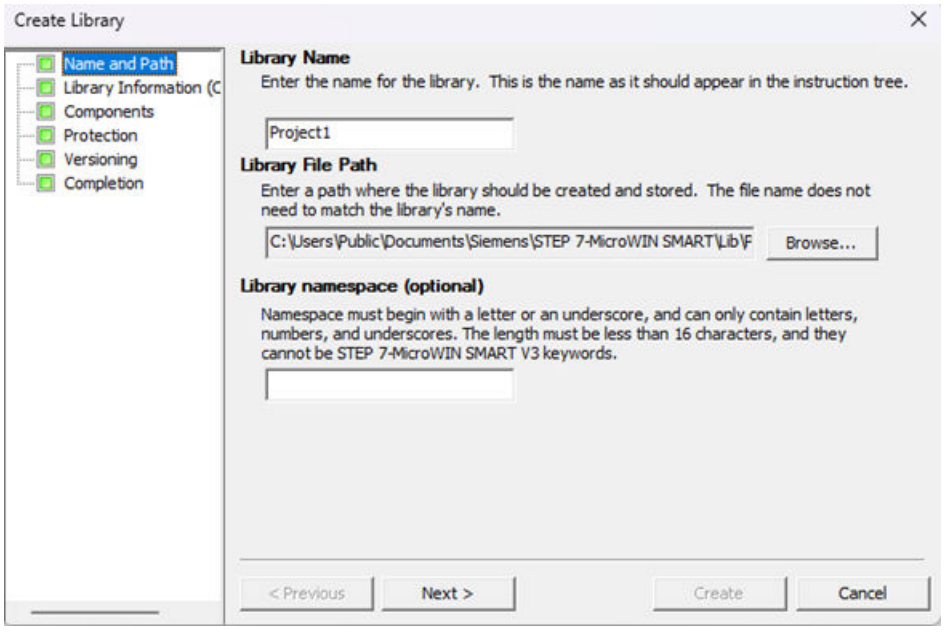
1. From the "Libraries" area of the "File" menu ribbon strip, click the "Create" button 

.
2. Under the "Name and path" node, enter the following information.
   - Library name: Enter a name for the library. For this example, provide the name "Analog_Scaling" for your library.
     The library name can contain spaces and mixed case letters. The name you supply here is the name you will see in the project tree once STEP 7-Micro/WIN SMART V3 adds the library to the project.
   - File path: By default, STEP 7-Micro/WIN SMART V3 uses the default library folder configured in your project options. You can, however, use the "Browse" button to navigate to and select a folder. If you want this library accessible to other projects, however, use the folder that you configured in the project options.
   - Library namespace: Library namespaces are used to resolve signature conflicts between program organization units (POUs) and user-defined data types (UDTs) that may occur when you import multiple libraries into a project. By specifying a unique namespace for a library, you ensure that the names of elements within it are globally unique. The namespace feature is only available for Program Organization Units (POUs) and User Defined Data Types (UDTs) in libraries. The namespace supports only a single-level structure and does not support nested sub-namespaces.
     If a namespace is set for a library, POUs and UDTs imported into the library will be displayed in the form of "namespace.POU/UDT". You must use the full path when accessing these elements in user code.

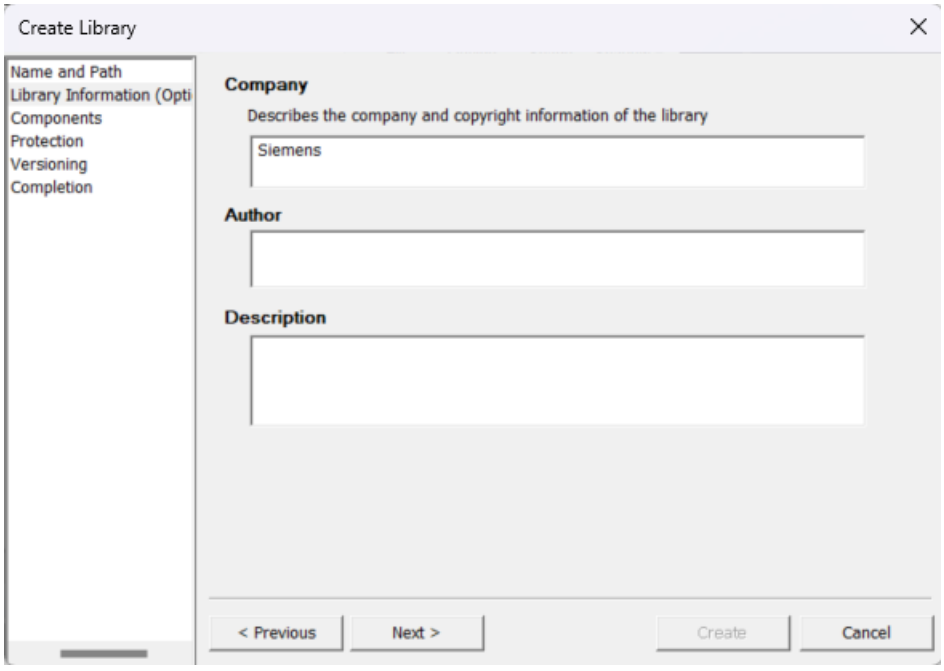
Note

Namespace names must follow the following rules:

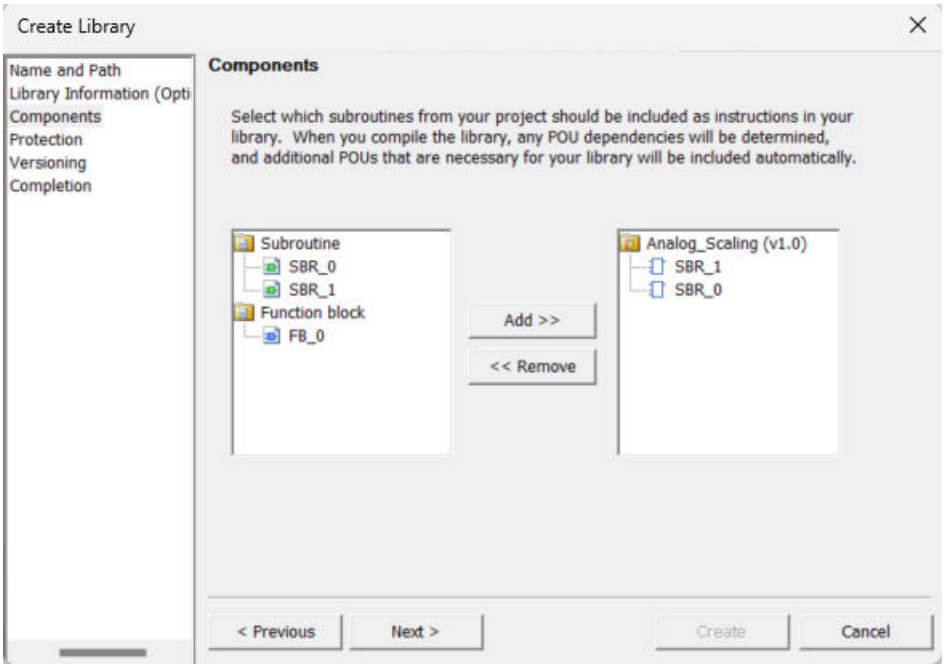
- Can only consist of letters (a-z, A-Z), numbers (0-9), and underscores (_)
- Must start with a letter or underscore
- Maximum length is 15 characters
- Chinese characters are not supported



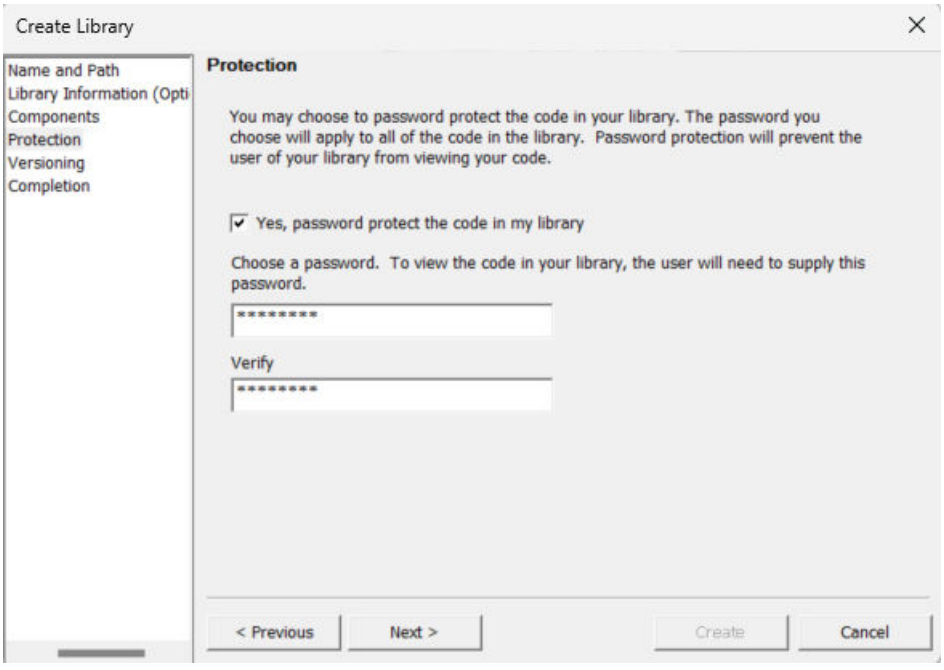
3. In the "Library information" node, enter the library details.



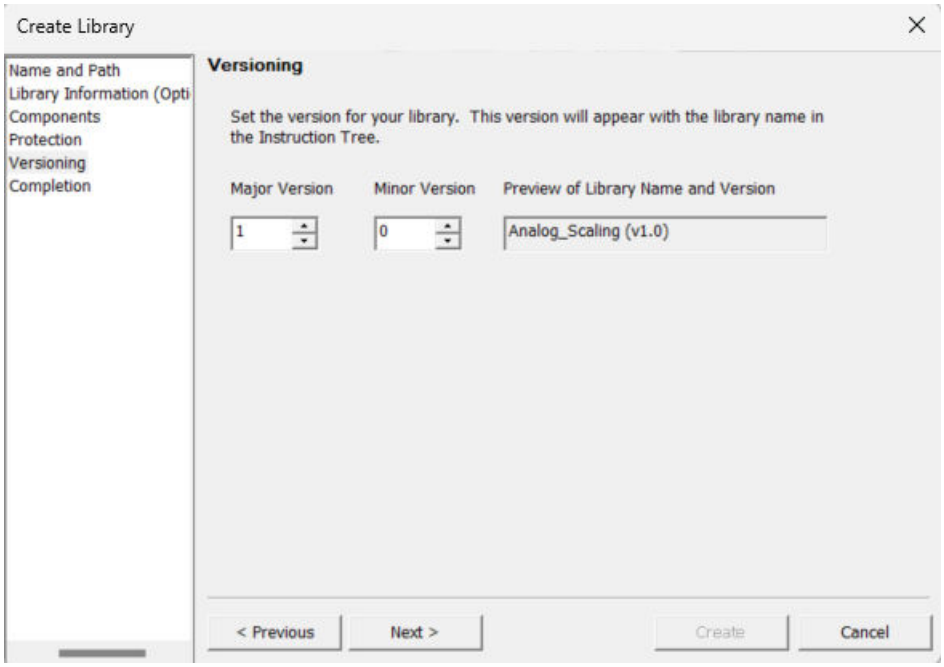
4. From the "Components" node, select each of the three scaling subroutines and click the Add button for each one.



5. Use the "Protection" tab if you want to protect your library with a password. Select the check box, enter and verify a password to require users to enter a password to view or modify the code in your library. Without password entry, users of your library can only see the presence of the library subroutines, and the variable table for each subroutine.



6. From the "Versioning" node, you can provide a version number, which can be useful if you have different versions of the same library.



7. Click "Create" from the "Completion" node when you finish. STEP 7-Micro/WIN SMART V3 compiles your library subroutines and displays any errors in the output window. Correct errors if present and repeat the process to create the library. When the library compiles successfully, STEP 7-Micro/WIN SMART creates the library and saves it to the location that you designated.

Note

Dependency management

When creating a new library from a project that contains dependent libraries, all referenced dependent library contents are exported in dependency order. When there are dependencies and no conflicts, the contents are merged; if there are conflicts, the entire namespace is replaced.

When a user library is created, the system will package and export the contents of all dependent libraries (including UDT, SBR, FB, and library variable table) by default. When this library is introduced into a new project, its own contents and all dependent library's UDTs, SBRs, FBs and library variable tables will exist in the new project at the same time. If you need to assign V area addresses, you must assign V area addresses for all relevant library memory areas through the library manager.

11.11.6 Using the library in your project

After you have created a library (Page 722), it is available to other projects. New projects display all of the libraries under the Libraries node of the Instructions folder of the project tree. For existing projects, you can refresh the library folder (Page 714) to show the existing libraries in the library folder that you configured in the project options.

Note that the name and version of the library is just as you entered it when you created the library. The subroutines that you selected appear as instructions in the library.

Calling a library subroutine

Once you have included the library, you can place library instructions by using Drag & Drop from the instruction tree, or in LAD and FBD from the Insert Box (F9) list.

To place a call to the S_ITR (scale integer to real) instruction from the Analog_Scaling library, follow these steps:

1. Open the POU in the program editor window and scroll to the network where you wish to insert the call.
2. Expand the Libraries folder In the Instructions folder of the project tree. You can either drag the appropriate library instruction from the tree and drop it on the appropriate cell of the network in the program editor, or place your cursor on the cell in the program editor and then double-click the library instruction in the tree. In LAD and FBD, you can also press F9 or click the Insert Box button  from the program editor.
   When you place a call to a library instruction, STEP 7-Micro/WIN SMART V3 adds one or more POUs to your project. Remember that the library instructions are actually subroutines that the library contains. STEP 7-Micro/WIN SMART V3 adds the POU when you place the first call to a library instruction in your project; when you place subsequent calls, this does not occur, since the required subroutine is already a part of your project.
   You'll notice that a new POU (S_ITR) has been automatically added to your project. It appears in the project tree, and it has a tab in the program editor. The instruction you placed is now a call to this subroutine.
3. Edit the parameters of the call instruction in your program and assign valid operands to each parameter. You can view the variable table for the S_ITR subroutine to find the data types of each of the parameters.

Example usage of the library scaling instructions

Consider an example where a closed container holds a liquid.

The differential pressure transducer supplies a current (0 mA to 20 mA) at the input of the analog input module. The current is proportionate to the level of the liquid in the container.

The analog module has to be calibrated so that, at a level of 10 m, the analog current value of 20 mA is converted to a value of 32000. At a level of 0 m, the analog current value of 0 mA must be

correspond to a value of 0. The program scales the converted digital values into height in meters.

For the liquid level indicator, the voltage has to be generated by an analog output module. This voltage is generated by writing a corresponding integer value at the analog output word (AQW).

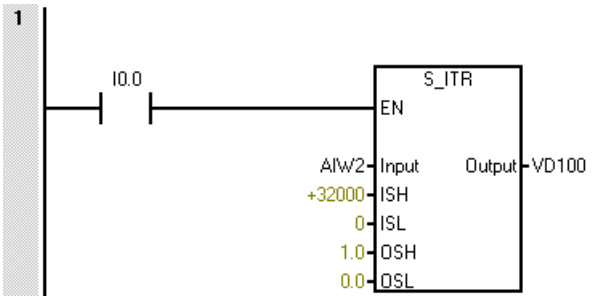
The analog output module feeds the liquid level (between 0 m and 10 m) to a measuring instrument as a voltage from 0 V to 10 V. The analog display of the measuring instrument responds to the voltage with a pointer deflection proportionate to the liquid level.

The formula will scale any given value between any given maximum and minimum scale points. This program, in particular, receives the analog input word (AIW) and scales it to the analog output module. First, the program reads the AIW, which is a value between 0 mA and 20 mA (0 and 32000), and scales it to a normalized value, between 0.0 and 1.0. This value is then scaled to both 0.0 to 100.0 and 0 to 32000.

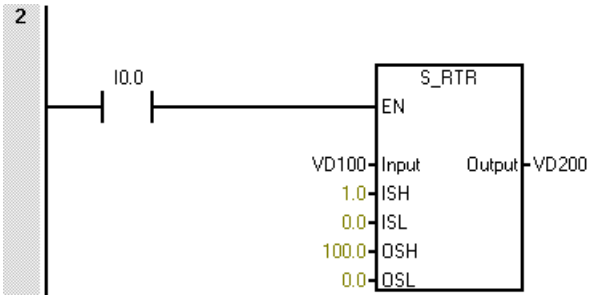
Variable	Definition
Output	scaled output value
Input	analog input value
OSH	high limit of the scale for the scaled output value
OSL	low limit of the scale for the scaled output value
ISH	high limit of the scale for the analog input value
ISL	low limit of the scale for the analog input value

Formula for calculating the scaled value: Output = [(OSH - OSL) * (Input - ISL) / (ISH - ISL)] + OSL

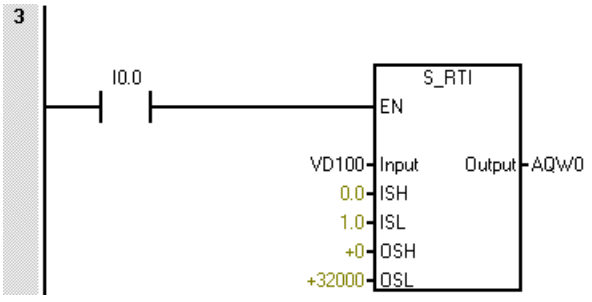
Network 1: Reads the analog input and normalize the range to 0.0 to 1.0.



Network 2: Scales the normalized value to the range 0.0 to 100.0 (as a percent)



Network 3: Scales the normalized value to the range 0 to 32000 and transfers it to the analog output



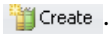
11.11.7 Changing the version of your library

Over time, you might make changes to your original library and need to create new versions.

Note

When changing the library version, ensure the library name and namespace remain consistent with the original version. You may modify the library file name or use a separate folder to store previous versions of the library. If you do not change the name or location of the library file, then STEP 7-Micro/WIN SMART V3 saves the new version of the library over the existing version. Your original library is then no longer available.

To create a new version of an existing library, follow these steps:

1. If you choose, copy your original library from the library folder that you configured in the project options to another location as a backup.
2. Open the original project from which you created the library. This is the project that includes the library subroutine code, not a project that calls a library instruction.
3. Make changes as necessary.
4. From the "Libraries" area of the "File" menu ribbon strip, click the "Create library" button 

.
5. Follow the same procedure as in creating a library (Page 722), but give the library a new version number and possibly a new name.

Debugging and troubleshooting 12

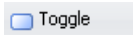
STEP 7-Micro/WIN SMART V3 provides software tools to help you debug and test your program. These functionalities are: Viewing the status of the program as it is executed by the CPU, selecting to run the CPU for a specified number of scans, and forcing values.

Use the hardware troubleshooting guide (Page 742) as a guide for determining the cause and possible solution when troubleshooting problems with the hardware.

12.1 Debugging your program

12.1.1 Bookmark functions

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.



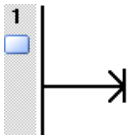
Next Bookmark: Click this button to move to the next bookmarked network of your program.



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.



12.1.2 Cross reference table

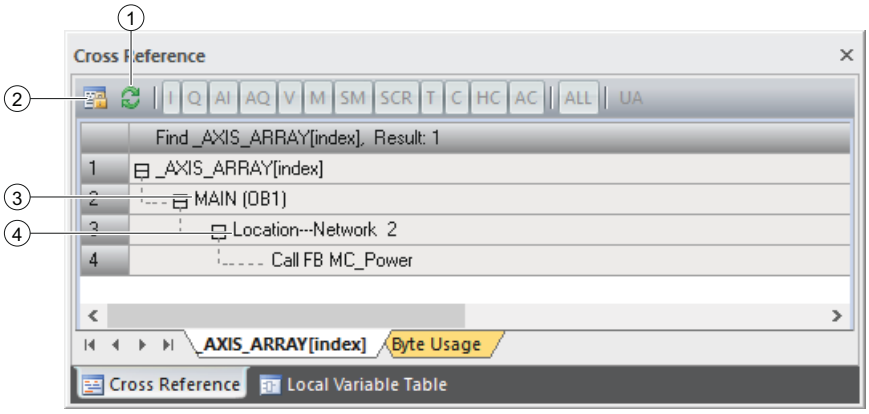
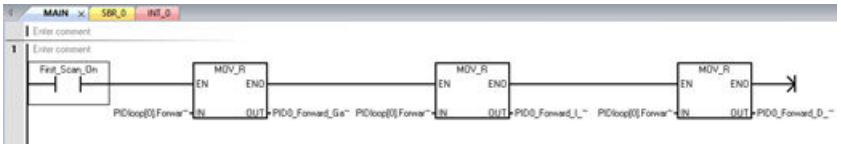
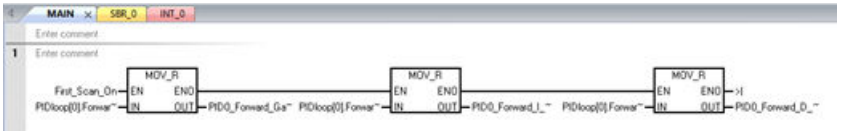
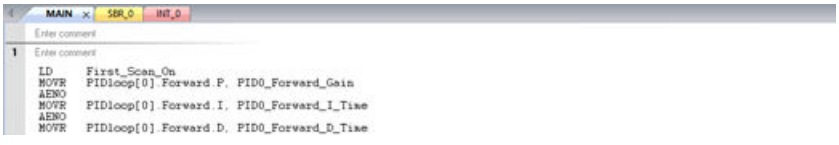
Use the cross reference table when you want to know whether a variable name or memory assignment is already in use in your program, and where it is used. The cross reference table identifies all operands used in the program, and identifies the POU (Page 143), network or line location, and instruction 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	

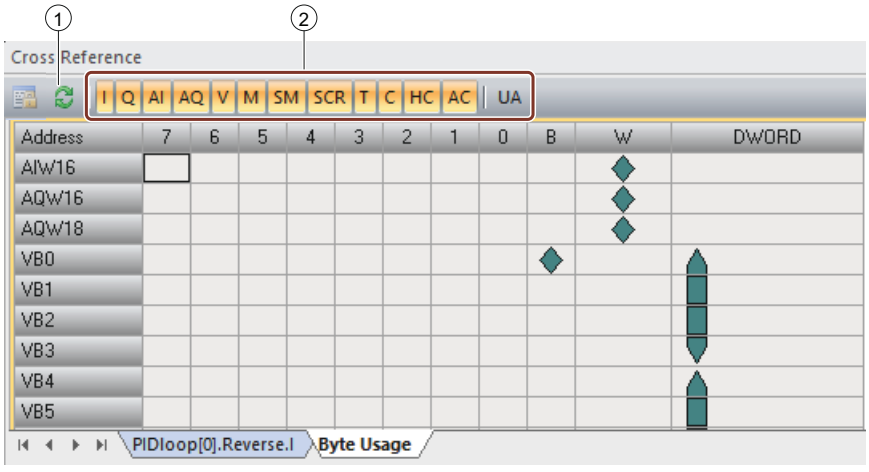
12.1.3 Byte usage table

The byte usage table of the cross reference table (Page 731) allows you to see which bytes, from which memory address, down to the bit level, have been used in your program. It also helps you recognize duplicate-assignment errors.

- **Address** indicates the address in memory.
- **0...7** Indicates that the bit of memory has been assigned.
- **B** Indicates that a byte of memory has been assigned.
- **W** Indicates that a word (16 bits) has been assigned.
- **DWORD** Indicates that a double word (32 bits) has been assigned.

Example 1: interpreting the byte usage table

This example of a byte usage table shows that its associated program makes use of the following memory locations: one byte within VB0; AIW 16; AQW 16; AQW 18; Dword VB0 to VB3; Dword VB4 to VB7.



① Refresh ② Memory type

12.2 Displaying program status

12.2.1 How the program editor displays status

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 737) 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.

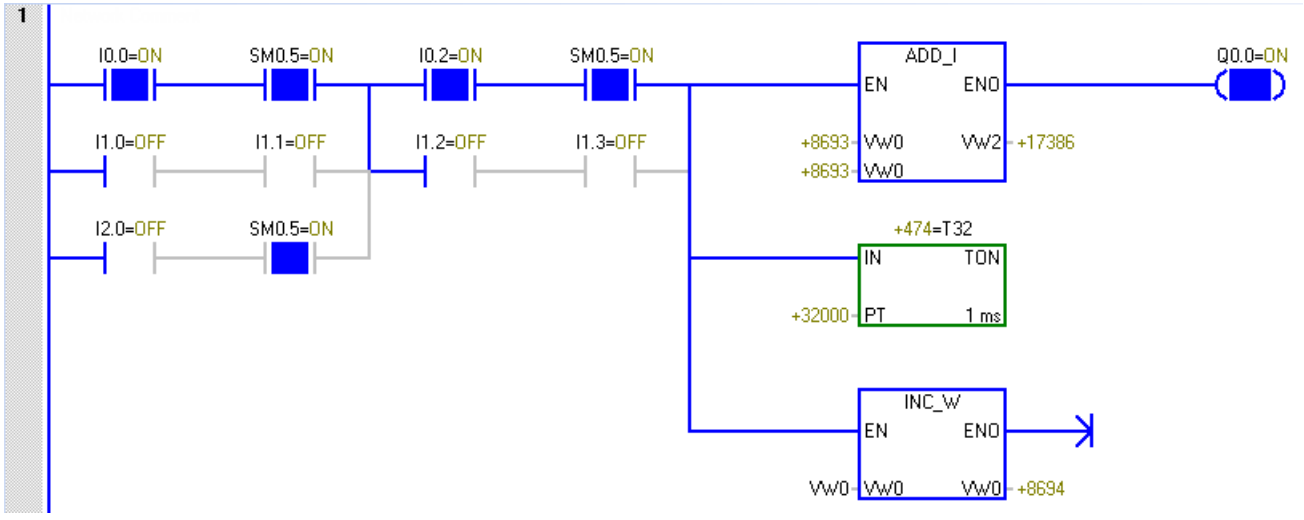
The STL Status tab of the Program Editor options (Page 736) in the Options settings of the Tools menu ribbon strip allows you to select or deselect any of these categories of values. If you deselect an item, it does not appear in the status display.

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

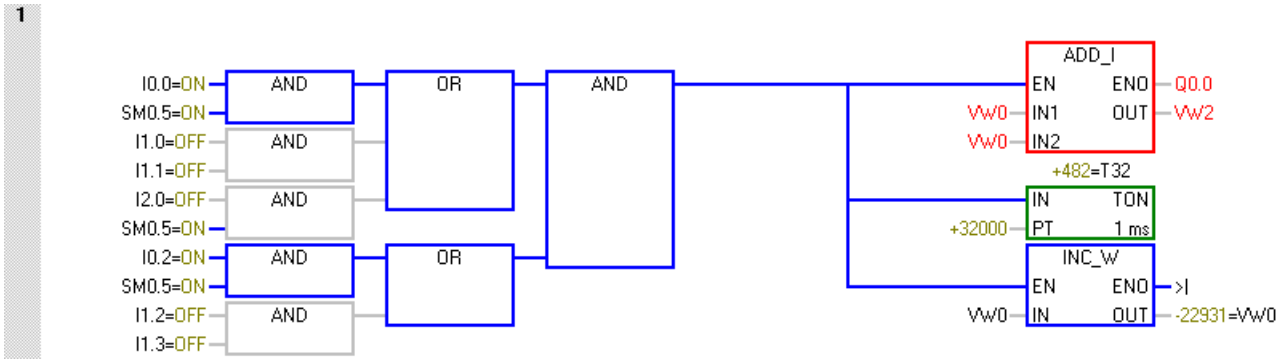
Example program status in the LAD program editor

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.



12.2.2 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 check-box, 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 733)

12.3 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 (Page 733) and Program Status (Page 733) is off and use one of these methods to create a new chart:

- From the project tree (Page 152), 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.

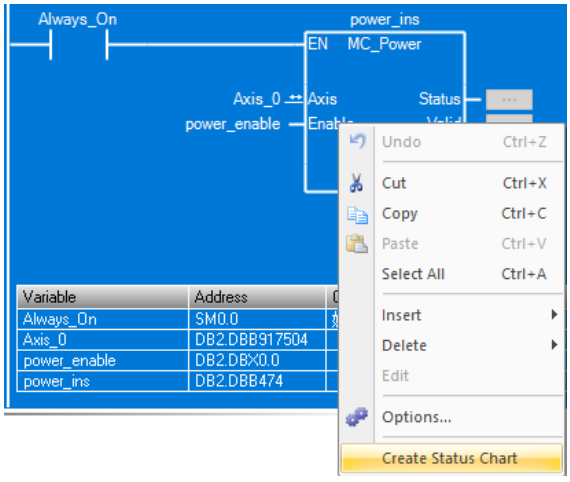
Building a status chart

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.

12.4 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 205). 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 138) indicate whether the CPU has any forced data.

12.5 Writing and forcing outputs in STOP mode

To enable Write and Force functions while in STOP mode (Page 81), 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 138) indicate whether the CPU has any forced data in STOP mode.

12.6 How to execute a limited number of scans

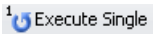
You can specify that the PLC execute your program for a limited number of scans (from 1 scan to 65,535 scans). By selecting the number of scans for the PLC to run, you can monitor the program as it changes the process variables.

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 138) if the PLC is not already in STOP mode.

Executing a single scan

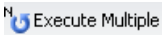
To execute a single scan, click the "Execute Single" button from the Scan area of the Debug menu ribbon strip.



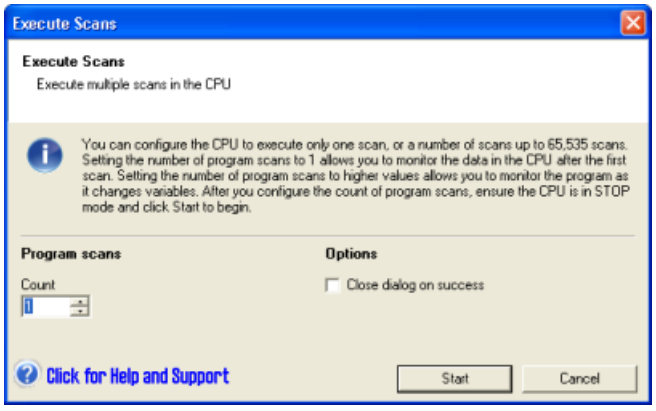
Executing multiple scans

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 138).

See also

Overview of debugging and monitoring features (Page 731)

How to display status in the editor windows (Page 733)

How to display status in a status chart (Page 737)

How to download a program (Page 126)

Timestamp mismatch error (Page 1183) (ensuring that project in programming device matches project in PLC)

Cross reference and element usage (Page 731) (ensuring that program edits do not cause duplicate assignments)

Forcing values (Page 739)

Forcing outputs in STOP mode (Page 740)

12.7 Hardware troubleshooting guide

Table 12-1 Troubleshooting guide for the S7-200 SMART G2 hardware

Problem	Probable cause	Possible solutions
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. Please refer to the wiring guide (Page 97).
	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 Wiring guidelines (Page 97) for information about installing the field wiring.
	Incorrect voltage	
Intermittent operation associated with high energy devices	Improper grounding	Please refer to the wiring guide (Page 97).
	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.
	Too short of a delay time for the input filters	Increase the input filter delay in the system data block.

Problem	Probable cause	Possible solutions
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 Wiring guidelines (Page 97) 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. Please refer to the appendix for the product numbers of S7-200 SMART devices.
Other communications problems (STEP 7-Micro/WIN SMART V3)		For information on network communication, please refer to the appendix "Error codes (Page 1183)" in the programming manual.

PID loops and tuning

13

13.1 Introduction to PID self-tuning function

Using the PID tuning control panel in STEP 7-Micro/WIN SMART, you can conveniently perform self-tuning of PID parameters for the S7-200 SMART PLC.

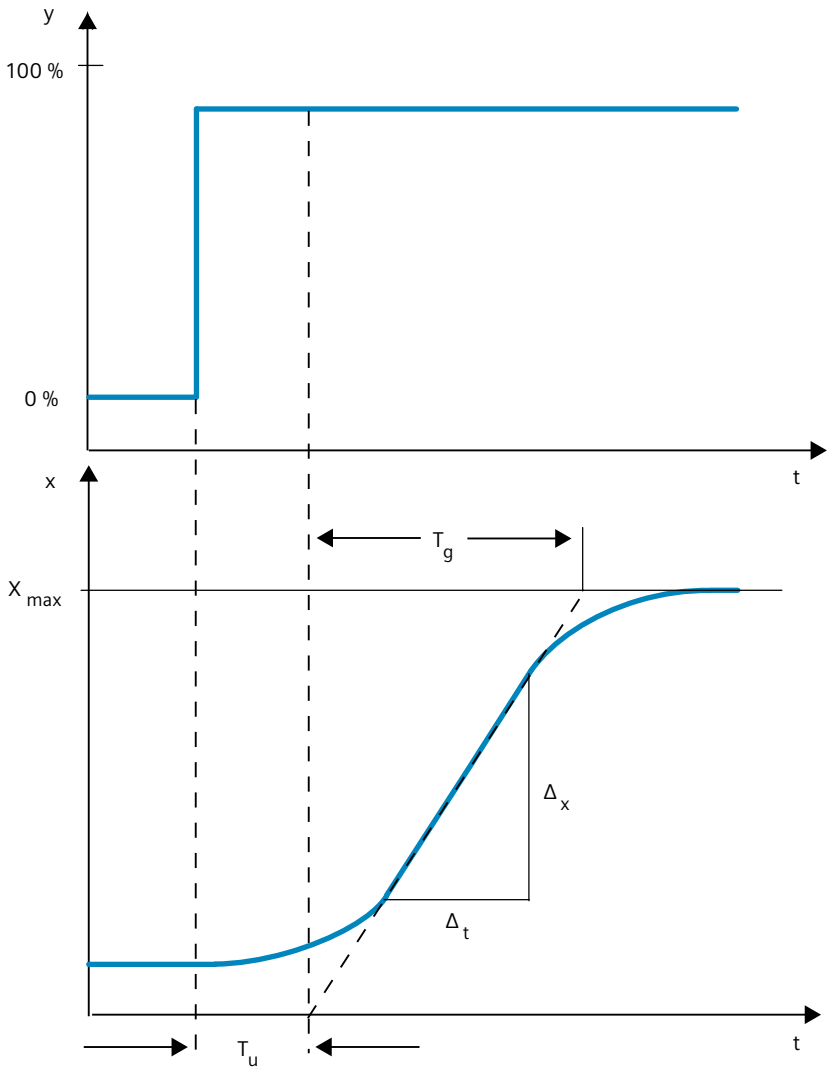
While the CPU is in the online RUN mode, you can trigger the PID self-tuning operation in the program through the control panel. This function will automatically calculate and recommend appropriate values for the gain, integral time, and derivative time parameters. Depending on the specific controlled object and the desired control performance, you can select an appropriate loop response type, including modes such as fast response, medium response, slow response, or very slow response.

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.

This document uses the bidirectional temperature PID control self-tuning process (heating in the forward direction and cooling in the reverse direction) as an example. The PID self-tuning methods for other physical quantities can also be implemented by referring to this document.

13.2 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

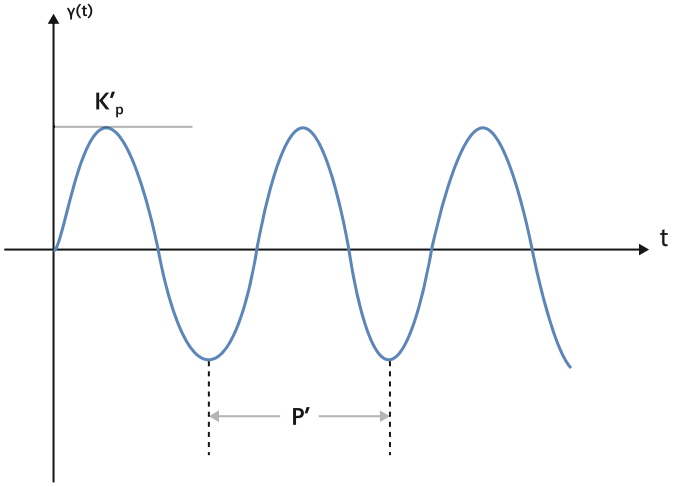
- Forward/heating pretuning successfully performed
- | setpoint - process value | < 0.05 × |upper process value - lower process value |

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

13.3 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.

13.4 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 references this loop table that contains the loop parameters.

- Unidirection: 120 bytes
- Bidirection: 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 13-1 Loop table for unidirectional PID

Shift by	Field	Format	Type	Description
0	Process variable (PV_n)	REAL	Inputs	Contains the process variable, which must be scaled between 0.0 and 1.0.
4	Setpoint (SP_n)	REAL	Inputs	Contains the setpoint, which must be scaled between 0.0 and 1.0.
8	Output (M_n)	REAL	Input/ output	Contains the calculated output, scaled between 0.0 and 1.0.
12	Gain (K_C)	REAL	Inputs	Contains the gain, which is a proportional constant. Can be a positive or negative number.
16	Sample time (T_S)	REAL	Inputs	Contains the sample time, in seconds. Must be a positive number.
20	Integral time or reset (T_I)	REAL	Inputs	Contains the integral time or reset, in minutes.
24	Derivative time or rate (T_D)	REAL	Inputs	Contains the derivative time or rate, in minutes.
28	Bias (MX)	REAL	Input/ output	Contains the bias or integral sum value between 0.0 and 1.0.
32	Previous process variable (PV_{n-1})	REAL	Input/ output	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	Inputs	See the table "Expanded description of control and status fields".
41	AT Status (ASTAT)	BYTE	Outputs	See the table "Expanded description of control and status fields".
42	AT Result (ARES)	BYTE	Input/ output	See the table "Expanded description of control and status fields".
43	AT Config (ACNFG)	BYTE	Inputs	See the table "Expanded description of control and status fields".
44	Reserved	REAL	Inputs	Reserved
48	Reserved	REAL	Inputs	Reserved
52	Initial Output Step (STEP)	REAL	Inputs	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	Inputs	Maximum time allowed between zero crossings in seconds (range: 60 to 7200).
60	Suggested Gain (AT_K_C)	REAL	Outputs	Suggested loop gain as determined by the auto-tune process.
64	Suggested Integral Time (AT_T_I)	REAL	Outputs	Suggested integral time as determined by the auto-tune process.
68	Suggested Derivative Time (AT_T_D)	REAL	Outputs	Suggested derivative time as determined by the auto-tune process.
72	Actual Step size (ASTEP)	REAL	Outputs	Normalized output step size value as determined by the auto-tune process.
76	Actual Hysteresis (AHYS)	REAL	Outputs	Normalized PV hysteresis value as determined by the auto-tune process.

Shift by	Field	Format	Type	Description
80	Reserved	WORD	Inputs	Reserved
82	Reserved	BYTE	Inputs	Reserved
83	Reserved	DWORD	Inputs	Reserved
87	Reserved	DWORD	Inputs	Reserved
91	Loop Setpoint Range	REAL	Inputs	The default value is 100.0.
95	Offsets for fine-tuning	REAL	Inputs	Offsets for fine-tuning
99	Control spec	BYTE	Inputs	Control spec: Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 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
100	Dead Zone	REAL	Inputs	Dead Zone: Range 0.0-1.0
104	Control Range	REAL	Inputs	Control range: Range 0.0-1.0
108	Reserved	BYTE	Inputs	Reserved
109	Integral Upper Limit	BYTE	Inputs	Integral Upper Limit: Range 0-100 • 0: Integral is disabled. • 100: Integral limit is disabled.
110	Reserved	WORD	Inputs	Reserved
112	Reserved	REAL	Inputs	Reserved
116	Reserved	REAL	Inputs	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 13-2 Loop table (Offset 120-240 used for reverse/cooling side)

Shift by	Field	Format	Type	Description
120	Reserved	REAL	Inputs	Reserved
124	Reserved	REAL	Inputs	Reserved
128	Output (M_n)	REAL	Input/ output	Contains the calculated output, scaled between 0.0 and 1.0.
132	Gain (K_C)	REAL	Inputs	Contains the gain, which is a proportional constant. Can be a positive or negative number.
136	Reserved	REAL	Inputs	Reserved

Shift by	Field	Format	Type	Description
140	Integral time or reset (T_I)	REAL	Inputs	Contains the integral time or reset, in minutes.
144	Derivative time or rate (T_D)	REAL	Inputs	Contains the derivative time or rate, in minutes.
148	Bias (MX)	REAL	Input/ output	Contains the bias or integral sum value between 0.0 and 1.0.
152	Reserved	REAL	Inputs	Reserved
156	PID Extended Table ID	ASCII	Constant	'PIDB' (PID Extended Table for bidirectional PID, Version B): ASCII constant
160	AT Control (ACNTL)	BYTE	Inputs	See the table "Expanded description of control and status fields".
161	AT Status (ASTAT)	BYTE	Outputs	See the table "Expanded description of control and status fields".
162	AT Result (ARES)	BYTE	Input/ output	See the table "Expanded description of control and status fields".
163	AT Config (ACNFG)	BYTE	Inputs	See the table "Expanded description of control and status fields".
164	Reserved	REAL	Inputs	Reserved
168	Reserved	REAL	Inputs	Reserved
172	Initial Output Step (STEP)	REAL	Inputs	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	Inputs	Maximum time allowed between zero crossings in seconds (range: 60 to 7200).
180	Suggested Gain (AT_K_C)	REAL	Outputs	Suggested loop gain as determined by the auto-tune process.
184	Suggested Integral Time (AT_T_I)	REAL	Outputs	Suggested integral time as determined by the auto-tune process.
188	Suggested Derivative Time (AT_T_D)	REAL	Outputs	Suggested derivative time as determined by the auto-tune process.
192	Actual Step size (ASTEP)	REAL	Outputs	Normalized output step size value as determined by the auto-tune process.
196	Actual Hysteresis (AHYS)	REAL	Outputs	Normalized PV hysteresis value as determined by the auto-tune process.
200	Reserved	WORD	Inputs	Reserved
202	Reserved	BYTE	Inputs	Reserved
203	Reserved	DWORD	Inputs	Reserved
207	Reserved	DWORD	Inputs	Reserved
211	Loop Setpoint Range	REAL	Inputs	The default value is 100.0.
215	Offsets for fine-tuning	REAL	Inputs	Offsets for fine-tuning

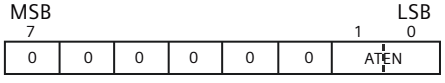
Shift by	Field	Format	Type	Description
219	Control spec	BYTE	Inputs	Control spec:

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Fast Response System	Reserved	CNTL_TYPE	Reserved		INTEGRAL_LIM_EN	DZ_EN	RG_EN

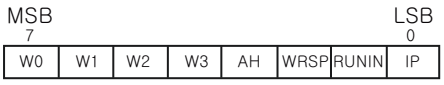
Fast Response System: 0 - Non-fast Response System; 1 - Fast Response System
DZ_EN: 0 - Dead zone not enable; 1- enable
RG_EN : 0 - Control range not enable; 1- enable
CNTL_TYPE: 0 - Normal, 1- Temperature
INTEGRAL_LIM_EN： 0 - Integral Limit not enable; 1- enable

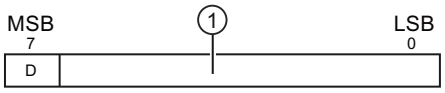
Shift by	Field	Format	Type	Description
220	Dead Zone	REAL	Inputs	Dead Zone: Range 0.0-1.0
224	Control Range	REAL	Inputs	Control range: Range 0.0-1.0
228	Reserved	BYTE	Inputs	Reserved
229	Integral Upper Limit	BYTE	Inputs	Integral Upper Limit: Range 0-100 • 0: Integral is disabled. • 100: Integral limit is disabled.
230	Reserved	BYTE	Inputs	Reserved
231	Reserved	BYTE	Inputs	Reserved
232	Reserved	REAL	Inputs	Reserved
236	Reserved	REAL	Inputs	Reserved

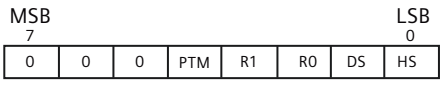
Table 13-3 Expanded Description of Control and Status Fields

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.

Field	Description	
AT Status (ASTAT) Output - Byte		
	W0	Warnings: The deviation setting is not four times greater than the hysteresis setting.
	W1	Warnings: Inconsistent process deviations may result in the incorrect adjustment of output step value.
	W2	Warnings: The actual average deviation is not four times greater than the hysteresis setting.
	W3	Warnings: 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			
	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	Quick 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.

13.5 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.

13.6 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.

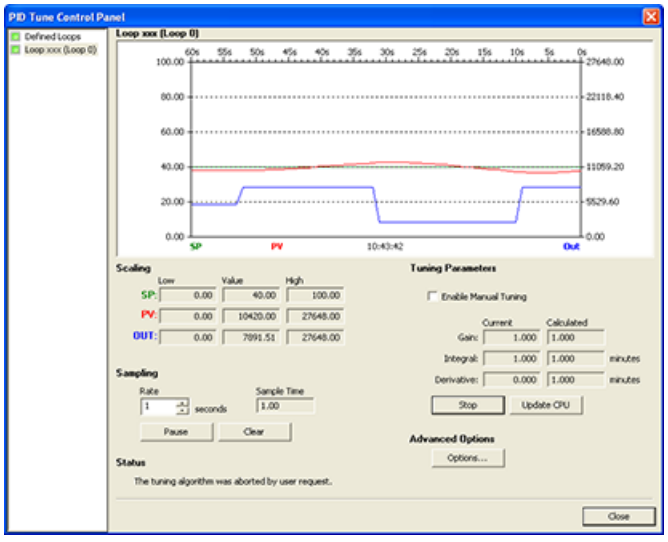
This change in output value causes a corresponding change in the value of the process variable. When the output change drives the PV away from setpoint far enough to exceed the hysteresis boundary, a zero-crossing event is detected by the auto-tuner. Upon each zero-crossing event, the auto-tuner drives 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.

13.7 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 13-4 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

13.8 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.

13.9 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.

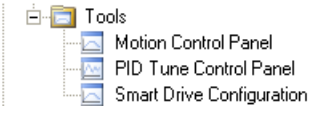
To use the control panel, you must be communicating with the CPU and a wizard-generated configuration for a PID loop must exist in the CPU. The CPU must be in RUN mode for the control panel to display the operation of a PID loop.

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 (Page 149), select the "PID Tune Control Panel" node and press Enter; or double-click the "PID Tune Control Panel" node.



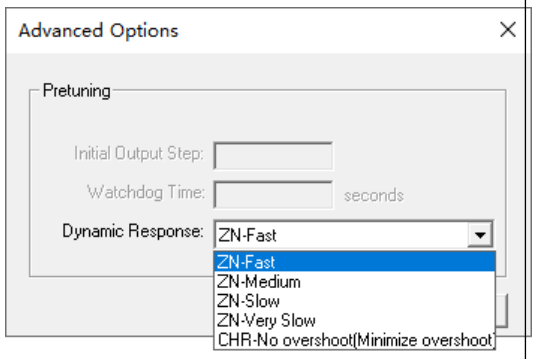
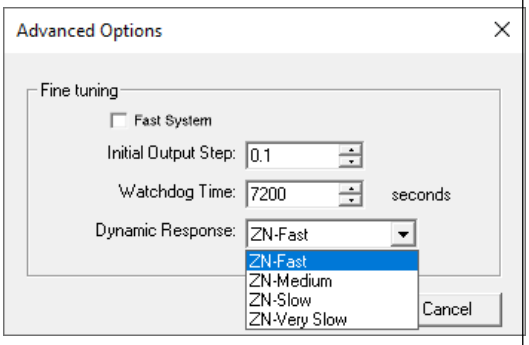
STEP 7-Micro/WIN SMART V3 opens the PID control panel if the connected CPU is in RUN mode:



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.

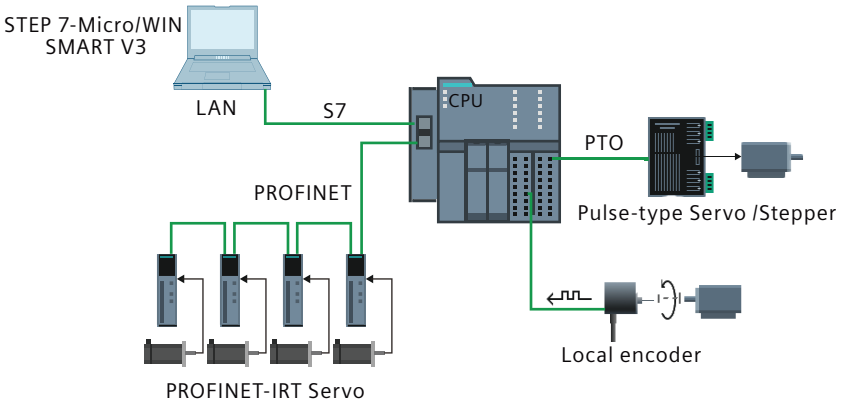
Motion control(PLCopen)

14

The S7-200 SMART motion control provides an easy-to-implement solution for open-loop and closed-loop control of motion axes.

According to PLCopen, drives with PROFIdrive functionality and pulse-type drives will be controlled via standard motion control instructions. The axis control panel and diagnostic functions support easy commissioning and optimization of drives.

S7-200 SMART motion control overview



Motion tasks and functions

S7-200 SMART Motion Control supports the following functions:

Motion task/Function
Continuous Continuous motion can be achieved in rollers, belts, chains, and other conveyor systems.
Speed-controlled continuous motions
Torque limiting
Positioning It enables precise cyclic positioning for motion tasks requiring dynamic response and accuracy.
Precise absolute and relative positioning (closed-loop position control)
High dynamic response
Travel range is limited by hardware and software limit switches.
Linear and rotation axes
Camming/gearing For continuous machining steps requiring precise speed, torque, and position, coordinated axis movement operations of production machines can be achieved.
Gearing without specification of the synchronous position
Camming

Motion task/Function
Axes group Handling control can be achieved through up to three mechanically coupled axes.
Calibration of coordinate systems and zones
Moving the motion system in a linear manner

14.1 Basic concept

RT classes in PROFINET IO

PROFINET IO is a scalable, real-time communication system based on Ethernet technology. The scalable approach is reflected in several real-time classes:

- **RT:** Transmission of data in prioritized, non-isochronous Ethernet frames. The required bandwidth is within the free bandwidth area for TCP/IP communication.
- **IRT:** Isochronous transmission of data with high stability for time-critical applications (for example, motion control). The required bandwidth is from the area of bandwidth reserved for cyclic data.

Depending on the device, not all real-time classes are supported.

IPO (Interpolate)

In systems with IRT axes, the core function of IPO is to perform interpolation calculations on pre-planned macroscopic motion curves (such as straight lines, arcs, or complex Traces) within each fixed IRT cycle, thereby accurately calculating the distance the axis moves and the target speed within the current cycle.

PROFIdrive

PROFIdrive is the standardized drive technology profile for connecting drives and encoders via PROFINET IO. Drives that support the PROFIdrive profile are connected according to the PROFIdrive standard.

PROFIdrive telegrams

PROFIdrive telegrams are used to transfer setpoints and actual values, control words and status words, as well as other parameters between the controller and the drive/encoder.

When a PROFIdrive telegram is used for connection, the drives and encoders are handled and switched on in accordance with the PROFIdrive profile.

Telegram types supported by IRT axes (Page 803):

- Main telegram: Supports telegram 5 and telegram 105
- Additional telegram: Supports telegram 750

Axis

In motion control systems, the object of motion control is called an "axis".

Axis types	Description
Virtual Axis	STEP 7-Micro/WIN SMART V3 offers the possibility to configure an axis as a virtual axis. A virtual axis behaves like a real axis, but has no drive or encoder connection. The setpoints are only processed within the controller.
PTO axis	Axis type that controls motor movement via pulse train output: • Open loop control • The pulse frequency determines the rotational speed, and the number of pulses determines the displacement.
Encoder Axis	Currently, only local encoder axes are supported. Encoder axes can be used to monitor the motion status of axes and can serve as master axes for applications such as cams and gears. • Single-axis instructions: Only MC_Home (modes 0 and 1), MC_TouchProbe, and MC_AbortTrigger are supported. • Synchronous motion: It can only be used as the master axis for the MC_CamIn and MC_GearIn instructions. • Axes Group: Not possible
IRT axis	Synchronous motion axes supporting IRT (Isochronous Real-Time) communication protocol: Each IRT is assigned a drive, e.g. via a PROFIdrive telegram, and an encoder, via a PROFIdrive telegram. By configuring parameters for the hardware interface, mechanical limit switches, and homing operation, a mapping relationship between encoder values and actual positions can be established. Even if a position reference has not been established (e.g., the axis has not homed), motion instructions can still be executed. Even when not homed, the axis can execute relative displacement instructions.

Axes group

A coordinated control unit for two or more motion axes.

Camming

During camming, master axes and slave axes are coupled by a synchronous operation function, which you specify using a cam.
The transmission behavior during camming is expressed by the cam curve.

Gearing

During gear operation, the position of the slave axis equals the master axis position multiplied by the gear ratio. You specify the gear ratio as a ratio of two integers. The result is a linear synchronous operation function.

Motion control instructions

Use motion control instructions to initiate motion control jobs in the axes of the user program, thereby executing the required functions in the axes. You track the status of running jobs with the output parameters of the Motion Control instructions.

Job

It is triggered upon receiving a motion control instruction to verify the input of the motion control instruction.

14.2 Motion control resources

Object			Description	
CPU version			V3.0/V3.1 CPU	V3.2 CPU
Maximum number of control axes			It supports a maximum of 16 axes in total.	• It supports a maximum of 16 axes in total. • It supports a maximum of 8 physically movable axes (PTO and PN IRT axes).
	PTO axis		5 max.	5 max.
	Virtual Axis		max. 16	max. 16
	Local encoder axis		2 max.	2 max.
	PN IRT axis		0	8 max.
	Linear interpolation control maximum number of axes		A single axis group can control up to 3 axes.	A single axis group can control up to 3 axes.
Maximum number of axis groups	2 max.			
Cam	Each PLC allows up to 8 cams to be added.			
	Cam data points	Maximum key-points per cam table support	361	
		Maximum points per cam table support	65535	
		Maximum number of key points added for all cam tables	1444	

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.

Note

The PN IRT axis does not currently support configuration as an axis group.

PROFINET IRT configuration recommendations

To ensure system performance and stability when planning and configuring PROFINET IRT-based motion control systems, refer to the following recommendations regarding IRT cycle, motion control resource utilization rate and number of axes:

IRT cycle	Motion control resource utilization rate	Recommended number of IRT axes
1 ms	50%	No more than 2
2 ms	50%	No more than 4
4 ms	50%	No more than 8
4 ms	65%	8 IRT axes + 8 virtual axes

Example of an extreme configuration:

For complex systems with three IRT axes, five PTO axes and eight virtual axes, it is recommended to set the IRT cycle at 4ms and adjust the motion control resource utilization rate to 75% in order to ensure stable system operation.

Note

In motion control applications, a processor resource utilization rate of 50% means that the motion control task has consumed half of the available central processing unit (CPU) computing power. Under this specific load level, if user interrupts and other communication overhead are not considered, the system's scan cycle will at most be twice as long as when there are no motion control tasks running.

PTO axis resource utilization

PTO (Pulse Train Output) axes are using a 4ms synchronous interpolation clock, and 5 PTO axes will occupy about 35% of the motion control resource.

14.3 Motion control overview

14.3.1 Control mode

14.3.1.1 Position-controlled mode (closed loop)

Together with the controller in the drive, the axis forms a cascade control system. The innermost control cascade is the current control, the next cascade is the velocity control. Both are located in the drive. The position controller is the outermost cascade loop within the axis object in the PLC.

The axis position controller is a closed-loop P controller with or without velocity precontrol. Use the servo gain factor to set the gain of the proportional-action controller. To achieve a high dynamic response of the position control loop, the Kv factor should be set as large as possible. The Kv factor is used in the position controller to form the portion of the

manipulated variable from the control deviation and thus to correct the position difference and disturbance variables.

When position control is enabled, encoder systems, actual value calculation, controllers and monitors are active.

If the closed-loop position control is inactive, the encoder system, actual value calculation, and monitoring are activated on the actual value side.

In follow-up mode, the setpoint tracks the actual value. Motion jobs are not executed. Actual position and actual velocity are updated. This means that tracking is possible when the axis is being moved by an external influence.

Dynamic Servo Control (DSC)

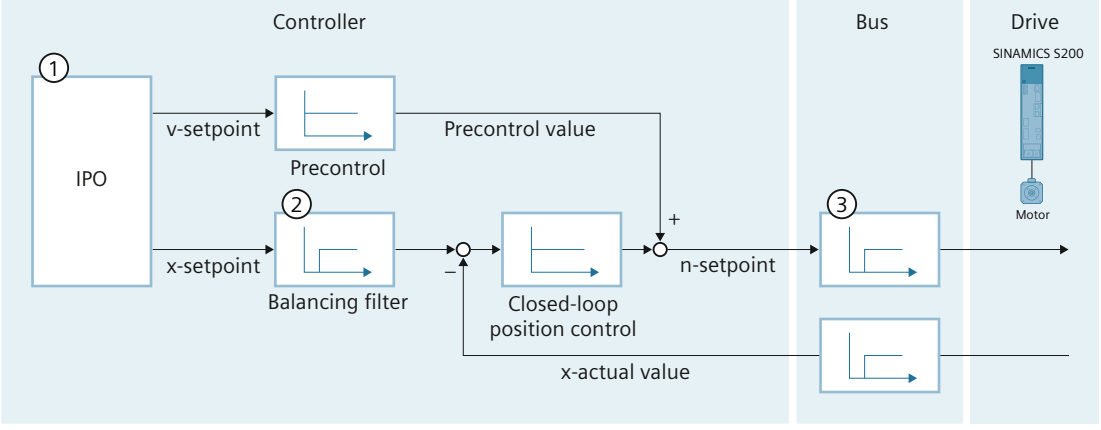
In drives that support Dynamic Servo Control (DSC), you can optionally use the closed-loop position controller in the drive. If you use telegrams that support DSC, DSC is automatically activated. The position controller in the drive is usually implemented with a rapid speed-control cycle. This improves the control performance for digitally coupled drives.

The following requirements must be met to use DSC:

- Use the motor encoder of the drive (the first encoder in the telegram) as the first encoder for the axis.
- One of the following PROFIdrive telegrams is used for the drive:
  - Standard telegram 5
  - SIEMENS telegram 105

Control structures

- Position control in the drive; for this the drive must support Dynamic Servo Control (DSC). When used with a SINAMICS drive, this is the standard case, since the faster control cycle in the drive (e.g. 125 µs) leads to an higher control quality.
  The following figure shows the effective closed loop control structure **with** DSC:



① Interpolator with motion control
② Internal considerations for speed loop response time
③ Communication between controller and drive

Speed precontrol

The speed precontrol can be used to minimize the following error during position control. As a result, faster positioning is achieved, if needed, because the position controller only has to compensate disturbances.

When using the speed precontrol, the speed setpoint is additionally added to the output of the position controller. You can weight this additional setpoint by a factor.

Speed loop response time

In this field, configure the speed loop response time (T_{vtc}).

The speed loop response time is included in the balancing filter.

The balancing filter is a simplified model of the closed speed control loop. The balancing filter is used to prevent the position controller from overriding the speed manipulated variable during the acceleration and deceleration phases. To accomplish this, the position setpoint of the position controller is delayed by the amount of the speed loop response time in relation to the velocity precontrol.

Regarding the configuration of the speed loop response time, please note the following:

- If no speed precontrol is used (0%), the configuration of the speed loop response time is irrelevant.
- If speed precontrol (>0%) is used and the equivalent time of the speed loop response time is set to 0.0 s (default value), the axis will overshoot. To find the right setting, optimize the position controller.

Position loop gain (Kv factor)

In this field, you configure the gain Kv of the position control loop.

The Kv factor affects the following parameters:

- Positioning accuracy and stop control
- Uniformity of motion
- Positioning time

The better the mechanical conditions of the real axis (stiffness), the higher the Kv factor can be configured. This reduces the following error, and a higher dynamic response is achieved.

14.3.1.2 Speed control mode

When the motion control axis uses Profidrive DSC control mode, the PLC will periodically send speed commands to the drive.

MC_MoveJog and MC_MoveVelocity support speed control modes.

Setpoints and parameters in speed control mode

Position settings are no longer effective; only speed commands are retained.

- Precontrol: 100%.
- Kv: 0, the position control loop is disabled.

Input pins

The MC_MoveVelocity and MC_MoveJog instructions have added a new pin, SpeedControlled, for selecting between position control or speed control.

Note

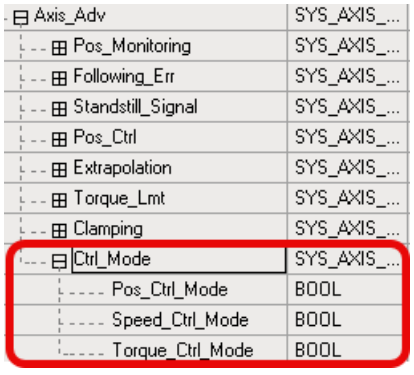
Speed control mode is supported only for the IRT axis and the virtual axis. This mode is not currently supported for PTO axes and encoder axes. For axes that do not support speed control mode, error code 0x1038 will be returned.

Operation instructions	Modify SpeedControlled pins
MC_MoveJog	It supports modifying the SpeedControlled parameter when MC_MoveJog is in Busy state. The modified parameters will take effect immediately.
MC_MoveVelocity	After modifying the SpeedControlled pin, a rising edge signal is required for the changes to take effect. Therefore, for the MC_MoveVelocity function block, this behavior is also considered an instruction restart.

Current control mode of the axis

A new field, "Current Axis Control Mode", has been added to the variable table to indicate whether the current axis is in position control mode, speed control mode, or torque control mode.

For PTO axes and encoder axes, this field is always 0.



Speed control mode

Following error

Following error monitoring is only effective in position control mode and is not currently effective in speed control mode.

Software and hardware limit switches

In speed control mode, both software and hardware limit switches remain in effect.

Examples

Preconditions

- The axis has been set up and correctly configured with following error monitoring and software and hardware limit switches.
- The program uses the MC_MoveJog or MC_MoveVelocity instruction.

Execution process and results

Execution conditions	Results
Set SpeedControlled to TRUE to trigger the instruction.	Following error monitoring will not work: The settings information shows that the position setpoint is 0. The axis control mode is displayed as speed control. When the actual position of the axis reaches the hard/software limit, the axis will stop abruptly according to the configured emergency stop method and enter ErrorStop.
Set SpeedControlled to False to trigger the instruction.	Following error monitoring is back in effect: The position displayed in the settings information is equal to the actual position after switching, and then increases or decreases according to the movement. The axis control mode is displayed as position control mode.

Cam and Gear

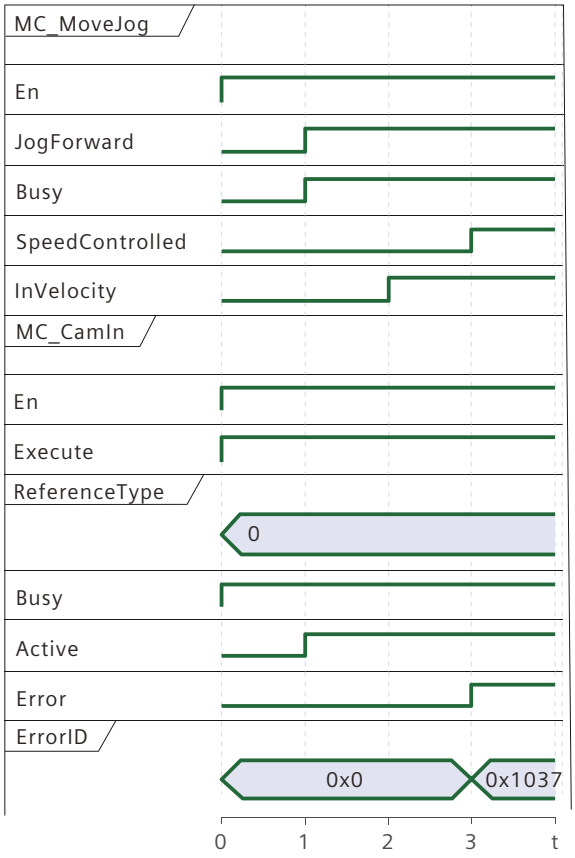
In speed control mode, the slave axis can perform cam-synchronized or gear-synchronized motion only when the synchronous reference position of the slave axis is configured as the actual position of the master axis.

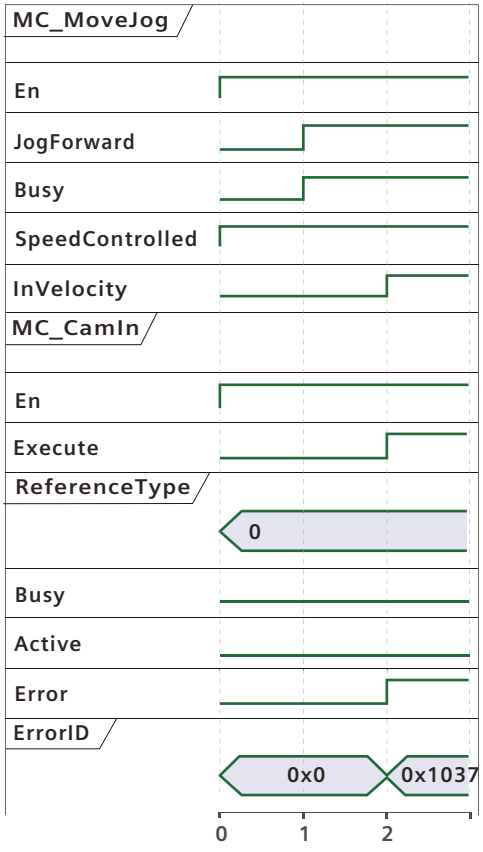
Examples

Preconditions

- Axis is configured.
- The master axis in the program calls the MC_MoveJog or MC_MoveVelocity instruction. By calling the MC_CamIn or MC_GearIn instruction on the slave axis, the slave axis can achieve synchronous movement with the master axis.

Timing diagrams


MC_MoveJog
En
JogForward
Busy
SpeedControlled
InVelocity
MC_CamIn
En
Execute
ReferenceType
0
Busy
Active
Error
ErrorID
0x0
0x1037
0
1
2
3
t



Switching control mode

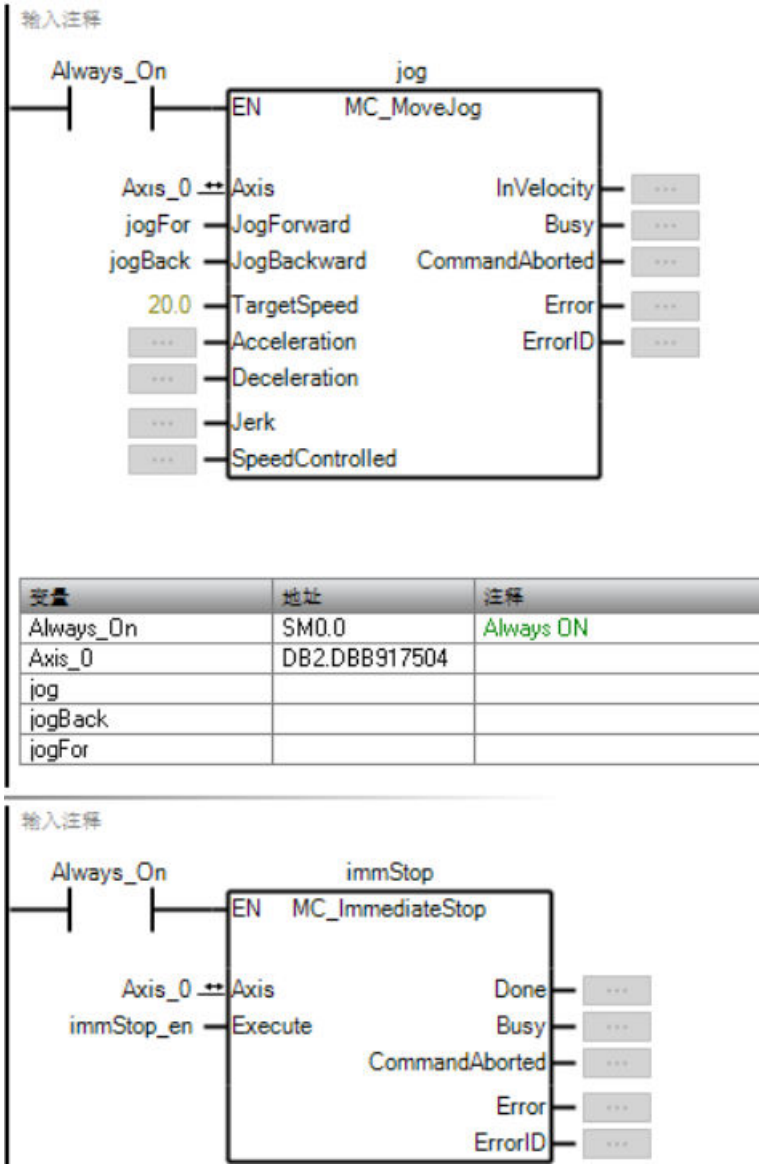
The speed control mode can be switched to the position control mode. After the switch is complete, Setting_Info.Pos will be automatically updated to the current actual position.

Can switch to position control mode	Can switch to torque control mode	instructions that will not affect the control mode	Instructions that cannot be executed in speed mode
MC_MoveRelative MC_MoveAbsolute MC_Power MC_CamIn MC_GearIn MC_CamOut MC_GearOut MC_Home	MC_TorqueControl (mode 1)	MC_ImmediateStop MC_Stop MC_Halt MC_Reset MC_TorqueRange MC_TorqueControl (mode 0) MC_TorqueLimit (mode 0: limit mode)	MC_CamIn (The axis running in speed mode is the main axis, and reference type=1) MC_GearIn (The axis running in speed mode is the main axis, and reference type=1) MC_TorqueLimit (mode 1 Fixed stop point mode)

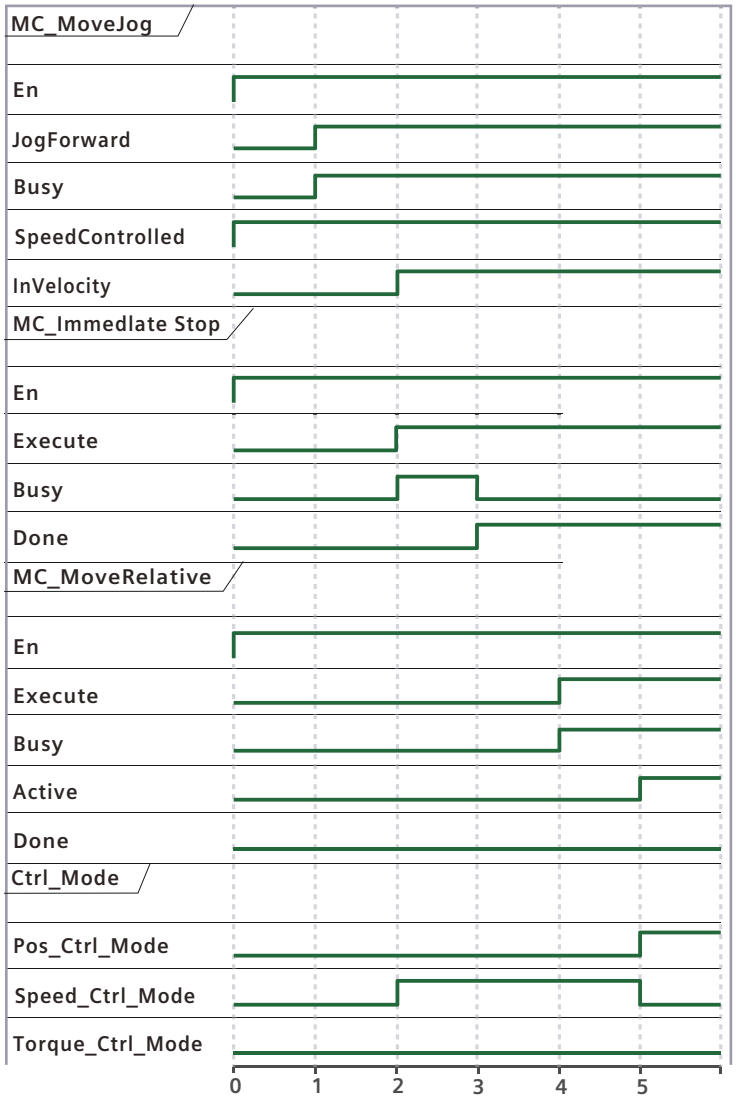
Example 1

Preconditions

- Axis is configured.
- The program uses the MC_MoveJog or MC_MoveVelocity instruction.



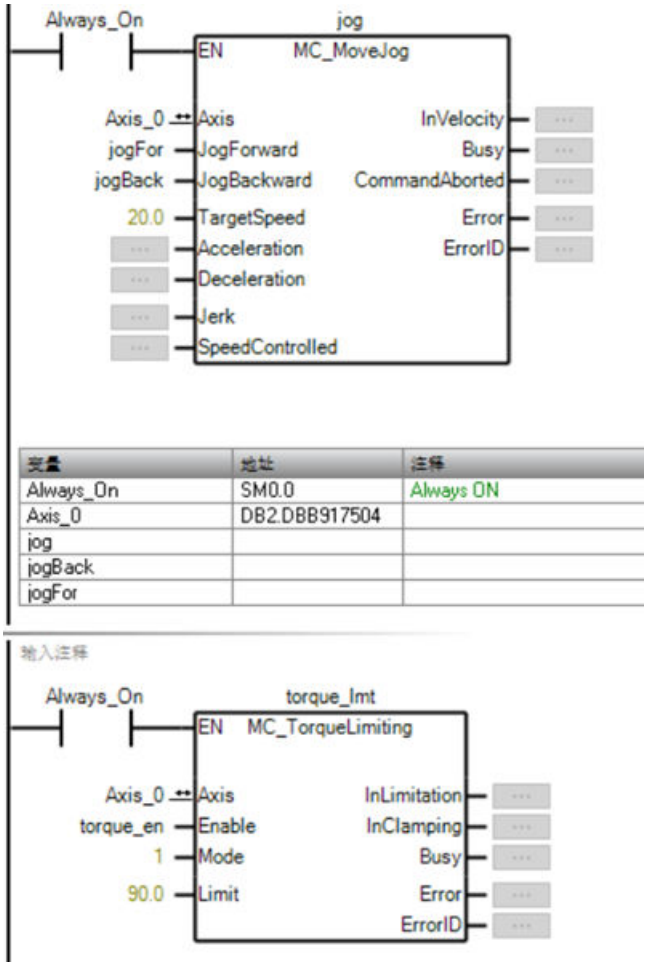
Timing diagrams



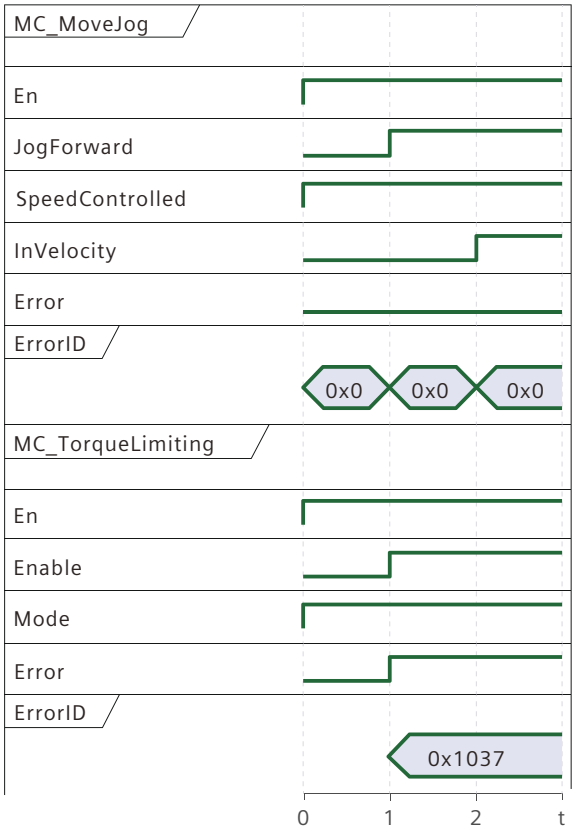
Example 2

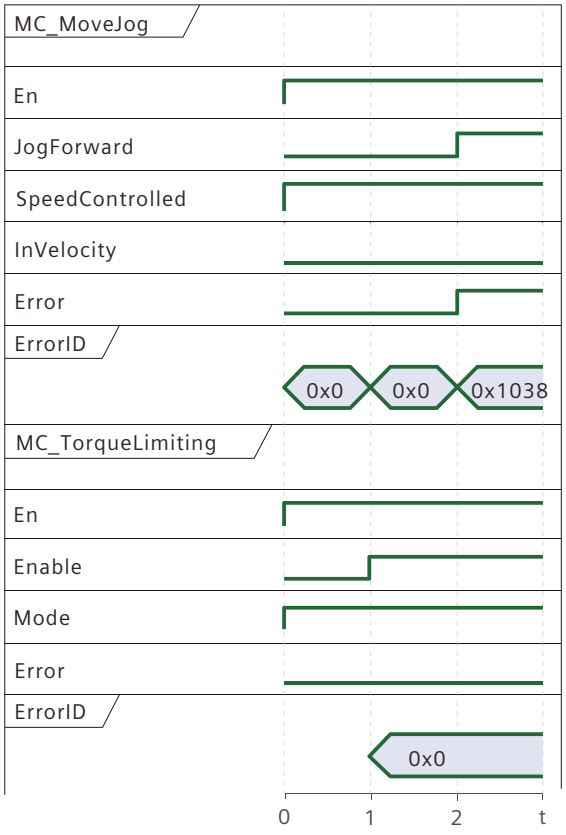
Preconditions

- Axis is configured.



Timing diagrams


MC_MoveJog
En
JogForward
SpeedControlled
InVelocity
Error
ErrorID
0x0
0x0
0x0
MC_TorqueLimiting
En
Enable
Mode
Error
ErrorID
0x1037
0
1
2
t



14.3.1.3 Single-axis torque control

Conditions for using single-axis torque control

- Applicable only to IRT axes.
- Telegram configuration
  - To use the torque control instruction, make sure telegram 750 is configured when you set Mode mode=0 or mode=1.
  - If you want to use mode=1 for torque control, you must configure telegram 105 in addition to telegram 750.

Note

When performing torque control, please be sure to set a speed limit value for safety.

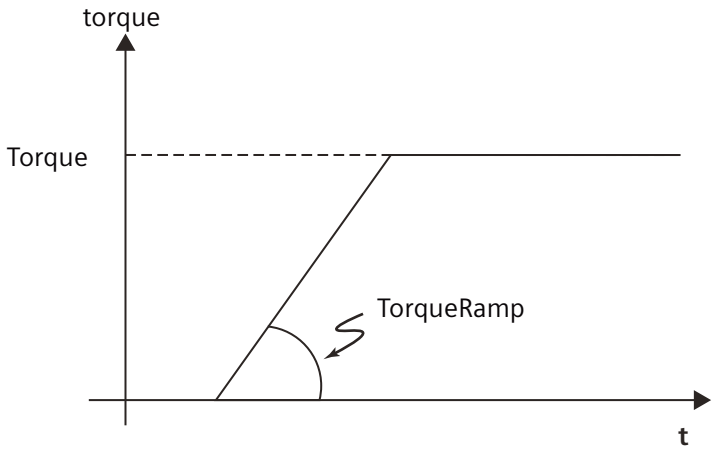
TorqueRamp parameter description

Torque control refers to the function of continuously generating a specified torque.

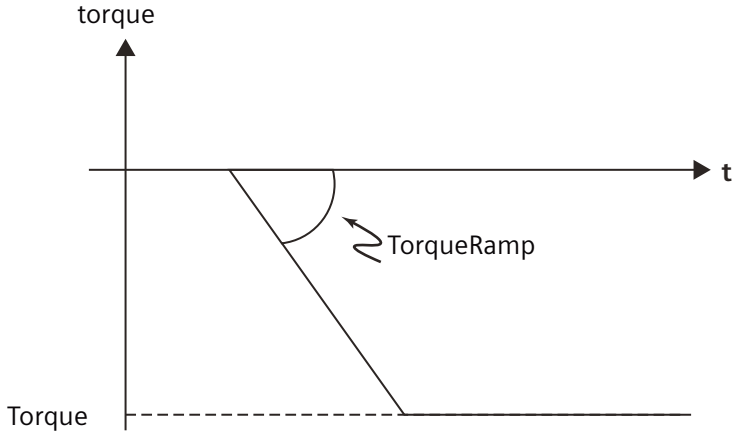
The rate of torque change before reaching the Torque (target torque) can be specified using TorqueRamp (torque slope).

When Torque (target torque) is specified as "0", the axis will not move, but the axis is in a state of motion.

- When direction is specified as = positive



- When the direction is specified as = negative



MC function block application description

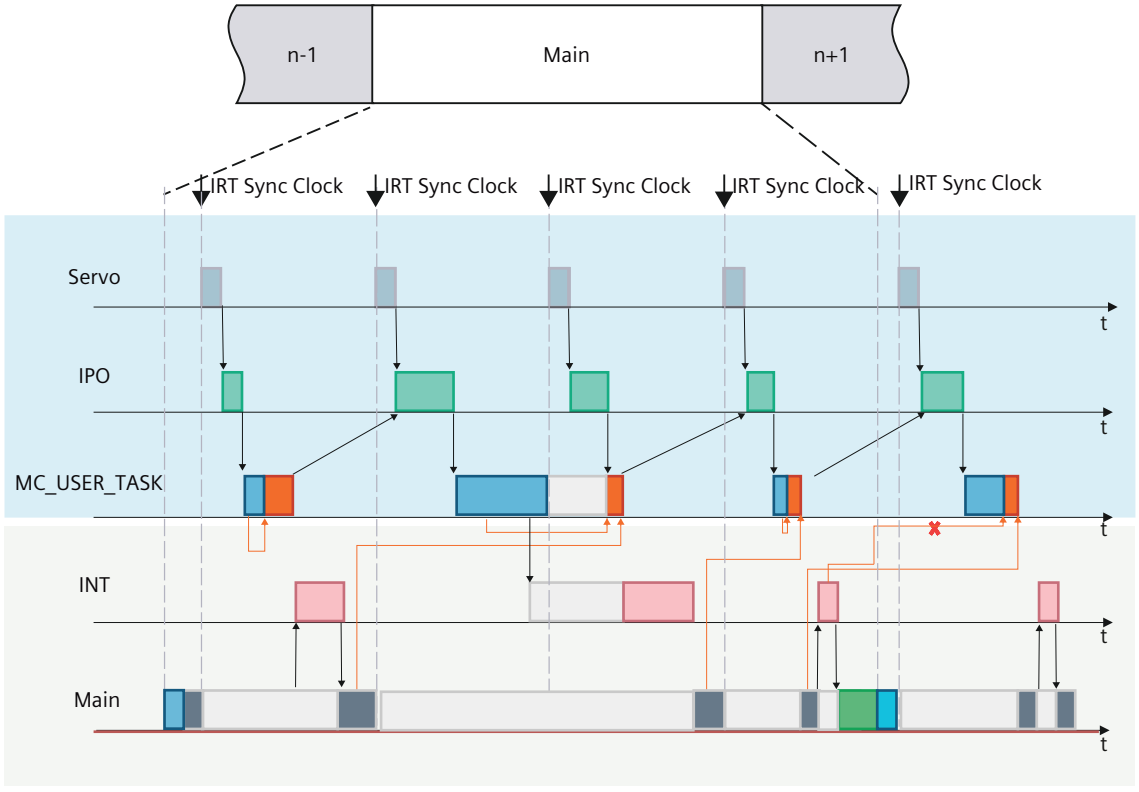
The MC function module uses the torque control mode of the servo driver.

The servo driver uses torque control loop to receive torque instruction values sent by the MC function module for control.

See also

Speed control mode (Page 771)

14.3.2 Task Mechanism



Servo

IRT Servo Control Task

Execution characteristics	Temporal Relationship
• Highest priority, indicating real-time servo control tasks • Periodic execution • Uninterrupted operation ensures real-time and precise control • Directly responds to the interpolation output of the IPO layer	• Execute within each master clock cycle • IPO layer maintains strict time synchronization

IPO

Interpolation task

Execution characteristics	Temporal Relationship
• Interpolation calculations are performed in ascending order of axis number • Event markers indicate the specific interpolation event trigger point • Responsible for generating precise position and motion instructions	• Post-IPO: Post-processing stage after interpolation task is completed • Add IPO task: Dynamically add new interpolation tasks to the execution queue • After completing motion planning, the motion control user task (MC User Task) receives a new task.

Motion control user tasks (MC_USER_TASK)

This task will be executed immediately after the interpolation (IPO) task is completed. All user-defined motion control instructions, whether based on IRT or PTO (Pulse Train Output) mode, will be uniformly parsed and processed in this task.

- When IRT is configured: The task runs synchronously with the IRT cycle to ensure high-precision and deterministic motion control.
- When IRT is not configured: If the system is not configured with IRT functionality, the motion user task will fall back to running in sync with the 4-millisecond base clock cycle of the PTO.

Phase 1	Phase 2	Phase 3
Motion control program execution: • Processing user-defined motion control logic • Prioritized over Main Scan to ensure deterministic motion control	Job processing 1. Collect all asynchronous instruction requests, including: – Main Scan – Motion control user tasks – Control Panel 2. Invoke motion control job to generate IPO task: – Task scheduling is performed uniformly according to priority – Motion planning and interpolation – Send interpolation task to IPO layer	System monitoring tasks

Main Scan

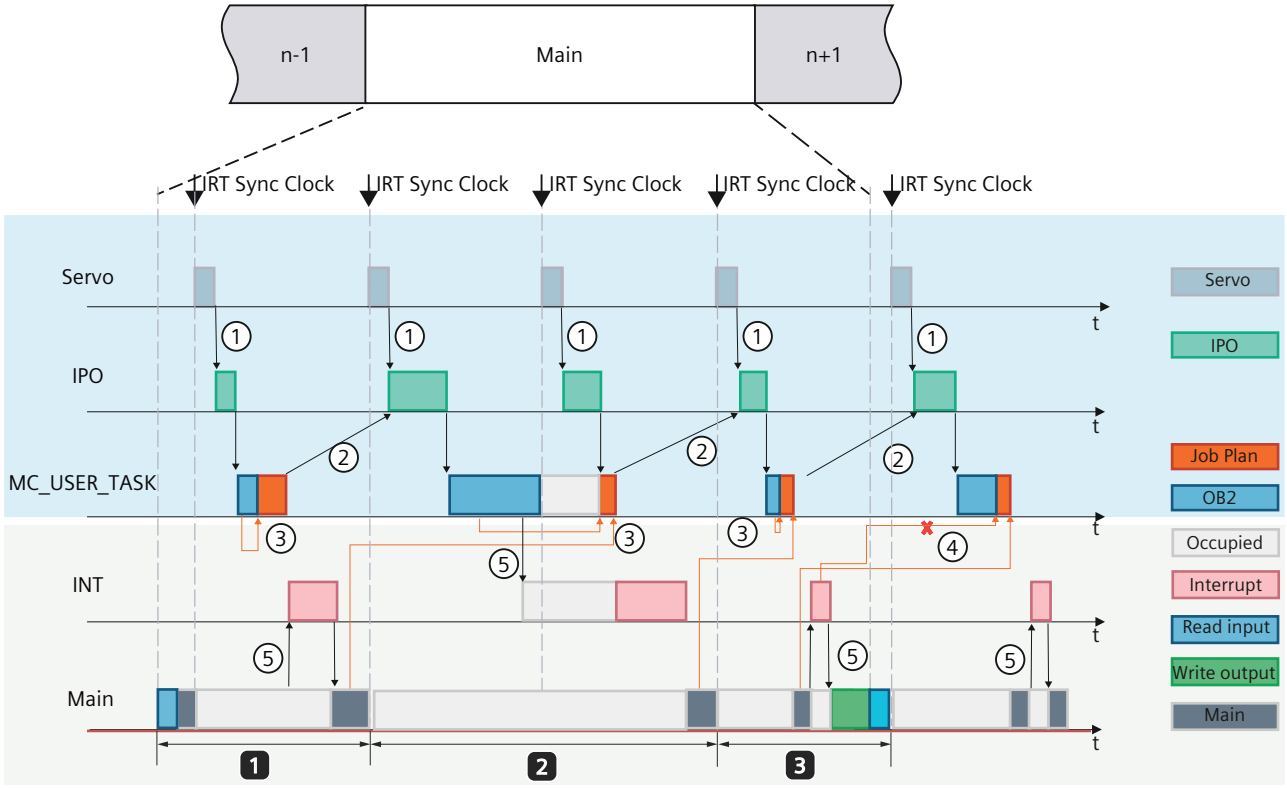
It executes asynchronously with motion control user tasks, without interfering with each other.

Priority sorting

Servo > IPO > MC_USER_TASK > User Interruption > Main Scan

14.3.3 Process reaction

If IRT axes are configured and the motion control user task (MC_USER_TASK) calls and executes motion instructions, it can ensure that the core calculation and interpolation tasks of motion control are processed in a timely manner. After these critical motion control OBs have been completed, the remaining time will be left to the main program to process until the next IRT cycle begins. When using a CPU with version V3.2 or higher and motion control functions need to be configured, MC_USER_TASK can be inserted into the program block.



① Event
② Add IPO (interpolation) task
③ Send job plan
④ Motion control instructions cannot be executed within the interrupt routine
⑤ At this moment, trigger a hardware interrupt

1 Motion control (servo communication + IPO + MC_USER_TASK) is executed within one IRT cycle. The main program scans continuously, and there may be multiple scans within one IRT cycle.

2 Motion control (Communication with Servo + IPO + MC_USER_TASK) cannot be completely executed within 1 IRT cycle. A user interrupt can be triggered, but it can only be executed after the motion control user task (OB2) has finished.
That is, the motion control user task (OB2) can interrupt the execution of a user interrupt, but a user interrupt cannot interrupt the execution of the motion control user task (OB2).

3 Motion control instructions cannot be placed inside interrupt routines

Motion control timeout conditions

Motion control

- Motion control exceeds the IRT cycle three consecutive times
- Motion control average time per 50 cycles, three consecutive times ≥ IRT cycle * Proportion of motion control in CPU resources (Page 218)

MC_USER_TASK timeout conditions

- MC_USER_TASK scan time for three consecutive times ≥ IRT cycle * Proportion of motion control in CPU resources (Page 218) * Proportion of MC_USER_TASK in motion control (Page 218)

14.3.4 Compatibility

14.3.4.1 Changes to motion control in V3.1 and V3.2

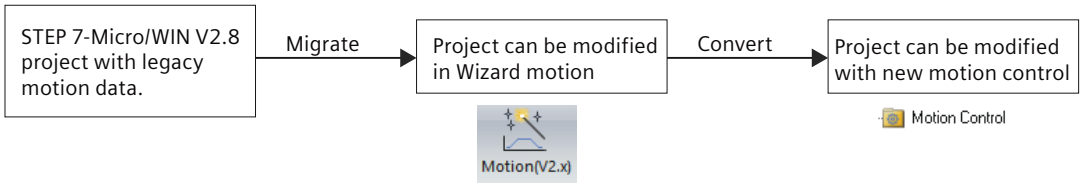
Categories	V3.1	V3.2
PN IRT axis	Not possible	Support
Torque control mode	Not possible	Torque control: MC_TorqueControl Torque limitation: MC_TorqueRange MC_TorqueLimiting
Speed control mode	Not possible	Supports PN IRT axes and virtual axes MC_MoveVelocity [SpeedControlled=1] MC_MoveJog [SpeedControlled=1]
Trace	IRT cycle not supported	IRT cycle is supported
MC_USER_TASK(OB2)	Not possible	Support
MC instruction Jerk pin	Jerk instructions not supported: MC_Halt MC_Stop MC_CamOut MC_GearOut MC_MoveJog (Deceleration segment)	Supported Jerk instructions: MC_Halt MC_Stop MC_CamOut MC_GearOut MC_MoveJog
Buffered	Supported Buffered instructions: MC_Power MC_Halt MC_MoveAbsolute MC_MoveRelative	Supported Buffered instructions: MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn

Categories	V3.1	V3.2
Electronic camming	1. MC_CamIn does not support the following pins: StartMode MasterStartDistance MasterOffset SlaveOffset 2. Error when slave axis overspeeds	1. When the slave axis is a PN IRT axis or a virtual axis, MC_CamIn adds support for the following pins: StartMode MasterStartDistance MasterOffset SlaveOffset 2. When the slave axis overspeeds, it uses maximum speed to catch up
Electronic gearing	Error when slave axis overspeeds	When the slave axis overspeeds, it uses maximum speed to catch up
MC_PhasingRelative	Errors will occur if the following pins exceed their limits: Velocity Acceleration Deceleration	The following pins will operate normally when their limits are exceeded: Velocity Acceleration Deceleration
Fault reset	Only allowed to be triggered under ErrorStop	Allows triggering under any state machine
Touch Probe	MC_TouchProbe: Independent of the axis state machine, independent of the homed flag	MC_TouchProbe: When the axis state machine is in Disabled, Homing, or ErrorStop states, instructions cannot be triggered When the homed flag is not set, instructions cannot be triggered
Virtual Axis	• The homed flag is lost when the axis state machine is Disabled • The axis position is cleared when the CPU switches to RUN mode • The velocity in the axis variable is the average velocity over the interpolation cycle	• The homed flag is not lost when the axis state machine is Disabled • The axis position is not cleared when the CPU switches to RUN mode • The velocity in the axis variable is the end speed of the interpolation cycle, and the InVelocity pin is set one cycle in advance
Software limit switch	Decelerate in advance and stop at the software limit switches Both linear axes and modal axes support software limit switches	Deceleration starts from the software limit switches Only linear axes support software limit switches
Control Panel	Startup is only allowed when the PLC is in STOP mode	• PTO Axis: Startup is allowed during PLC RUN or STOP • PN IRT axis: Startup is only allowed during PLC RUN
MC instruction restart	Interrupt itself	The following instructions allow restarting while in motion: MC_MoveAbsolute MC_MoveRelative MC_Halt MC_MoveVelocity MC_Stop MC_GearIn MC_CamIn
Calling MC instructions in INT	No error message	Error during download

14.3.4.2 New motion control fully replaces the legacy motion wizard

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 (Version 2.8).

- 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 (Page 788) it first. After converting, motion wizard and legacy instructions are disabled.



14.3.4.3 Convert to STEP 7-Micro/WIN SMART V3 (PLCopen) Motion Control

If legacy motion control is migrated from an earlier version STEP 7-Micro/WIN SMART, Siemens recommends you convert the axes created in the legacy motion wizard to the latest STEP 7-Micro/WIN SMART V3 Motion Control.

Note

Convert cannot be restored

This operation cannot be restored, please backup the original file.

Note

The project must be reprogrammed with new instructions

After the conversion, the data blocks generated by "Motion Wizard" and the Motion instruction SBR are not available. The instructions used by the project are also invalid.

You must reprogram the project with new motion control instructions.

Note

Legacy axes group configuration discarded

The legacy axes group configuration is discarded in the current version.

Conditions

- Your project is migrated in STEP 7-Micro/WIN SMART V3.

Covert the project

1. Open the Wizards folder in the project tree.
2. Right-click "Motion(V2.x)" and select "Convert".
3. Click the "Start" button to proceed.
4. Double-click the converted axis to open the configuration view, check each panel and configure accordingly.

Matching rules

Pre-conversion and post-conversion matching rules between some features.

Motion wizard		3.x motion control	Remarks
Name of the axis		Name of the axis	The space in the axis name is replaced with an underscore "_"
Profile of axis **Note**: Profile is bind with axis.		Variables of type SYS_AXIS in the global variables Table 1 **Note**: Any axis can use variable table1	• The vel, endPosition and comments in profile are converted only when the relative or absolute mode is selected. For the other two modes, the pin mode and speed operations must be specified with subsequent profile instructions.
acceleration time/deceleration time: ms		acceleration time/deceleration time: unit/s^2	• acc = (maxVel - ss_speed) / accTime * 1000 • dece= (maxVel - ss_speed) / decelTime * 1000 • ss_peed: start-stop speed. The PTO velocity must be greater than 200 pulses. – pulse: ss_speed=200 – others: ss_speed=200.0 * (the movement of the worktable in one rotation/ (number of pulses per rotation of the motor)) maxVel: Max Velocity
Jerk time (ms)		Jerk time (unit/s^3)	jerk = acc / jerkTime * 1000
Measurement system	foot	inch	
	Rad	degree	
	m	cm	
Single phase (1output)		pulse + direction	
RPS search Sequence 2		home mark (positive) + offset	
Reference point offset		deleted	
ZP input counts in "Search Sequence" (pulses)		Offset in "Homing" page (mm)	offset[1] = (pulse-1)* revolution-unit revolution_unit: the distance that the configured axis (workbench) moves in one revolution.
Axes group, 2D/3D linear interpolation		Axes group type, Coordinate System (2D/3D)	

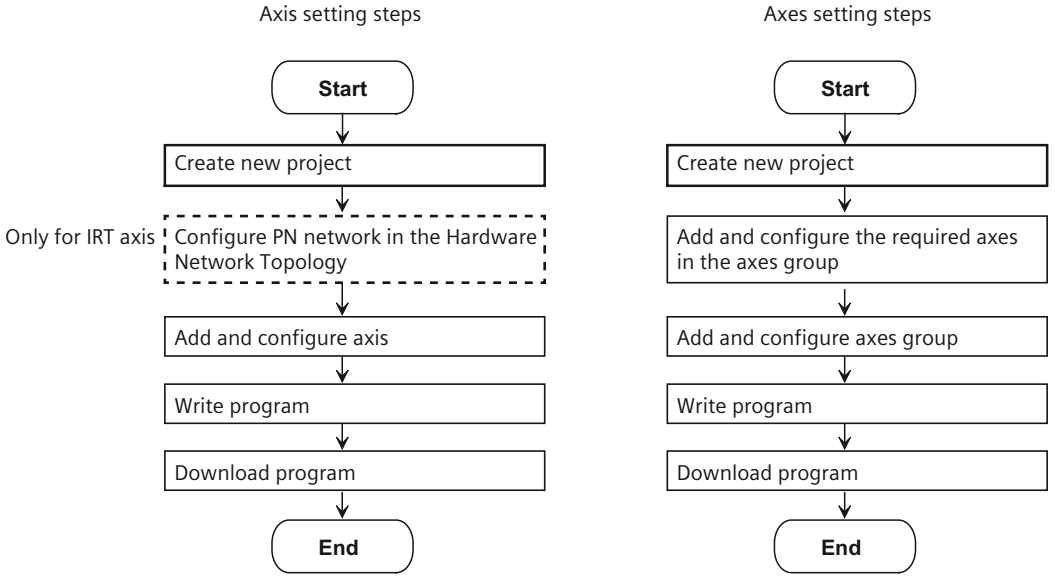
Motion wizard	3.x motion control	Remarks
Path segment of axes group	Variable Table1 in global variable table	The name of the converted variables start with "Grpx_"
Backlash compensation	deleted	

[1] If the approach direction is negative, the offset value is negative; If it is a pulse output, the offset will be "pulse -1" after conversion.

14.4 Configuration

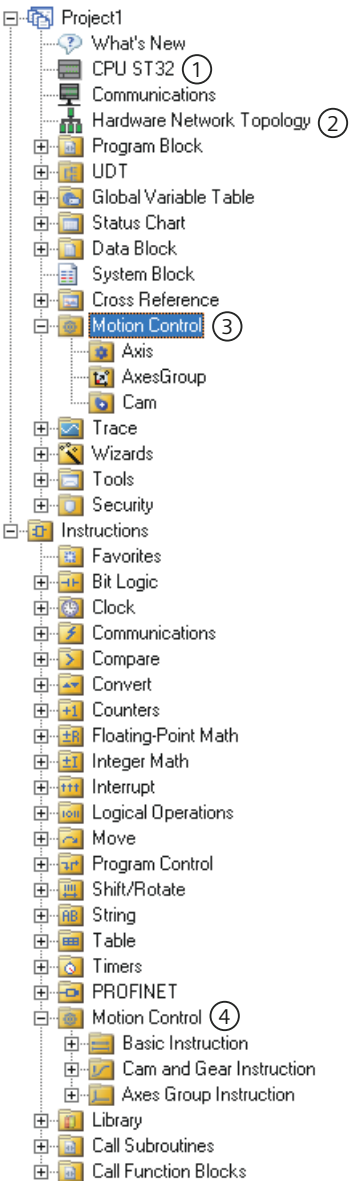
14.4.1 Setting steps

Axis setting process



Note

When creating a new project that includes an IRT axis, ensure that you select the S7200 SMART G2 CPU and that the firmware version is at least 3.2.



① Double-click to select the CPU and its version.

② Double-click to enter the hardware network topology (only applicable to IRT/RT networks).

③ Add and configure motion control axes, axes groups, and cams.

④ Write motion control programs using motion control instructions.

14.4.2 Hardware Network Topology View

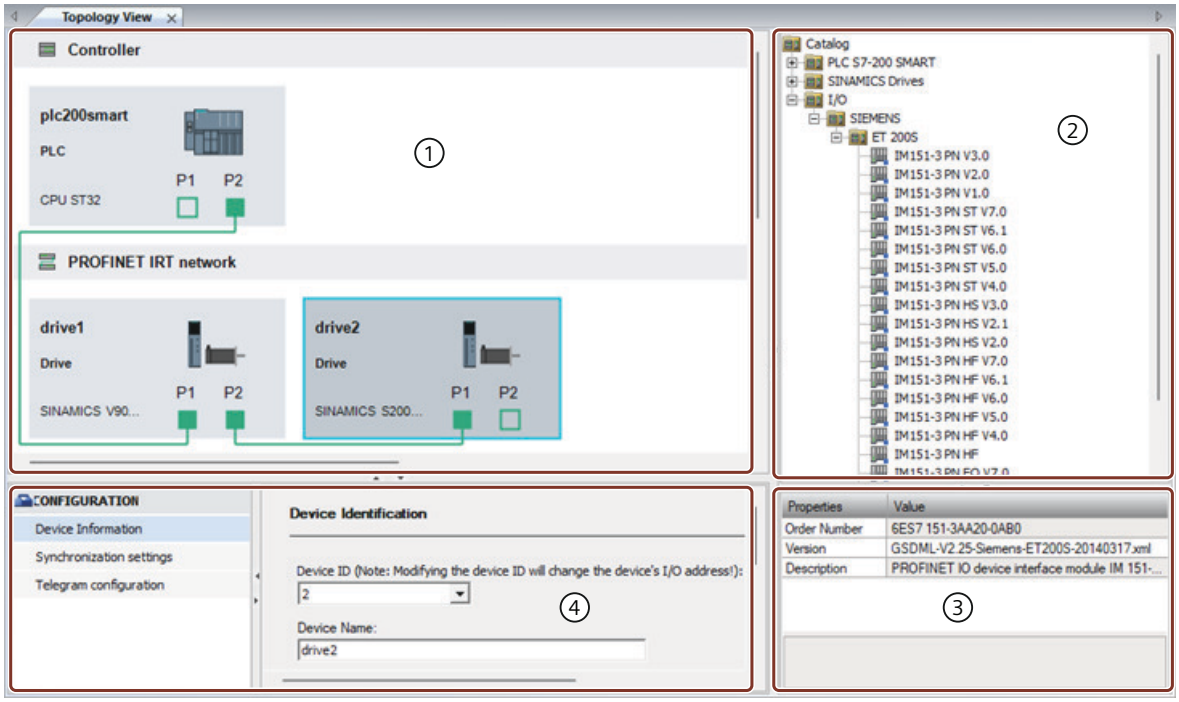
14.4.2.1 Hardware Network Topology

Hardware network topology functions

Hardware network topology is used for configuring PROFINET RT+IRT networks. Double-click "Hardware Network Topology" in the project tree to open the hardware network topology. Hardware network topology is used to configure and assign parameters to devices and modules.

Hardware network topology structure

The hardware network topology consists of the following parts:



① Topology View (Page 793)
② Device Catalog
③ Device Properties
④ Device Configuration

You can use the mouse to change the separation between areas in the network view. To change the position of the divider, click between the area dividers and change the spacing by moving the divider to the left or right while keeping the mouse button pressed. The Speedy Splitter (the two small arrow keys) allows you to use a single click to minimize a view, maximize a view or restore the last selected split.

To efficiently manage and configure network resources, hardware topology networks provide two views: Topology view and device view. In the topology view, you can configure the entire network topology. When you need to edit and configure a single device and its internal modules, you must enter the device view.

Tool bar

The toolbar provides the following functions:

Icon	Meaning
	Set PLC to RUN mode
	Set PLC to STOP mode
	Compile

Icon	Meaning
	Upload project from PLC to STEP 7-Micro/WIN SMART V3
	Download from STEP 7-Micro/WIN SMART V3 to a PLC
	Topology Comparison (Page 810)
	Scan topology (Page 810) Initiate detection of the current online topology.

Device Catalog

When you open the hardware network topology, the device directory tree will automatically load the following devices:

- PROFINET device embedded in the CPU
- PROFINET devices supported by imported GSD files

Device Properties

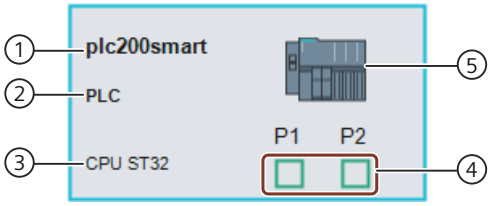
When you select a device in the device catalog, the device properties window will display the device's basic properties.

Property	Description
Order number	Read only
Edition	You can select the GSD file version for your device via the drop-down menu. If multiple GSD files containing the same device are installed on your system, you can select one here. After completing the setup, the device version in the tree view will be updated to the version you selected. By default, the system will automatically select the highest version of the GSD file.
Description	Read only

14.4.2.2 Topology View

In the topology view, the entire page is divided into three layers: Controller layer, PROFINET IRT network layer, PROFINET RT network layer.

Plant



①	Device Name	If the device name is changed, the device view will be updated accordingly.
②	Device Type	PLC, Drive, I/O
③	Device versions	The device model must match the type selected in the system.
④	Port information	• Only PLCs and IRT devices that support IRT functionality will display detailed information for the two ports, labeled P1 and P2 respectively. • RT device: Specific port information will not be provided.
⑤	Device diagram	

Controller layer

The system will automatically create a device based on the CPU selected in the system block.

- The controller layer contains only this one automatically created device.
- This device cannot be deleted.
- Dragging other devices from the hardware catalog or other layers to this layer is not allowed.
- When the CPU switches from supporting IRT to not supporting IRT, all existing topology configuration information will remain unchanged. If incompatibility exists, the system will report an error during compilation.

PROFINET IRT layer

The PROFINET IRT network layer is used to add IRT devices.

- Up to eight IRT devices can be configured in this layer. At the same time, the total number of all IRT and RT devices must not exceed 10.
- Devices at the IRT layer are connected in a daisy-chain manner.
- The layer is divided into two rows, with each row supporting a maximum of four devices.
- If the current CPU type and version do not support IRT functionality, the IRT network layer will disable device drag-in operations.

PROFINET RT network layer

The PROFINET RT network layer is used to add RT devices.

- Up to 10 RT devices can be configured in this layer. At the same time, the total number of all IRT and RT devices must not exceed 10.
- The layer is divided into two rows, with each row supporting a maximum of 5 devices.

Note

Only RT devices support access to the device view. This device view function is not currently available for IRT devices.

14.4.2.3 Device view

The Device View is used to further display and manage detailed information about RT type devices. You can access the device view of an RT device in any of the following ways:

- Double-click the RT device in the topology view.
- Right-click the RT device in the topology view and select the "Go to device view" option from the pop-up context menu.

Note

Only RT devices can access the device view; IRT devices do not have a device view.

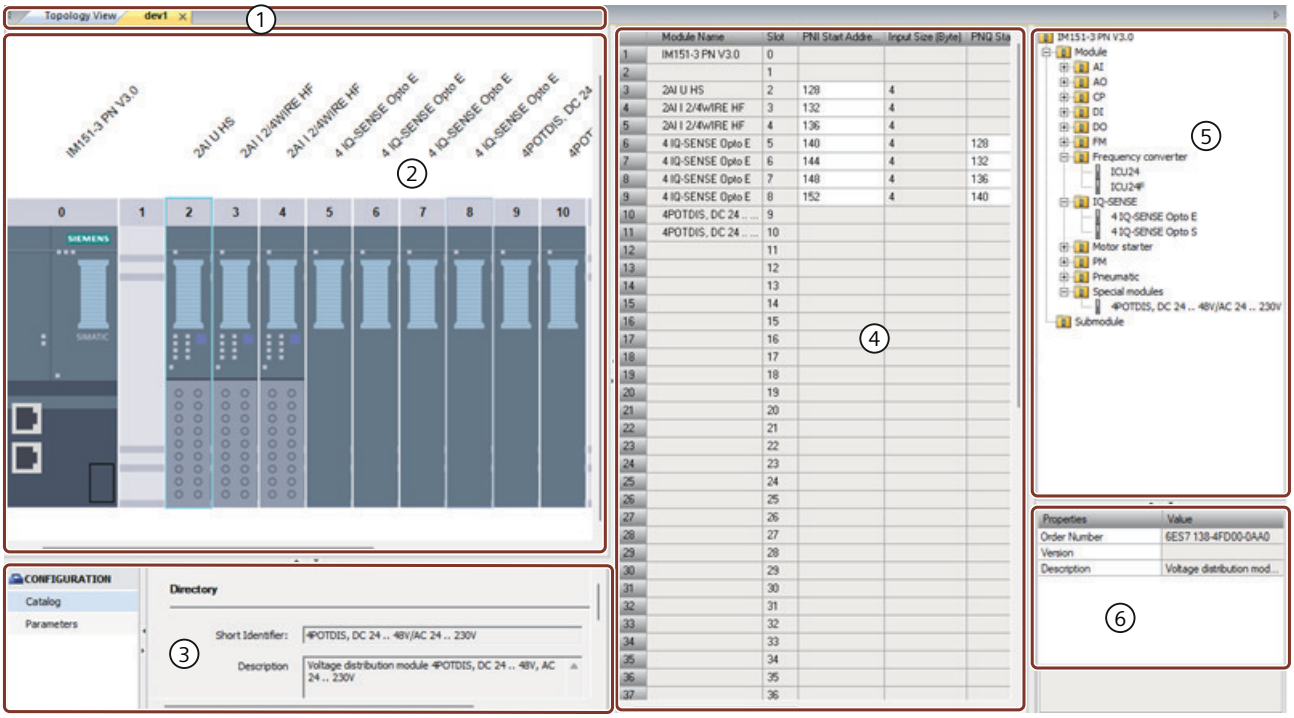
Introduction

You can perform the following tasks in the device view:

- Assigning device parameters
- Configuring and assigning module parameters

Structure

The following diagram shows the components of the device view:



① Changeover switch: Device View/Topology View
② Graphic area of device view
③ Module configuration: When a module or submodule is selected in the graphic or table area, its corresponding parameters will be displayed in the module configuration, and a navigation bar will be provided on the left.
④ Table area of device view
⑤ Module tree
⑥ Module properties: When a module is selected in the module tree, its corresponding parameters will be displayed in the module properties.

You can use the mouse to change the separation between areas in the device view. To change the position of the divider, click between the area dividers and change the spacing by moving the divider to the left or right while keeping the mouse button pressed. The Speedy Splitter (the two small arrow keys) allows you to use a single click to minimize a view, maximize a view or restore the last selected split.

Table area

The table area of the device view gives you an overview of the modules used and the most important technical and organizational data.

14.4.3 Editing devices and networks

14.4.3.1 Adding a device to PROFINET network

When IRT and RT devices are added to the topology view, the system automatically assigns a unique ID to these devices. The ID allocation process will start with the smallest available ID and allocate it to each IRT/RT device in ascending order.

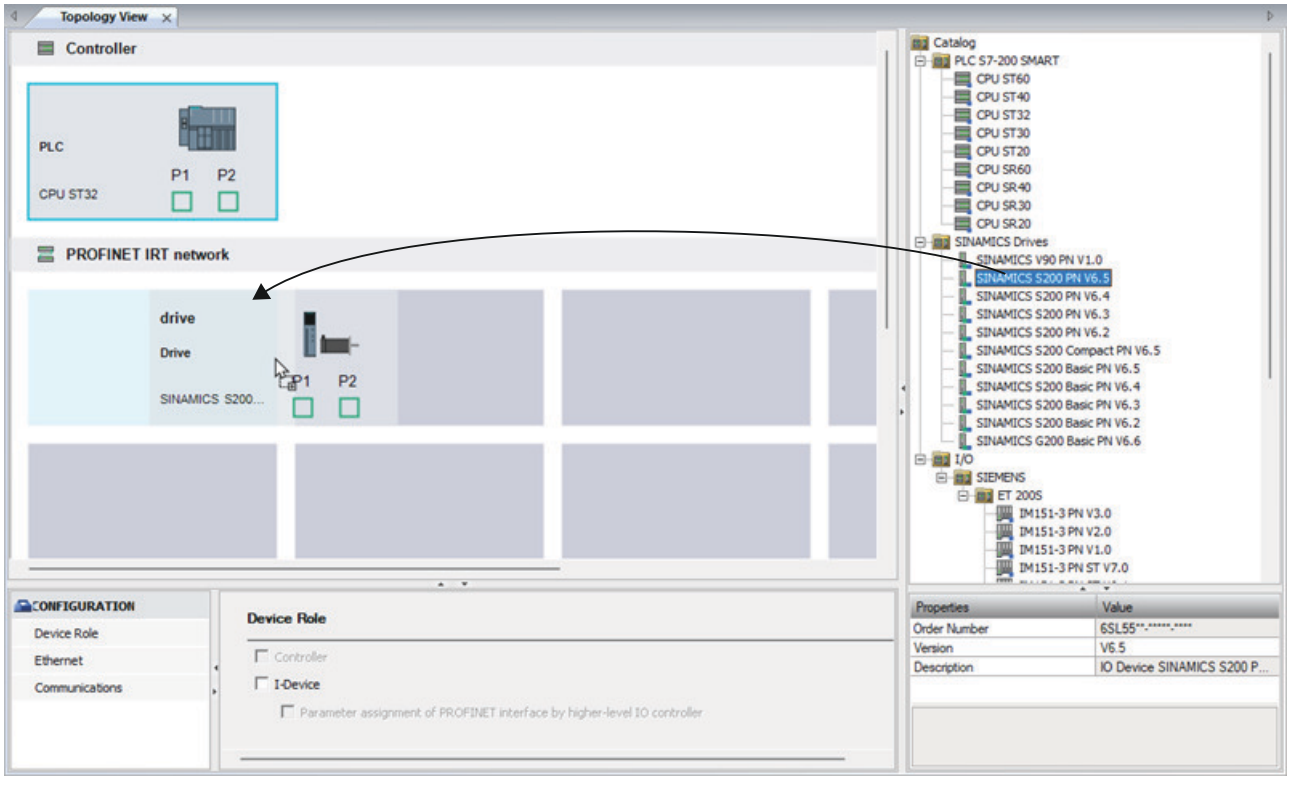
For devices added to the PROFINET network, their names will be automatically generated according to the order in which they were added.

- The drive naming format is drive1, drive2, drive3...
- The I/O module naming format is dev1, dev2, dev3...
- The PLC module naming format is PLC1, PLC2, PLC3...

Dragging and dropping the device

To use the project tree to add a device to the hardware configuration, proceed as follows:

1. Open the hardware network topology.
2. In the device catalog, select the required device.
3. Click on the chosen device to select it.
4. If necessary, set the firmware version through the drop-down list in the device properties window.
5. Drag-and-drop the device into the topology view.

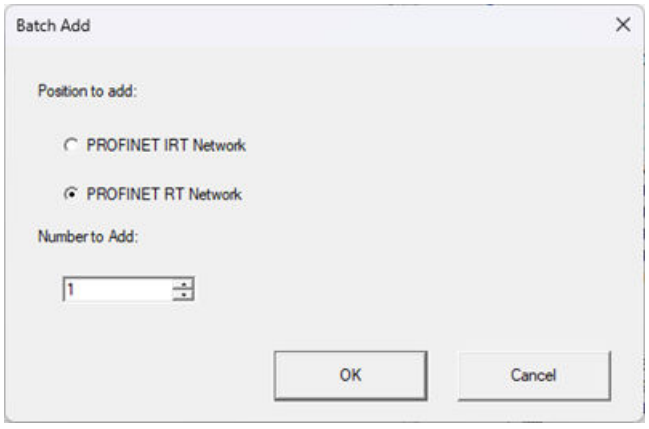


The device's drag-and-drop network depends on the PROFINET type it supports:

- RT device: It can only be dragged to RT networks, not IRT networks.
- IRT device: It can only be dragged to the IRT network.
- Devices that support both RT and IRT: You can drag it to an RT network or an IRT network.

Batch Add

1. In the device tree view, you can add devices in bulk using the "Batch Add" option in the right-click menu.
2. Select the location for adding devices in bulk and the quantity to be added.



3. Click "OK".

PROFINET device variables

When adding PROFINET devices to the configuration of the hardware network topology, STEP 7-Micro/WIN SMART V3 will add PROFINET device variables to the PROFINET device variable table.

14.4.3.2 Go online

This feature allows you to access the Web Server of devices within the hardware network topology.

The "Go online" feature is only available for the following device types configured on the topology page:

- S200
  It can be configured in either the IRT domain or the RT domain.
- G200 Basic
  It can only be configured in the RT domain.

Note

The system supports opening up to ten online pages simultaneously, but each device can only open one at a time.

Workflow

1. After selecting the S200 or G200 Basic device in the topology view, select the "Go online" option in the right-click menu.
2. If a device is detected, the system obtains the device's current IP address:
   - If the IP address is 0.0.0.0, the system will display a dialog box prompting you to configure the device's IP address before proceeding with the next steps.
   - If the IP address is valid, the system will open the device's Web Server.
3. If the target device is not detected to be online, the system will display a message box saying "The current device is not online".

Note

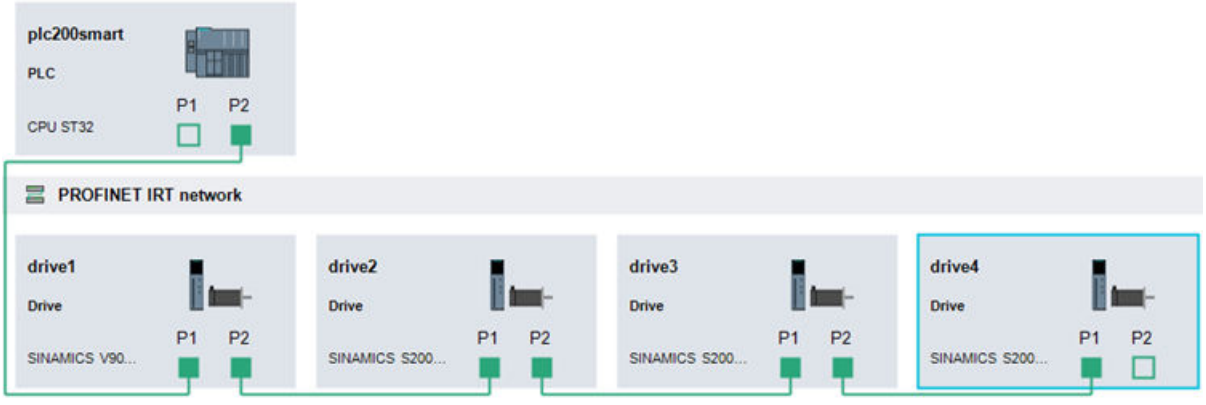
When accessing the device's Web Server for the first time, a security warning page indicating "untrusted" may appear. At this point, you need to manually confirm and select "Continue Access" to enter normally.

It is recommended that after your first access to the Web Server, you follow the pop-up instructions to download, install, and trust the device certificate

14.4.3.3 Device connections

After an IRT device is added to the topology view, connections are automatically established between IRT devices and between IRT devices and the controller. RT devices do not support any form of connection.

The system does not support manually establishing connections between devices.



Controller connections

As a controller, its connection port is flexible and can be connected to the P1 port of the IRT device via either the P1 or P2 port.

The communication port of the PLC device can be adjusted by right-clicking and using "Swap ports" function.

IRT device interconnection

IRT devices use an automatic "hand-in-hand" connection mode. That is, the P2 port of an IRT device will automatically connect to the P1 port of the next adjacent IRT device.

For the SINAMICS S200 PROFINET servo drives, for example, port P1 for the IRT device corresponds to port X150-P1 servo drives and port P2 for the IRT device corresponds to port X150-P 2 servo drives.

14.4.3.4 Copying/pasting/deleting a device

Deleting a device

Remove a device using any of the following steps:

- Right-click the selected device and select "Delete" from the context menu that appears.
- Select the device and press the Delete key.

Note

Deleting a rule

- Controller: Deletion is not allowed.
- IRT device: After confirming deletion, the system will execute the removal operation and automatically reconfigure the daisy-chain connections among the remaining IRT devices to maintain network topology integrity.
- RT device: After confirming deletion, the system will execute the deletion operation, and all devices following the deleted RT device will automatically shift forward one position.

Copying/pasting a device

Hardware devices can be copied in the topology view. The copied devices and modules can be pasted into the IRT network or RT network in the topology view.

To copy a hardware device, follow these steps:

1. Select the hardware device you want to copy in the topology view.
2. Select "Copy" from the shortcut menu.
   If the "Copy" menu item is unavailable, your selection contains at least one component that cannot be copied.
3. Select the position at which the content of the clipboard is to be pasted.
4. Select "Paste" from the shortcut menu.
   If the "Paste" menu item is unavailable, the clipboard is empty or contains at least one component that cannot be pasted at this point.

Note

Naming rules

- If the name of the copied device ends with a number (e.g., Device1), the pasted device will automatically be named with that number incremented (e.g., Device2). If the incremented name already exists, the system will continue to increment (n+2, n+3, etc.) until a unique name is found.
- If the name of the device being copied does not end with a number (e.g., Sensor), the pasted device name will be the original name followed by the number "1" (e.g., Sensor1).

14.4.3.5 Moving/swapping/inserting a device

You can select a device in the topology view and then drag the mouse to move, swap, or insert the device.

Rule

- Controller: The controller device must not be moved or swapped.
- RT device: Devices in the RT network layer do not have strict sequential dependencies. Therefore, moving, swapping, or inserting devices within the RT layer has no practical significance, and the system does not allow such operations between RT devices.
- When performing a move or insert operation, the target position must support the type of the source device. Otherwise, the operation will fail.

Note

When cross-layer movement or insertion operations may fail due to missing telegram information or other reasons, please follow the system prompts for assistance.

Moving a device

Note

When a device moves between different network layers, the system will automatically clear its configuration information.

Proceed as follows to move the device:

1. Select the hardware device you want to move.
2. Drag and drop the device to your desired position.

The selected device will be moved to the target position.

Swapping a device

Note

This operation only supports swapping positions between hardware devices within the IRT network layer.

Proceed as follows to swap a device:

1. Select the device you want to move.
2. Drag and drop the device onto the device you want to swap it with.

The device positions are swapped.

Insert a device

Note

This operation only supports inserting a device between devices within the IRT network layer.

When the target position is between two devices, dragging and dropping a device will trigger an insertion operation:

1. Select the device you want to insert.
2. Drag and drop the device to the target position.

The device is inserted at the target position.

14.4.4 Configuring the device

14.4.4.1 Configuring a controller

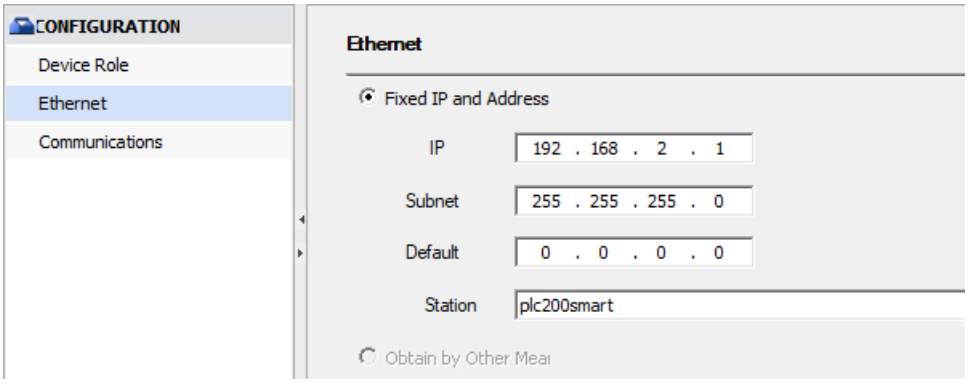
To change the properties and parameters of a controller in the topology view, follow these steps:

1. In the topology view, select PLC in the controller layer.
2. Click "Device Role" in the configuration window.

Note

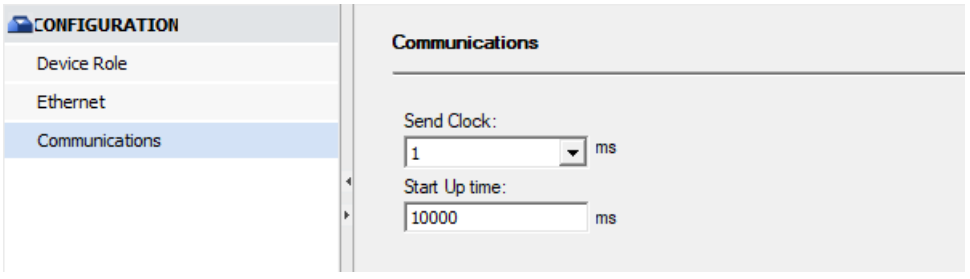
When configuring the CPU as a controller, please ensure that the GSDML file for the PROFINET device has been imported.

3. Click "Ethernet" in the configuration window.



Ether-net	• When acting as a controller: A static IP address must be configured. • When acting as an I-device: Supports fixed IP or dynamic allocation
Station Name	If you change the station name, the device name will be updated in the device view.

4. In the configuration window, click "Communication" and assign "Send Clock" and "Start Up time" to the controller.



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:

– 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" expires, the CPU switches to RUN mode at the end of the "Start Up time".

Note

If the current project's "Send clock" is set to 1ms, when you add an IRT device with a minimum send clock of 2 ms, the system will automatically adjust the "Send clock" to 2 ms and provide a corresponding prompt.

14.4.4.2 Configuring IRT devices

After inserting a device into the hardware network topology, you can edit the device's default properties in the topology view.

To change device properties and parameters in the topology view, follow these steps:

1. In the topology view, select the device in the IRT layer.
2. Click "Device Information" in the configuration window.

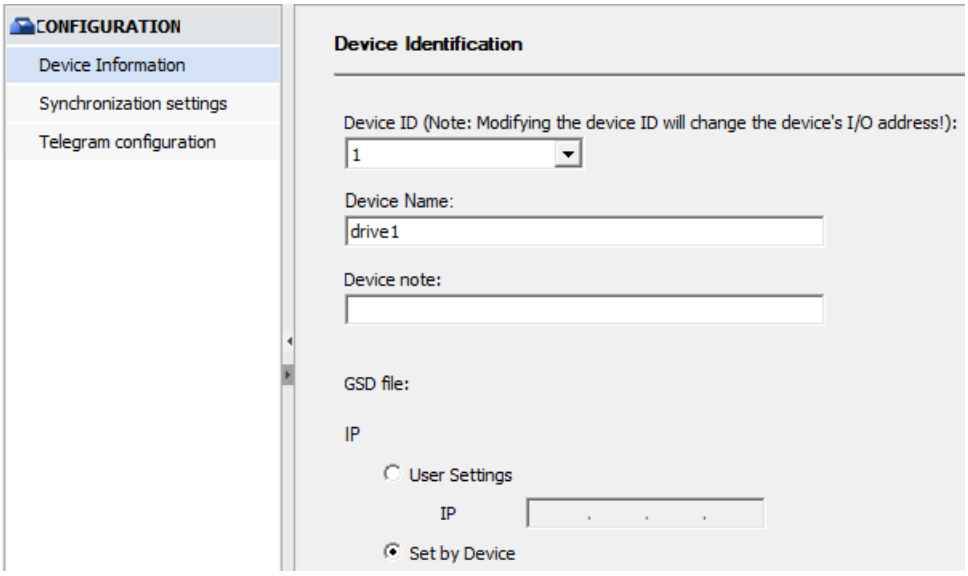
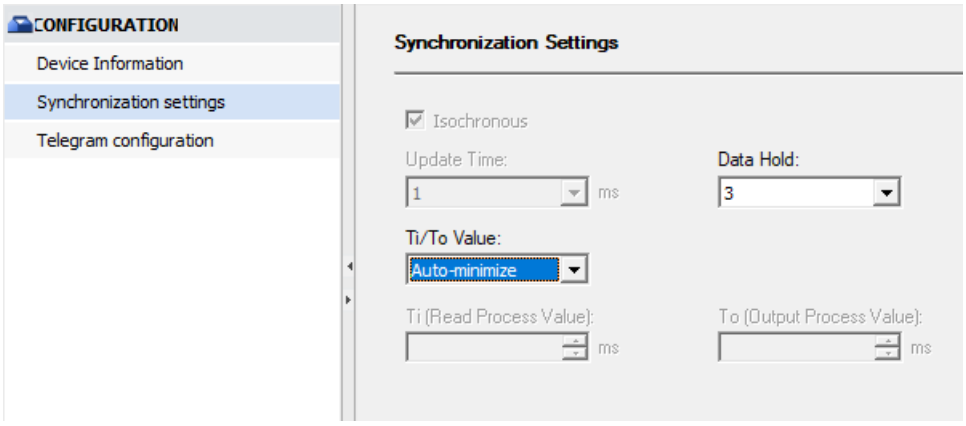


Table header	Meaning
Device ID	When the module connects to the PLC via PROFINET, the PROFINET device number is used.
Device Name	The device name must match the name of the inserted module. Example: • If a module is inserted into a front-end module named "IODevice1", then it must be set to "IODevice1" here.
Device Note	Add comments to the device.
IP	There are two methods for configuring IP addresses: • User settings: The controller assigns IP addresses to the devices. • Fixed IP

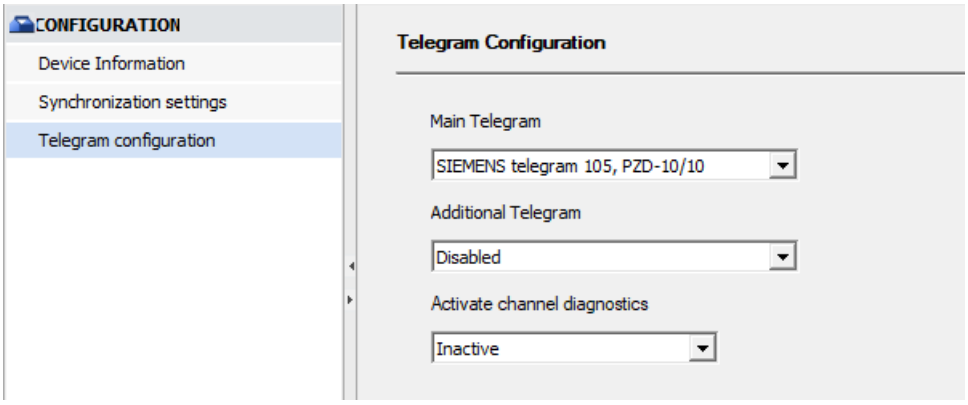
3. Click "Synchronization Settings" in the configuration window.



Isochronous

Update times	The "Send Clock" setting is configured in the "Communications" section of the controller.
Data hold	The data hold time can be set to "n times the update time". The value range is 3-255, with a default value of 3.
Ti/To value	The Ti/To value can be set in the following two ways: • Automatic minimization: Automatically calculate and use the minimum value of Ti/To. • Manually: Ti and To can only be edited when "manual" setting is selected.

4. In the configuration window, click "Telegram Settings" to select or modify the configuration telegram.



- Main Telegram: Telegram 105/5 is optional; the default telegram number is 105.
- **Additional Telegram:** Telegram 750 can be selected or disabled; it is disabled by default.
- Active diagnostics: If the device's GSD file contains the "Active diagnostics" parameter, the diagnostic function will be displayed; otherwise, it will not be displayed.

Communication times T_i, T_o, T_{DC}

When calculating the following error, the transmission times of the setpoint to the drive and of the actual position value to the controller are deducted. The transmission times result from the following communication times:

- T_i: Time to import process values
- T_o: Time to export process values
- T_{DC}: Send clock of the PROFINET interface

The following error is calculated from the delayed position setpoint by the communication times $T_i + T_o + T_{DC}$ and the motion control cycle T_{Servo} minus the actual position in the controller.

Telegram types

Telegram	Brief instructions
Standard telegrams	
5	• Control words STW1 and STW2, status words ZSW1 and ZSW2 • Speed setpoint 32 bit (NSET), actual speed value 32 bit (NACT) • Actual encoder value 1 (G1_XIST1, G1_XIST2) (motor encoder) • Dynamic Servo Control (DSC)[1)] – Velocity precontrol value – Position difference (XERR) – Kv factor gain position control (KPC)
Siemens telegrams (with torque limiting)	
105	• Control words STW1 and STW2, status words ZSW1 and ZSW2 • Speed setpoint 32 bit (NSET), actual speed value 32 bit (NACT) • Actual encoder value 1 (G1_XIST1, G1_XIST2) (motor encoder) • Dynamic Servo Control (DSC)[1)] – Velocity precontrol value – Position difference (XERR) – Kv factor gain position control (KPC) • Torque limiting

1) For use of Dynamic Servo Control (DSC), the motor encoder (first encoder in the telegram) of the drive must be used as the first encoder for the axis.

14.4.4.3 Configuring RT device

After inserting a device into the hardware network topology, you can edit the device's default properties in the topology view.

To change device properties and parameters in the topology view, follow these steps:

1. In the Topology View, select the device in the RT layer.
2. Click "Device Information" in the configuration window.

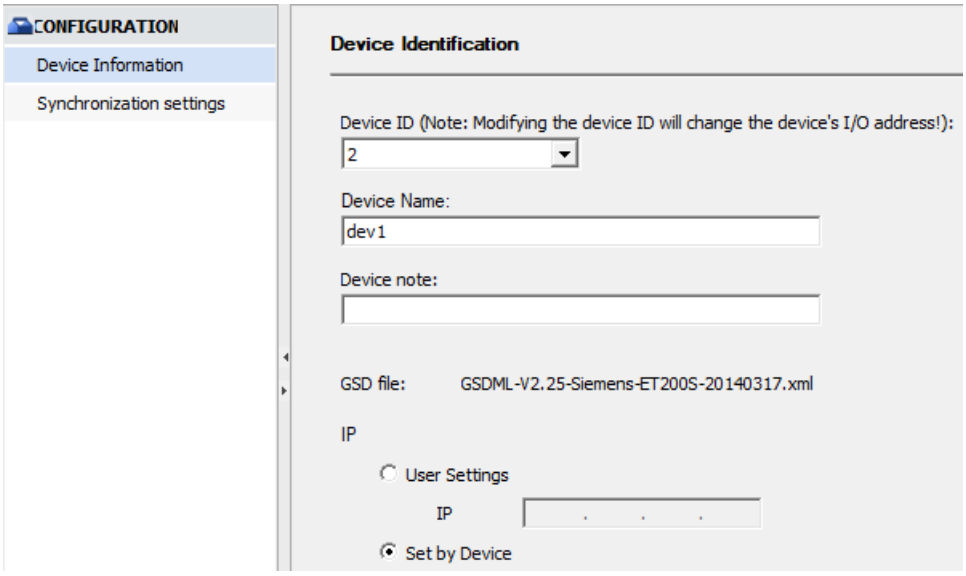
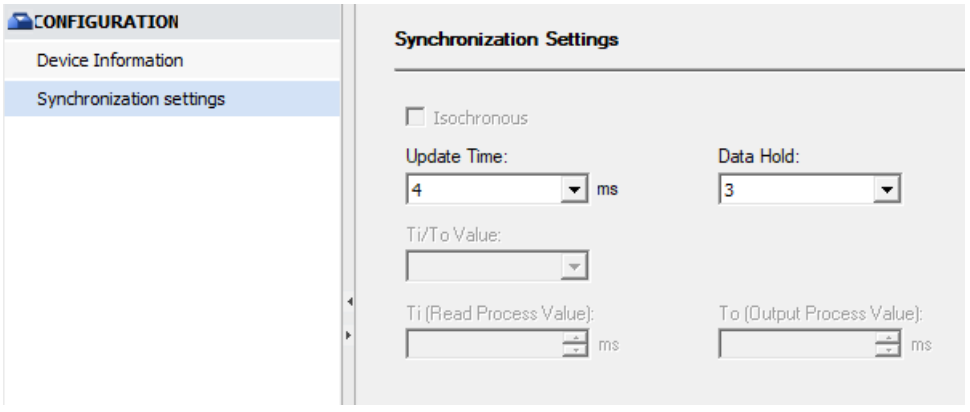


Table header	Meaning
device id	When the module connects to the PLC via PROFINET, the PROFINET device number is used.
Device name	The device name must match the name of the inserted module. Example: • If a module is inserted into a front-end module named "IODevice1", then it must be set to "IODevice1" here.
Device Notes	Add comments to the device.
IP Address	There are two methods for configuring IP addresses: • User settings: The controller assigns IP addresses to the devices. • Fixed IP

3. Click "Synchronization Settings" in the configuration window.



Update times: The data exchange cycle.
Data hold: The default value is 3.

14.4.4.4 Add/remove/move RT modules

Introduction

Once you have added devices from the hardware catalog to your configuration in Topology View, you can add modules to the devices. The graphic area and table area in the device view are linked. When you make changes in one region, the other region will automatically refresh to ensure that the data and configuration remain consistent.

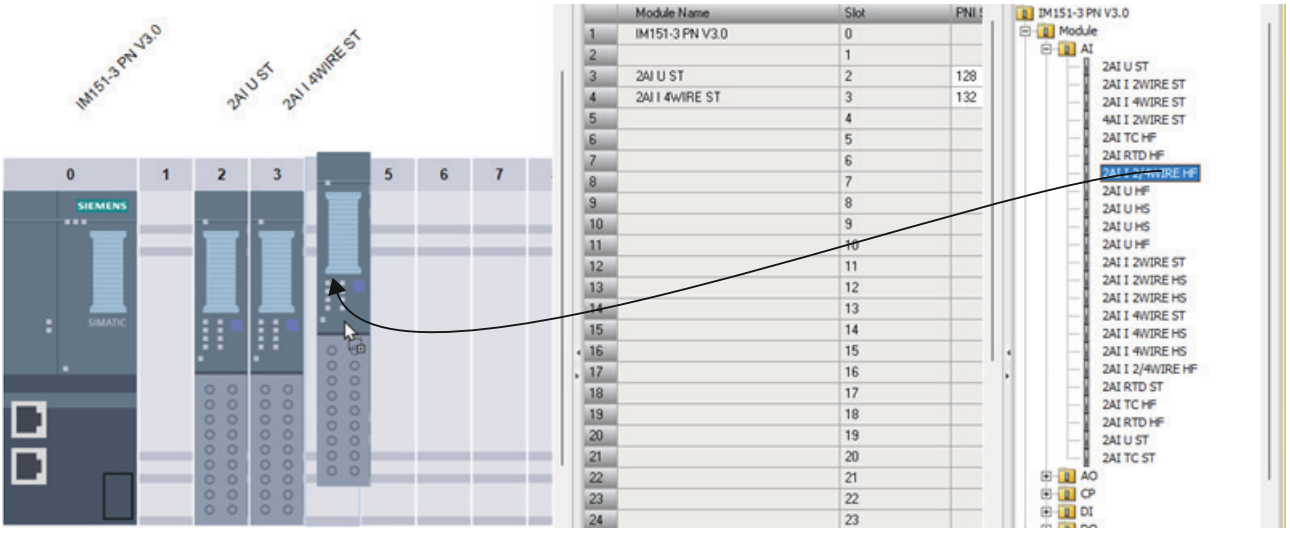
Note

No other modules may be dragged into or moved to SLOT-0.

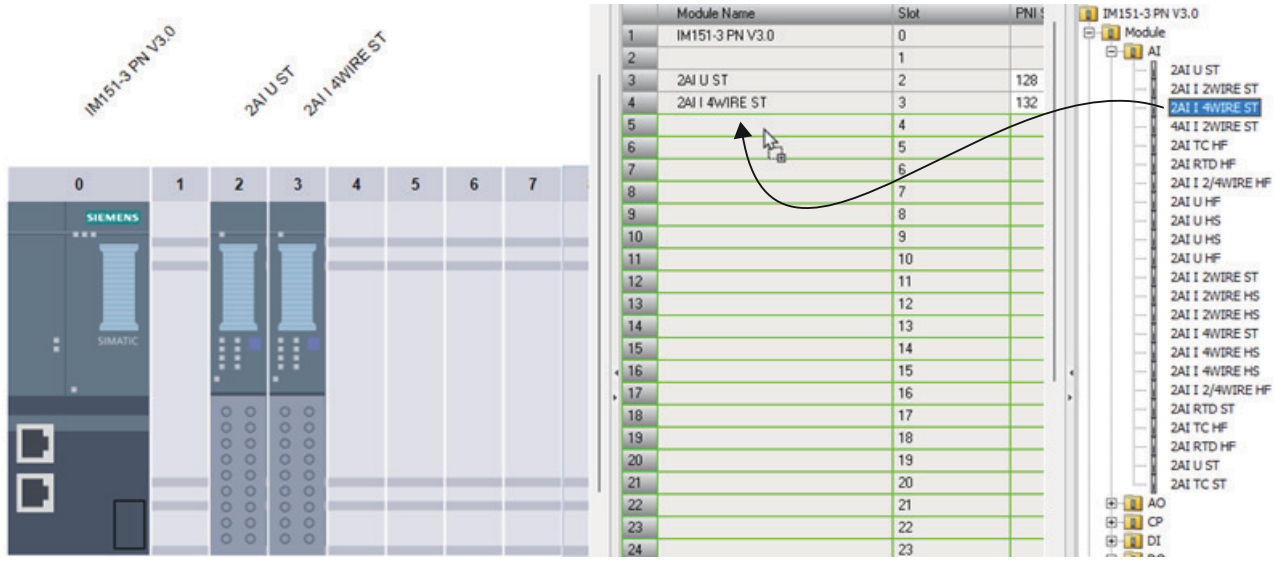
Adding a module

To add a module from the hardware catalog, follow these steps:

1. In the hardware catalog, select the required module.
2. If necessary, set the firmware version using the drop-down list in the hardware catalog.
3. Use drag-and-drop to move the module from the hardware catalog to an available valid slot in the graphic or table area:
   - Drag the module to an empty slot in the graphic area.



   - Drag the module to an empty slot in the table area.



After adding a module, STEP 7-Micro/WIN SMART V3 will add PROFINET device variables to the PROFINET device variable table.

Deleting modules

Delete the module using any of the following methods:

- Right-click the module in the graph or table area and select "Delete" from the context menu.
- Select the module in the graph or table area and press the Delete key.

Moving modules

To move a module, follow these steps:

1. Select the module you want to move.
2. Drag and drop the module to the location you want to move it to.

The selected module will be moved to the target location.

14.4.5 Topology Comparison

Topology comparison is used to compare the configuration of the IRT topology view in the current project with the network topology information of the actual port connected to the PLC. This feature allows you to promptly identify and locate any connection relation inconsistencies between the actual network and the configuration.

Note

If the IRT device is not configured in the topology view, the comparison function will be disabled.

Precondition

To ensure correct scanning, the device must meet the following conditions:

- Connected to the controller.
- The IP address has been initialized.

Operations

To find the differences between the configured and the actual topology, follow these steps:

1. In the toolbar, click the "Compare" button  .
2. Click the "Scan Topology" button  to scan for connected devices and configurations.

Note

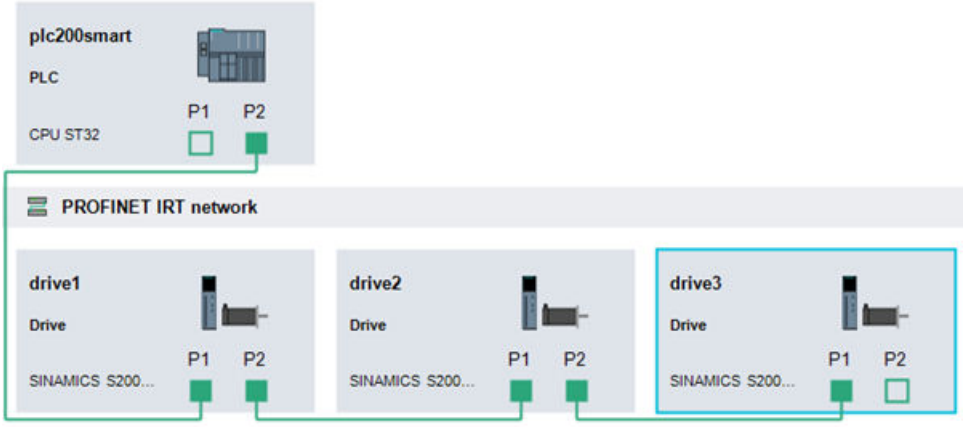
If any changes are made to device configuration or connections, click the "Scan Topology" button to ensure the system accurately reflects the latest updates.

Results

Once the scan is complete, the system will display the update time and detailed topology comparison information.

Special cases

When performing topology comparisons, since the controller cannot directly query the complete network connection relations of devices, the system will attempt to infer the actual connections of IRT devices with IP addresses configured as 0.0.0.0 by analyzing the network connections of their neighboring devices. The following will describe several special cases based on the connection type between the device and the controller (assuming the physical connection is as shown in the diagram below):



- When a device with the IP address of 0.0.0.0 is directly connected to the controller (e.g., Drive1):
  - The partnership of P2 on the PLC200Smart corresponding to the device with IP address 0.0.0.0 will be displayed as empty.
  - The partnership of P1 on the controller corresponding to Drive1 will also be displayed as empty.
  - The P2 port can still be correctly identified and displayed.
- When a device with the IP address of 0.0.0.0 is not directly connected to the controller (e.g., Drive2 or Drive3): It can correctly infer the network connection relationship of the 0.0.0.0 device and correctly display the partnership and connection status of all ports.

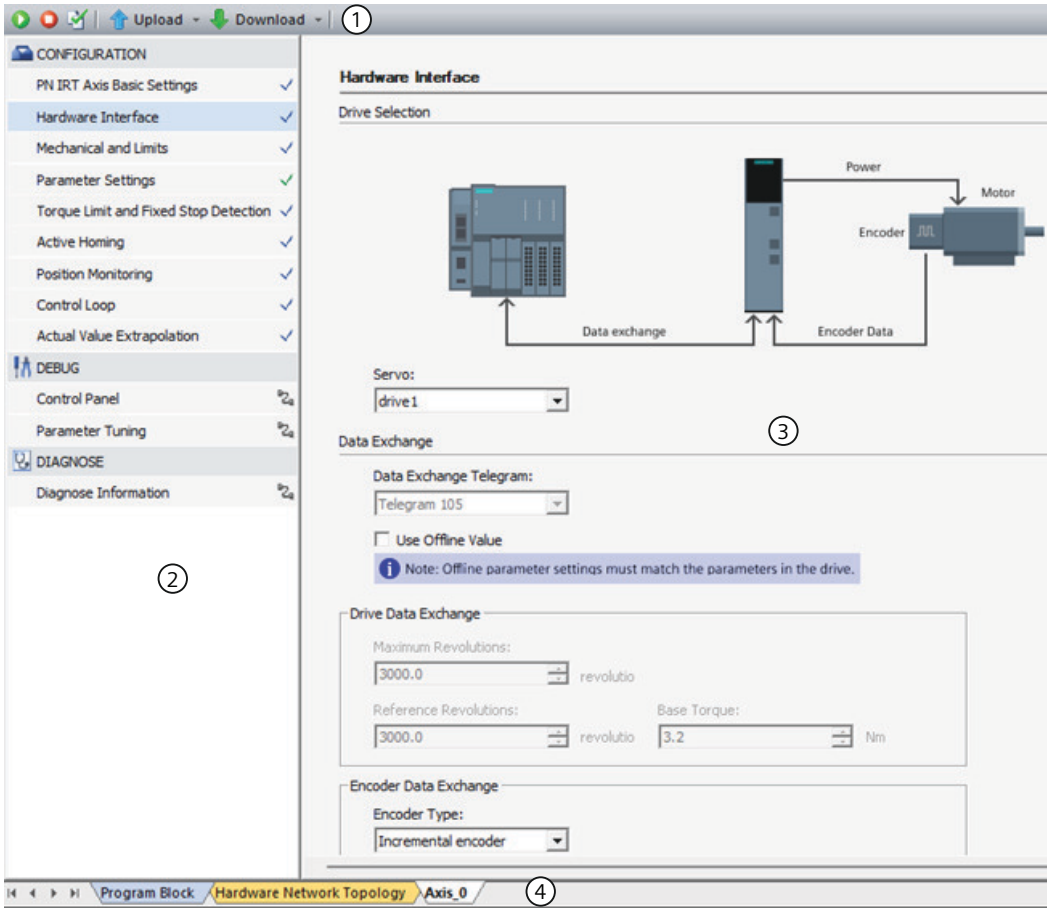
14.4.6 Motion control component configuration window

Use one of the following methods to open a configuration window after you added an axis:

- Open the Motion control folder in the project tree by clicking the branch expansion icon or double-clicking the motion control folder Motion Control icon. Then double-click the component you want to configure, or you can select the component and press Enter.
- Double-click the component table at the bottom of the motion control window.

Axis

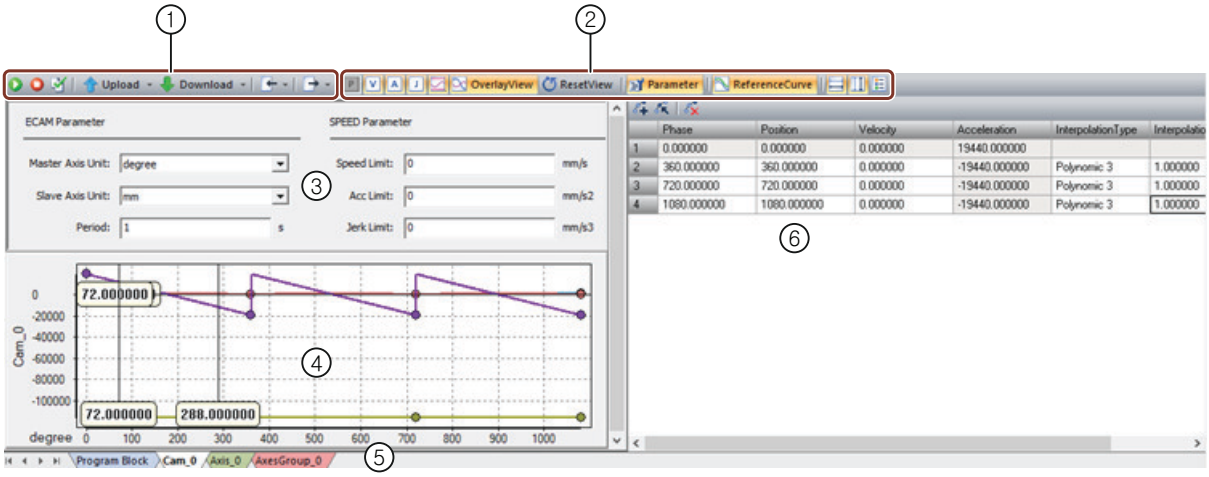
The Axis editor window includes the following components:



① Toolbar: buttons for commonly used operations
② Directory of configuration, debug and diagnose
③ Configuration/debug/diagnose page
④ Component selector: allows you to switch between configuration page of different motion control component.

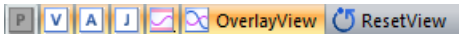
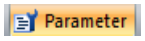
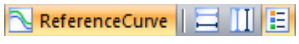
Cam

The Cam editor window includes the following components:



① Toolbar: buttons for commonly used operations
② Curve toolbar
③ Parameter configuration panel
④ position/speed/acc/jerk curve page
⑤ Component selector: allows you to switch between configuration page of different motion control component.
⑥ Key-point configuration table

Cam toolbar

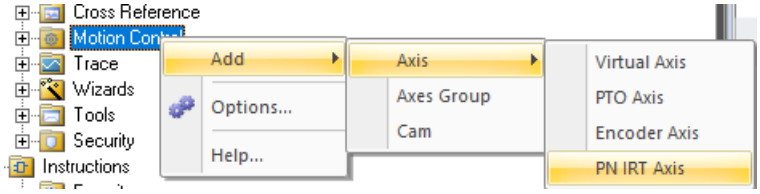
	Change CPU operating mode to RUN or STOP Compiling programs
	Upload and download transfers
	Import (Page 870) and Export (Page 870) Cam data table/configuration table
	Graph control tools (Page 869)
	Show or hide the parameter setting panel (Page 868)
	Curve control tools (Page 869)

14.4.7 Configure the axis

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.



Note

Currently, only ST32 CPUs with the firmware version of 3.2 support adding PN IRT axes.

Once the axis is added, the configuration page will display its default configuration and automatically generate the corresponding system motion control variable table.

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.

Note

The PN IRT axis does not currently support configuration as an axis group.

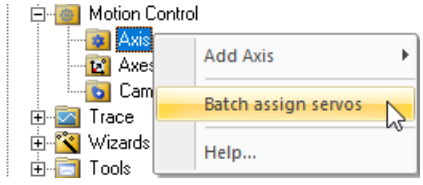
Batch adding IRT axes

Precondition:

- You have successfully added at least one IRT device to the hardware network topology view in your project.
- In the current project configuration, the number of available IRT axes does not exceed the system limit (Page 768).

Through this function, you can automatically create IRT axes in batches for IRT devices that have already been added to the hardware network topology.

1. Double-click to open the "Motion control" folder in the project tree.
2. Right-click on the "Axes" folder, and select "Batch add servos" from the context menu that appears.

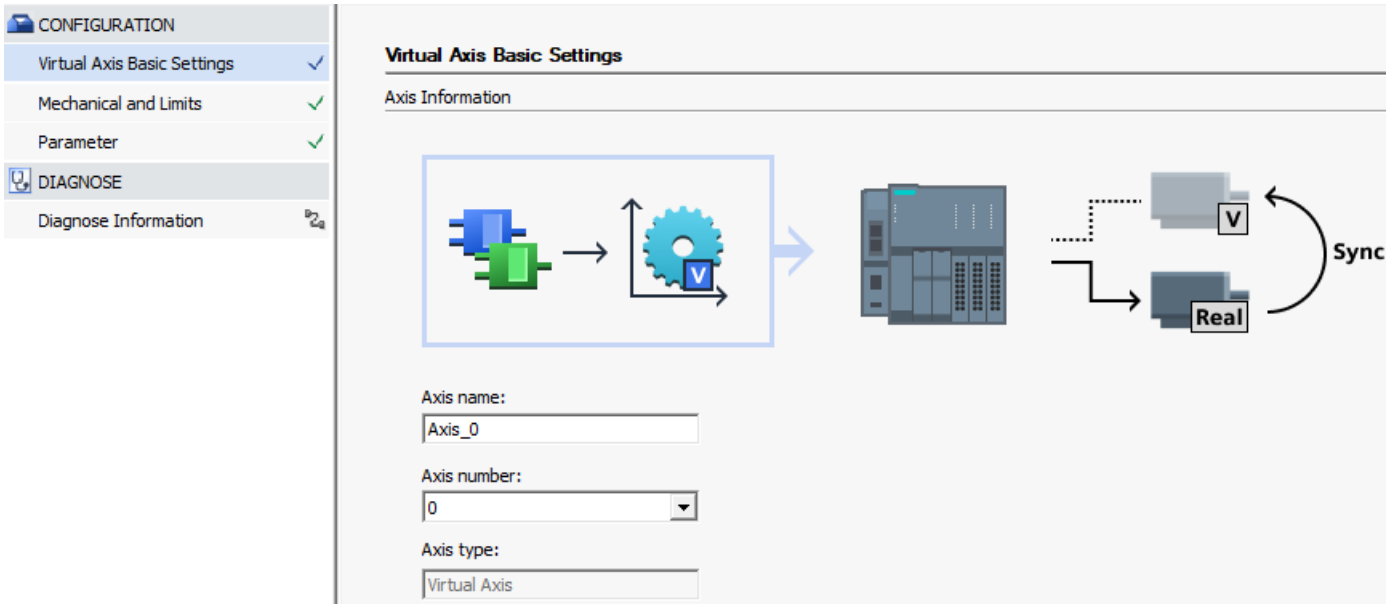


14.4.7.1 Configuring the Virtual Axis

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.



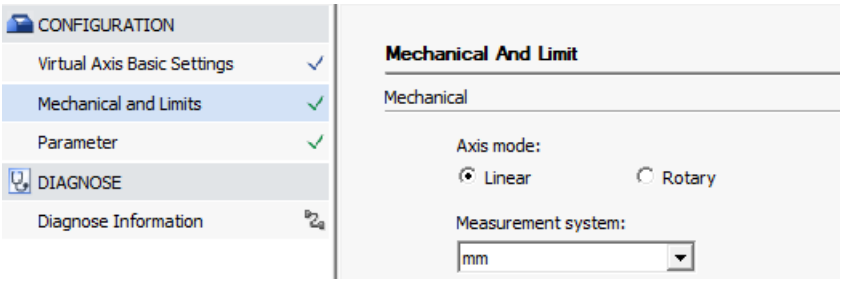
Note

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 must be a letter, Chinese character, 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.

Mechanical and Limits

Mechanical

When you click the "Mechanical and Limits" in the "CONFIGURATION" panel, the following dialog displays:



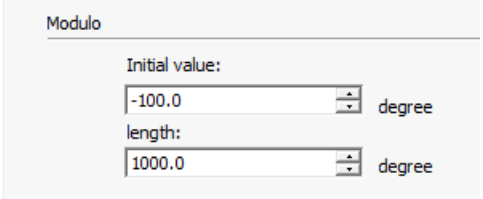
Configuration item	Mode	Description
Axis mode	Linear	In this mode, the axis can move along a straight line, typically guided by linear rails or tracks.
	Rotate	In this mode, the axis can rotate around a central point or axis of rotation.
Measurement system	pulses cm, mm, inch, or degree	All speeds in controlling the motion of the axis are expressed in units of "Measurement system" per second.

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 500.0.



Software Limit

When the software limit function is enabled, the axis will only initiate decelerated stop after exceeding the set limit position. This means the final stopping position of the axis will exceed the configured limit.

The Software limit is disabled by default. When the input is disabled, all other related parameters are disabled. When the "Enable software limit" checkbox is selected, you can access the "Software negative limit position", "Software positive limit position", and "Response" parameters.



Note

The software limit function only supports linear axes. When the axis type is configured as a rotary axis, the relevant software limit configuration options will be automatically hidden.

Note

If Homed=0 in TO, the software limit will not take effect.

Software negative/positive limit

You can define the position at which the axis stops moving.

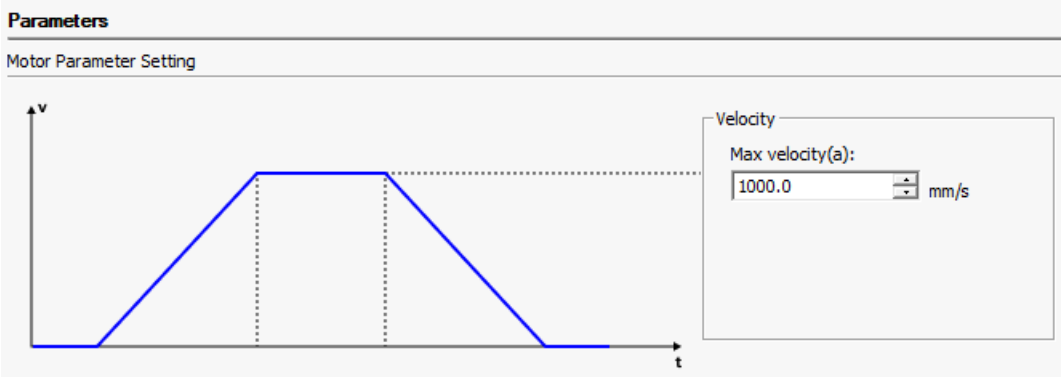
Response

The deceleration rate for "Decelerated Stop" is defined by the "Max deceleration" in Parameters (Page 818).

Parameters

Motor Speed

In the Motor Speeds dialog, you can define the maximum velocity for your application.

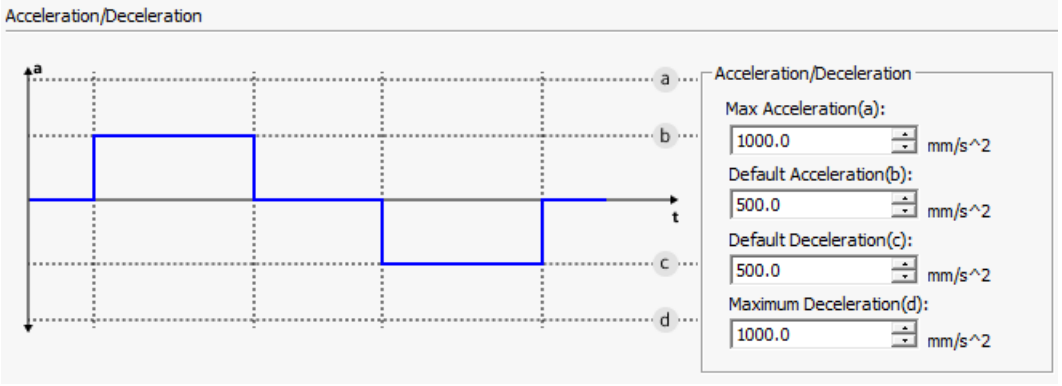


Max Velocity

- 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 816); the default value is 1000.0 mm/s or 36000.0 deg/s.

Acceleration/Deceleration

In the Acceleration/Deceleration dialog, you can define the acceleration speed and deceleration speed for your application.

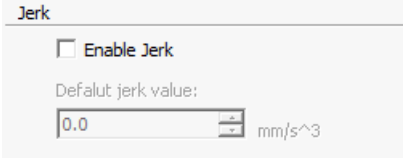


- **Max acceleration**: used to limit the maximum acceleration for user-configured motion instructions.
- **Default acceleration**: the default value for the acceleration parameter in Motion Control Instructions (Page 906).

- **Default deceleration**: the default value for the deceleration parameter in Motion Control Instructions (Page 906).
- **Max deceleration**: used to limit the maximum deceleration for user-configured motion instructions.
  And
  – For virtual axis, the maximum value used in Decelerated Stop for software limits. (Page 816)

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.

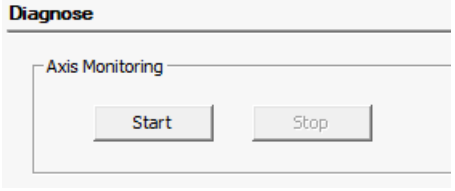


Diagnose Information

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 state

The status of the axis is shown in the dialog. The details would be displayed in "ErrorCode (Page 1015)", "Position", and "Velocity" dialogs.

Axis state	Description
Error	An axis error occurred. Refer to the ErrorID and ErrorInfo parameters in the Motion Control instructions for details
Disable	The axis is disabled and cannot be controlled via Motion Control instructions
Standby	The axis is at a standstill state

Axis state	Description
Discrete motion	The axis is motion in Discrete motion state
Continuous	The axis is motion in Continuous motion state
Synchronous motion	The axis is motion in Synchronized motion state
Homed	The axis has returned to its home position
Stopping	The axis is Stopping
ErrorStop	The axis is experiencing an error condition and is currently in the ErrorStop state
Software positive limit switch	The software positive limit switch was reached or exceeded
Software negative limit switch	The software negative limit switch was reached or exceeded

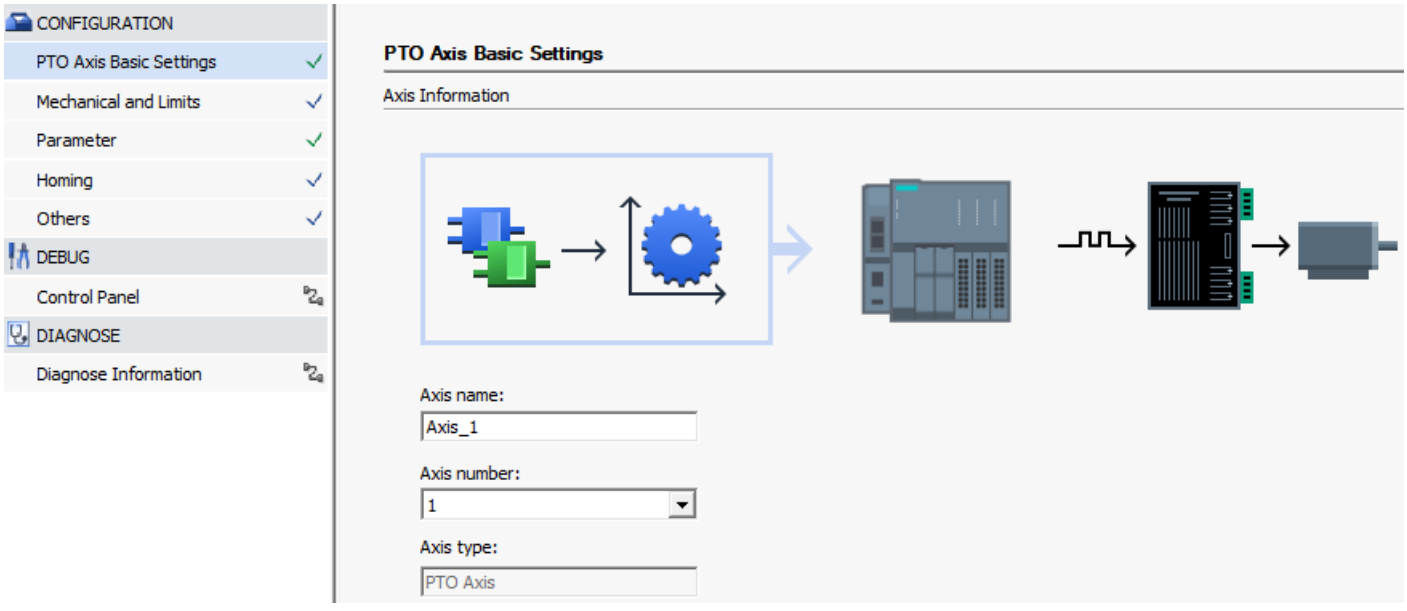
14.4.7.2 Configuring the PTO Axis

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.

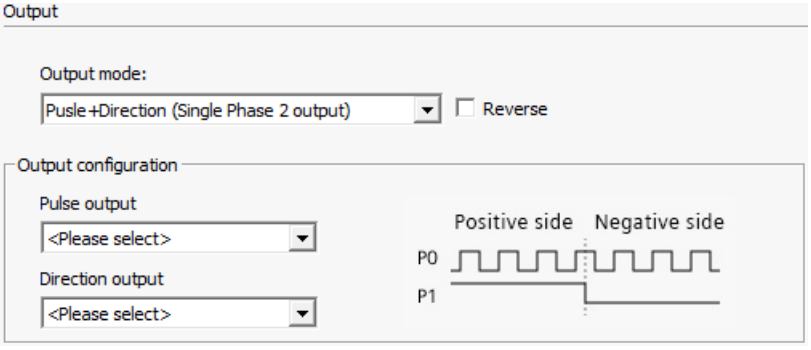


Note

- 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 must be a letter, Chinese character, 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.

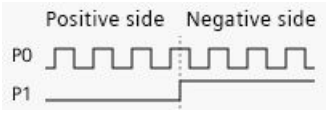
Outputs

You can configure the pulse and direction output interfaces of the stepper/servo driver.

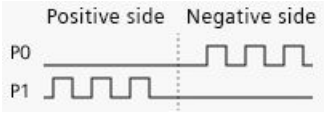


There are three options for the "Output mode" interface to the stepper/servo drive as follows:

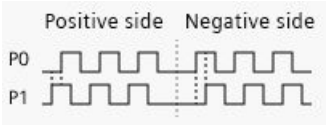
- Pulse+Direction (Single Phase 2 output): If you select the Pulse+Direction (Single Phase 2 output) option, then one output (Pulse) controls the pulsing, and one output (Direction) controls the pulse direction. Direction is high (active) if pulsing is in the positive direction. Direction is low (inactive) 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

Once a Q point is configured and assigned to a Pulse Sequence Output (PTO) axis function, that Q point cannot be forced again.

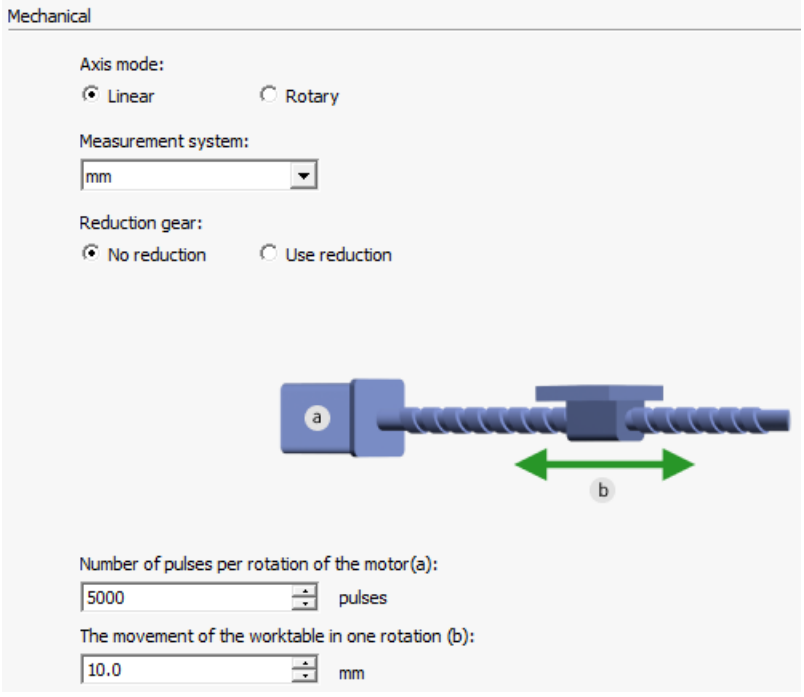
Note

The default setting for "Output mode" is "Pluse+Direction (Single Phase 2 output)" and the "Reverse" checkbox is unseleceted.

Mechanical and Limits

Mechanical

When you click the "Mechanical and Limits" in the "CONFIGURATION" panel, the following dialog displays:



Configuration item	Mode	Description
Axis mode	Linear	In this mode, the axis can move along a straight line, typically guided by linear rails or tracks.
	Rotate	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".

Configuration item	Mode	Description
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.

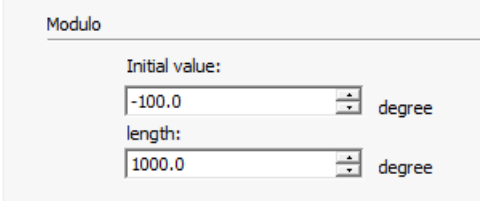
[1] **Note:** The range of the "Number of pulses per rotation of the motor/encoder" parameter is 1 to 9,999,999 pulses. The default value for this parameter is 5,000.

Modulo

The "Modulo" configurations can only be applied if you select "Rotary" for the **Axis mode**.

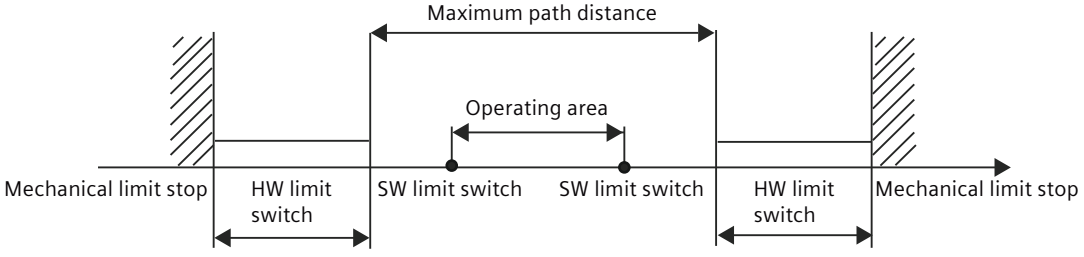
"Initial value" defines from which degree you want the axis to start moving, and "Length" defines how much degrees you want the axis to move. The module range is left-closed and right-open. For example, in the following example, the range of the modulo is [-100, 900).

- 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 500.0.



Hardware limits

Software/hardware limit switches ensure that axes operate within the workbench's effective range. If an axis malfunctions and exceeds the limit switch, the axis will stop running and trigger an error upon reaching either limit.



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 item	Description
Input	You can assign LMT-/LMT+ to an input on 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 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 active level. The default level is "High".
Response	In this parameter, you can assign what should happen if the LMT-/LMT+ input goes active.

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 device.

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 "Enable software limit" checkbox is selected, you can access the "Software negative limit position", "Software positive limit position", and "Response" parameters.



Note

If MC_Home is not executed after power-on, the soft limits will not take effect.

Software negative/positive limit

You can define the position at which the axis stops moving.

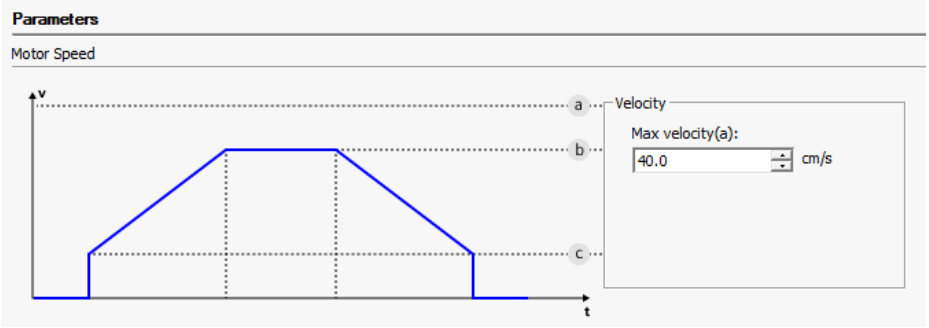
Response

The deceleration rate for "Decelerated Stop" is defined by the "Max acceleration/deceleration" in Parameters (Page 826).

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 for velocity is configured in the Measuring system; the velocity cannot be lower than the start/stop velocity of STEP 7 Micro/WIN SMART V3 motion control.

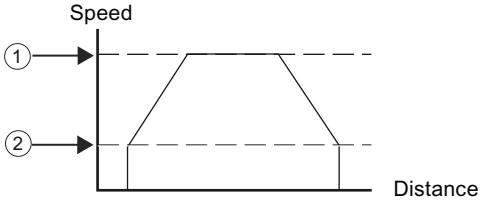
> **WARNING**
>
> **Set an appropriate max velocity**
>
> Default max velocity for linear axes is 400.0 mm/s and default max velocity for modal axes is 400.0 deg/s.
>
> Be sure to adjust the maximum velocity based on actual application scenarios to avoid equipment damage or personal injury.

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.

$$\text{CPU pulse} = \frac{\text{Number of pulses per rotation of the motor(a)} * \text{Gear numerator (c)}}{\text{The movement of the work table in one rotation (b)} * \text{Gear denominator (d)}} * \text{move distance}$$

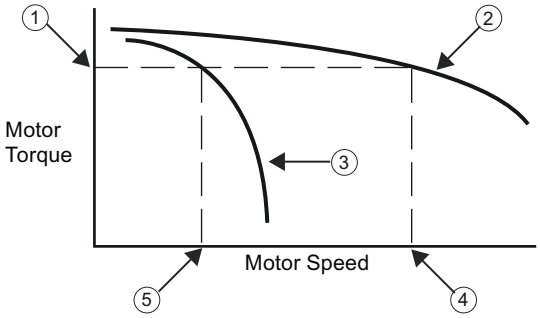
The maximum velocity and the start/stop velocity define the ramping characteristics of the Axis of Motion as shown in the following figure:



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.

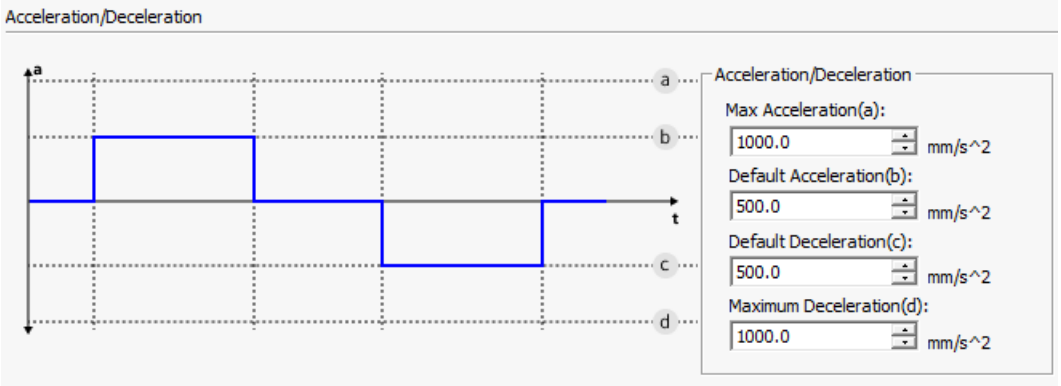
To help you select the correct speeds for your application, refer to your motor data sheet. The figure below shows a typical motor torque/speed curve:



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

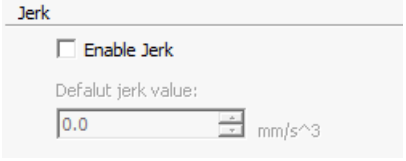
In the Acceleration/Deceleration dialog, you can define the acceleration speed and deceleration speed for your application.



- **Max acceleration**: used to limit the maximum acceleration for user-configured motion instructions.
- **Default acceleration**: the default value for the acceleration parameter in Motion Control Instructions (Page 906).
- **Default deceleration**: the default value for the deceleration parameter in Motion Control Instructions (Page 906).
- **Max deceleration**: used to limit the maximum deceleration for user-configured motion instructions.
  - For PTO axis, the maximum value used in Decelerated Stop for hardware limits/software limits, and Decelerated Stop for STP function in the Others (Page 831) configuration. Motor acceleration and deceleration settings are determined by trials and errors. You should start by entering a large 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.

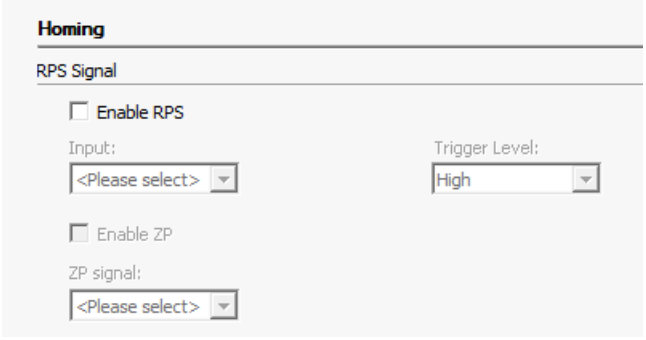


Homing

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), this 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

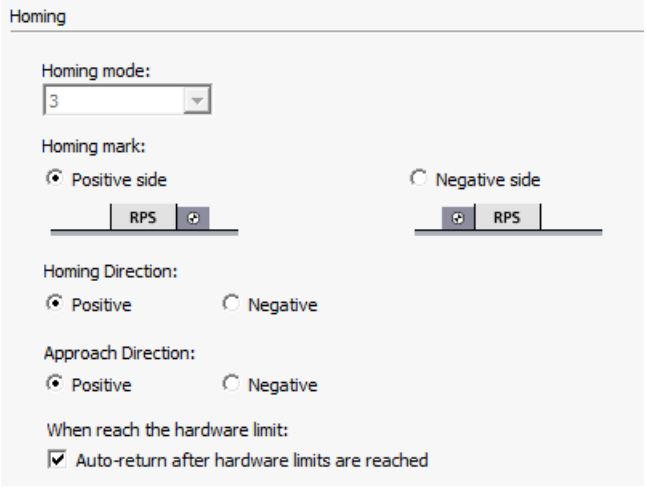


Figure 14-1 Homing of PTO axes

Homing mark

Select whether to use the positive or negative side as the origin.

Approach Home 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.

When reach the hardware limit

If this "Auto-return after hardware limits are reached" checkbox is selected, the axis is allowed to move in the reversed direction to the homing point.

Velocity

- Approach Home velocity: Define Approach Home velocity
- Homing start velocity: Define Homing start velocity

Note

The limit for the input value: Start/stop velocity ≤ Approach Home velocity ≤ Homing velocity ≤ Max velocity

Home 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 offset", the axis is at the home position that was specified in input parameter "Position" of the "MC_Home" Motion Control instruction.

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

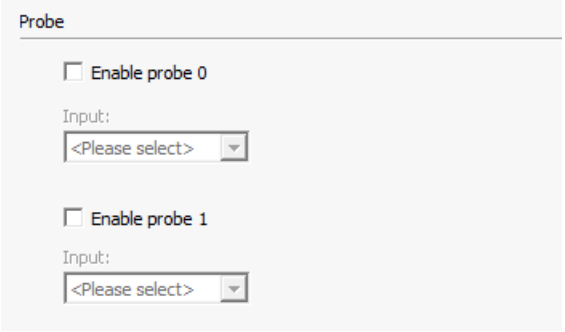
In this parameter, you can assign the trigger of the STP input. The default setting is "Level". You can select between the following two options:

- 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.

Touch Probe

You can enable the probe control by selecting the "Enable probe 0" and "Enable probe 1".



Input

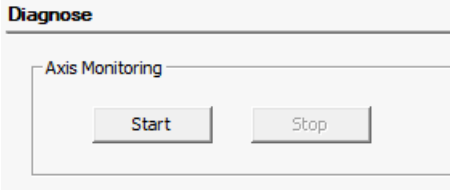
Select the input for the touch probe. You can assign any of the Inputs from I0.0 through I1.7 of the CPU.
You must configure the Probe before using the MC_TouchProbe instruction.

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 state

The status of the axis is shown in the dialog. The details would be displayed in "ErrorCode (Page 1015)", "Position", and "Velocity" dialogs.

Axis state	Description
Error	An axis error occurred. Refer to the ErrorID and ErrorInfo parameters in the Motion Control instructions for details
Disable	The axis is disabled and cannot be controlled via Motion Control
Standby	The axis is at a standstill state
Discrete motion	The axis is moving in Discrete state
Continuous	The axis is moving in Continuous state

Axis state	Description
Synchronous motion	The axis is moving in Synchronized state
Homed	The axis has returned to its home position
Stopping	The axis is Stopping
ErrorStop	The axis is experiencing an error condition and is currently in the ErrorStop state
Hardware positive limit switch	The hardware positive limit switch is reached or exceeded
Hardware negative limit switch	The hardware negative limit switch is reached or exceeded
Software positive limit switch	The software positive limit switch is reached or exceeded
Software negative limit switch	The software negative limit switch is reached or exceeded
Reference switch	The RPS signal is triggered when the axis is homing
Emergency stop switch	The STP signal is triggered

14.4.7.3 Configuring the Encoder Axis

Encoder Axis Basic Settings

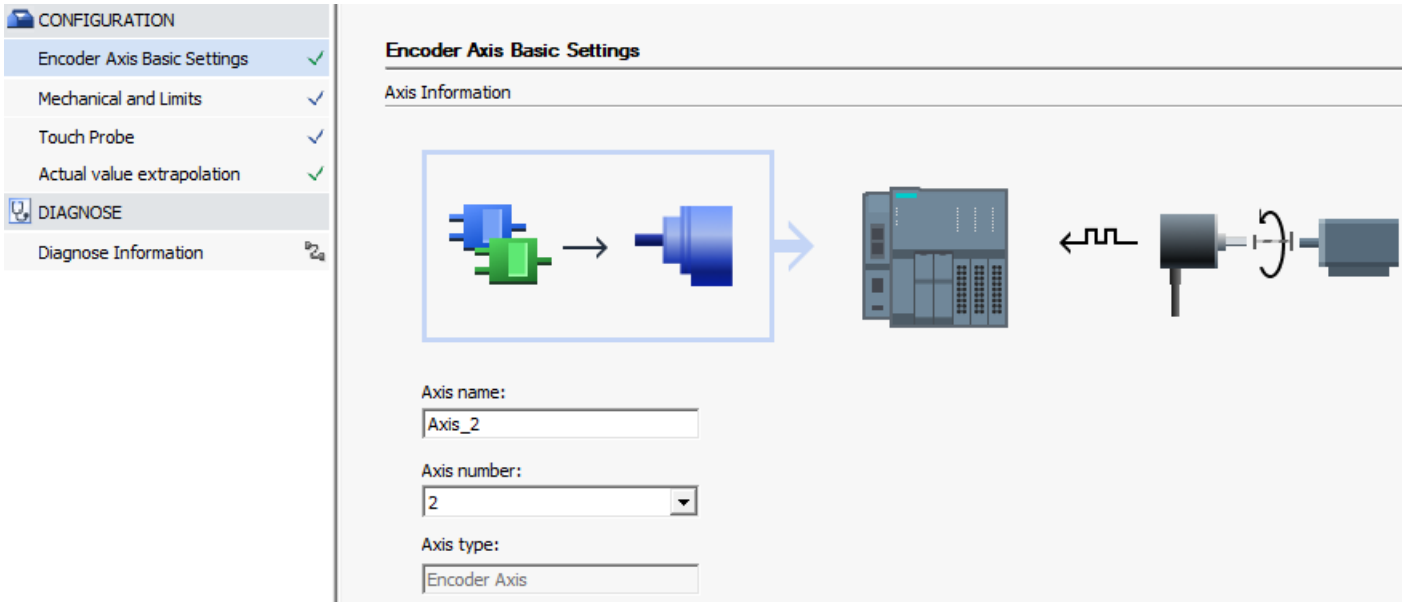
Axis Information

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 must be a letter, Chinese character, 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

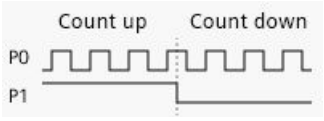
You can configure the pulse and direction input interface to your stepper motor/servo drive.

There are four options for the "Input mode" interface to the stepper motor/servo drive as follows:

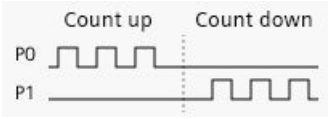
- Pulse (A): If you select the Pulse(A) option, then the input (Pulse) receives the pulsing. The CPU accepts only positive motion commands in this mode. The figure below shows Pulse(A) (assuming positive polarity):



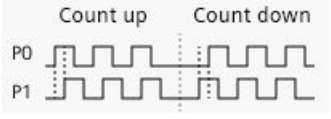
- Pluse(A) + Direction(B): If you select the Pluse(A) + Direction(B) option, then one input (Pulse) receives the pulsing, and one input (Direction) controls the pulse direction. Direction is high (active) if pulsing is in the positive direction. Direction is low (inactive) if pulsing is in the negative 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)".

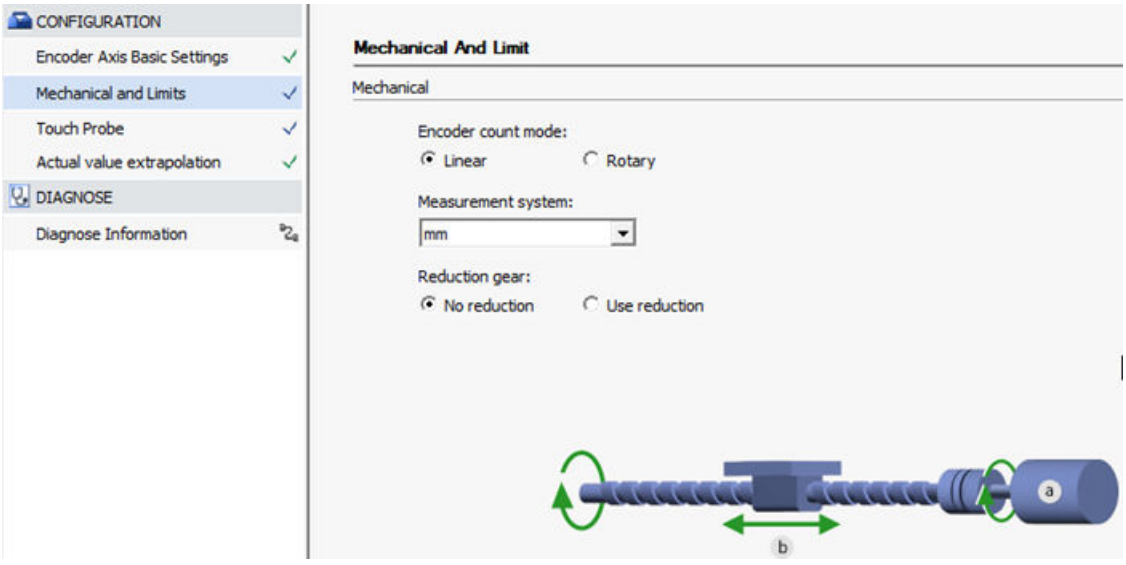
Note

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.

Mechanical and Limits

Mechanical

When you click the "Mechanical and Limits" in the "CONFIGURATION" panel, the following dialog displays:



Configuration item	Mode	Description
Axis mode	Linear	In this mode, the axis can move along a straight line, typically guided by linear rails or tracks.
	Rotate	In this mode, the axis can rotate around a central point or axis of rotation.

Configuration item	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.

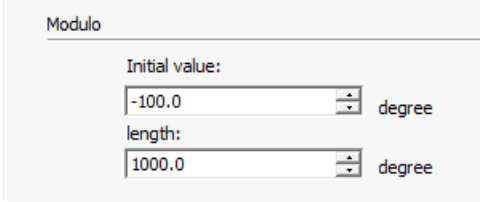
[1] **Note**: The range of the "Number of pulses per rotation of the motor/encoder" parameter is 1 to 9,999,999 pulses. The default value for this parameter is 5,000.

Modulo

The "Modulo" configurations can only be applied if you select "Rotary" for the **Axis mode**.

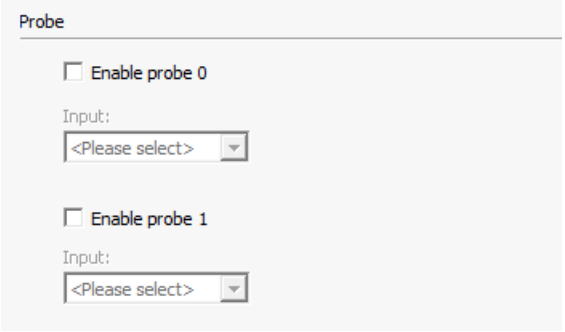
"Initial value" defines from which degree you want the axis to start moving, and "Length" defines how much degrees you want the axis to move. The module range is left-closed and right-open. For example, in the following example, the range of the modulo is [-100, 900).

- 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 500.0.



Touch Probe

You can enable the probe control by selecting the "Enable probe 0" and "Enable probe 1".

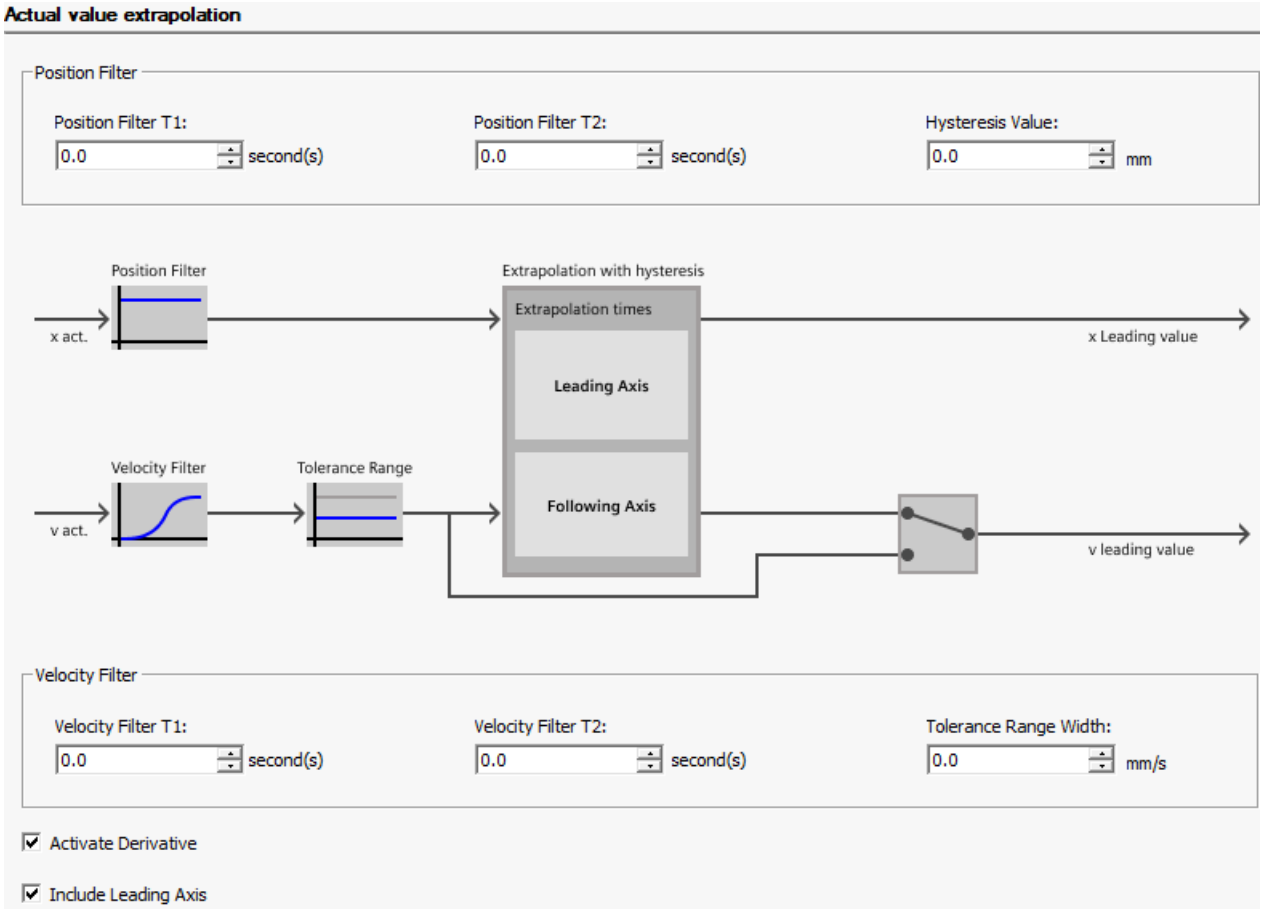


Input

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.

Active Value Extrapolation

Configure the properties of the extrapolation for an actual value coupling for synchronous operation in "Actual value extrapolation". The value set here only takes effect when the actual value of the current axis is used as the leading value.



Position filters T1 and T2

T1 and T2	Enter the time constants of the PT2 filter for smoothing the position.
Hysteresis value	Enter a value for the application of the hysteresis function to the extrapolated actual value of the position. The specification is made in the configured length unit.

Velocity filter

Velocity filters T1 and T2	Enter the time constant of the PT2 filter used to smooth the actual velocity, and the tolerance bandwidth of the smoothed actual velocity.
Tolerance band	For optimized application of the tolerance band, enter the same bandwidth for the tolerance band as the width of the noise signal.

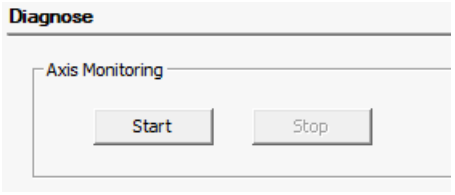
Activate derivative	When you select this checkbox, the leading value velocity will be obtained by differentiating the extrapolated leading value position. When you clear this checkbox, the filtered actual velocity is applied.
Include leading axis	When you select this checkbox, the leading axis-dependent extrapolation time is included in the calculation of the effective extrapolation time. When you clear this checkbox, the leading axis-dependent extrapolation time is not included in the calculation of the effective extrapolation time.

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 state

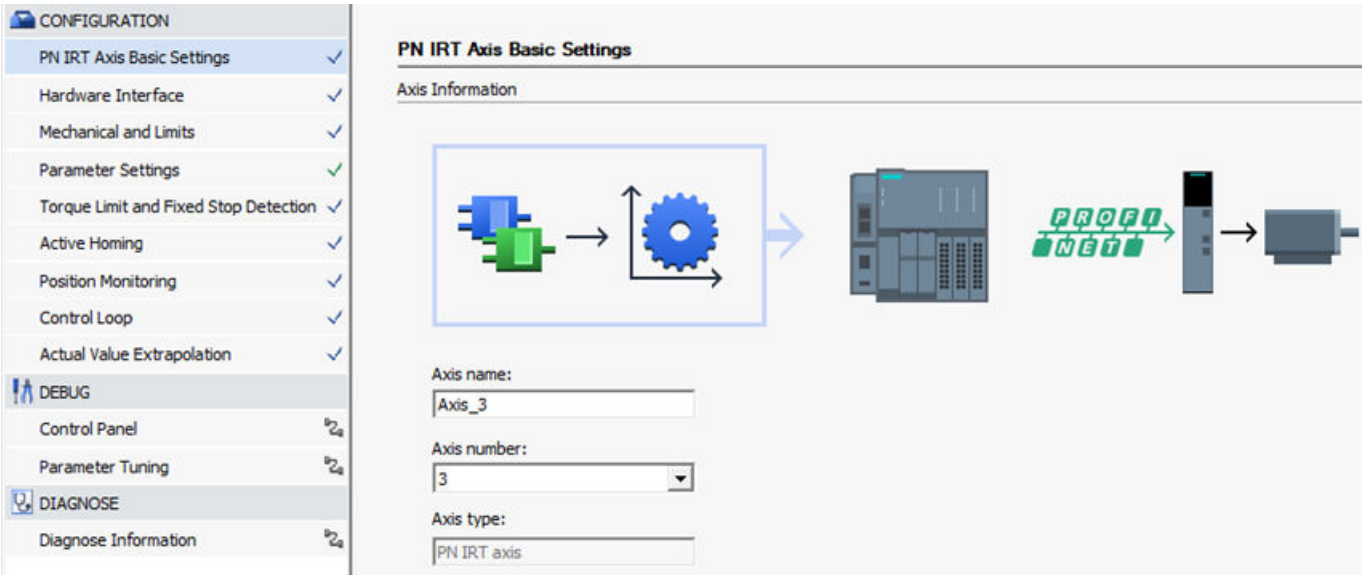
The status of the axis is shown in the dialog. The details would be displayed in "ErrorCode (Page 1015)", "Position", and "Velocity" dialogs.

Axis state	Description
Enable encoder axis	The axis is enabled.
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 its position during subsequent operations

14.4.7.4 Configure PN IRT axis

PN IRT Axis Basic Settings

After a PN IRT axis is added, the "PN IRT Axis Basic Settings" dialog displays as shown in the following figure.



On this page, you can switch the axis ID and modify the axis name, but you cannot modify the axis type.

For newly added axes, you can customize their names on the current page, in the tabs at the bottom of the page, or in the project tree. 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 must be a letter, Chinese character, 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.

Hardware Interface

In the "Hardware Interface" configuration window, select and configure your desired drivers.

Requirement

- You have completed the configuration of the IRT network in the hardware network topology and added all the necessary drives.

If the drive is not available in the hardware catalog, it must be installed in the "Options" menu as a device description file (GSD).

Note

You can only assign each drive to a single axis. For example, if Drive_1 has been assigned to Axis_1, then that drive will not appear in the drive selection drop-down list for other axes.

Drive Selection

In the drop-down list, select the drive you added to the IRT network layer of the hardware network topology.

Data exchange

The data exchange telegram and additional telegrams you configure in your hardware network topology can be viewed in the "Data Exchange" section.

Use offline value

If you check the "Use Offline Value" checkbox, both drive offline values and encoder offline values will be enabled.

Note

- Online value first: If the offline value you enter differs from the online value actually read by the device, the system will prioritize the online value of the current device.
- Unable to read the device online value, and you have selected online mode: The system will directly report an error.
- Unable to read the device online value, and you have selected offline mode: The system will use the offline value you entered.

- Driver
  - **Max.** rotational speed
    Configure the maximum velocity of the drive in this field.
  - **Reference** speed
    Configure the reference velocity of the drive in accordance with the manufacturer's specifications in this field.
  - Base Torque
    Configure the base torque in accordance with the drive configuration in this field.
    Base torque is needed for force/torque deceleration, which can be supported via telegram 10x.
- Encoder: For closed-loop position control, positioning axes require an actual position value in the form of an encoder position. The encoder position is transmitted to the controller by means of a PROFIdrive telegram.
  Select the encoder type of the encoder in the drop-down list. The following encoder types are available:
  - Incremental encoder
  - Single-turn absolute encoder
  - Multi-turn absolute encoder

	Input range	**Default**	**Units**
Max. rotational speed	1.0 ~ 10000.0	3000.0	rpm
Reference speed	1.0 ~ 10000.0	3000.0	rpm
Base Torque	0.0 ~ 1000000000.0 (1E09)	3.2	N·m
Increments per encoder revolution	1 ~ 9999999	2048	
Speed	1 ~ 9999999	1	
Gx_XIST1	0 ~ 31	11	
Gx_XIST2	0 ~ 31	9	

Mechanical and Limits

Mechanical

When you click the "Mechanical and Limits" in the "CONFIGURATION" panel, the following dialog displays:

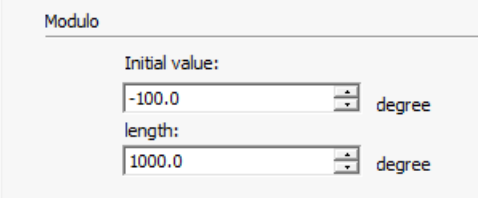
Configuration item	**Mode**	**Description**
Axis mode	Linear	In this mode, the axis can move along a straight line, typically guided by linear rails or tracks.
	Rotate	In this mode, the axis can rotate around a central point or axis of rotation.
Units	deg, cm, mm, inch, or degree	All speeds in controlling the motion of the axis are expressed in units of "Measurement system" per second. The default value is millimeters.

Configuration item	Mode	Description
Reduction gear	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 the "Drive motor speed" and "Load revolutions", the changes in output speed and torque are affected.

Modulo

The "Modulo" configurations can only be applied if you select "Rotary" for the **Axis mode**.

"Initial value" defines the initial position of the axis, and "Length" defines how many degrees the axis needs to move. The module range is left-closed and right-open.



Motor and encoder reversal

Select the "Motor and encoder reversal" checkbox to reverse the rotation direction of the motor and encoder.

Position parameters

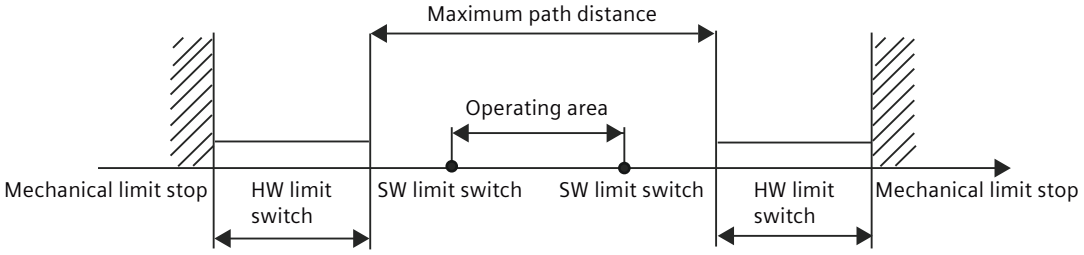
	Linear axis	Rotary axis
Units	mm	degree
Position parameters	In the "Leadscrew pitch" configuration field, configure the distance by which the load is moved when the leadscrew makes one revolution.	In the "Length of one rotation" configuration field, configure the number of degrees for one full rotation of the load.

Reduction gear

After checking "Use reduction", you can configure "Drive motor speed" and "Load speed".

Hardware limits

Software/hardware limit switches ensure that axes operate within the workbench's effective range. If an axis malfunctions and exceeds the limit switch, the axis will stop running and trigger an error upon reaching either limit.



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 item	Description
Input	You can assign LMT-/LMT+ to an input on 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 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 active level. The default level is "High".
Response	In this parameter, you can assign what should happen if the LMT-/LMT+ input goes active.

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 device.

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 "Enable software limit" checkbox is selected, you can access the "Software negative limit position", "Software positive limit position", and "Response" parameters.



Note

If Homed=0 in TO, the software limit will not take effect.

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.

Note

For IRT axes, when operating in non-position control loop mode (e.g., when using instructions such as MC_MoveJog, MC_MoveVelocity, or TorqueControl (Mode 1)), the reference position of the software limit will automatically switch from the instruction-given position to the actual position of the axis.

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 deceleration" in Parameter Settings (Page 848).

Parameter Settings

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 Unit (Page 844);

⚠ WARNING

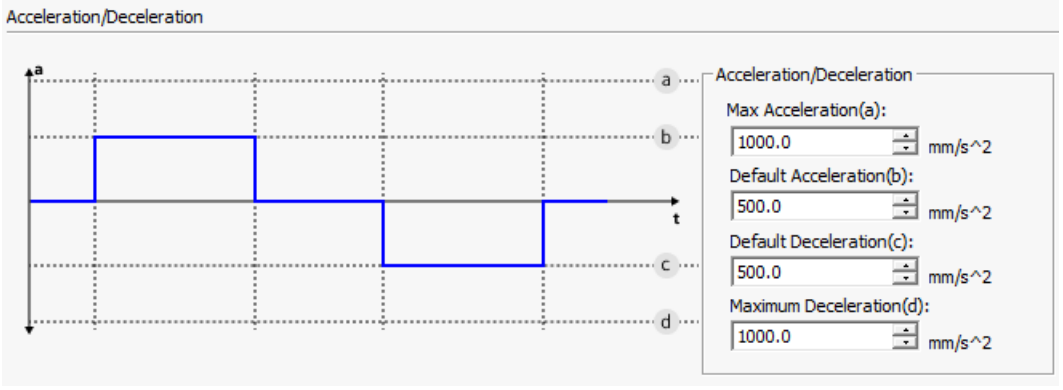
Set an appropriate max velocity

Default max velocity for linear axes is 1000.0 mm/s and default max velocity for modal axes is 36000.0 deg/s.

Be sure to adjust the maximum velocity based on actual application scenarios to avoid equipment damage or personal injury.

Acceleration/Deceleration

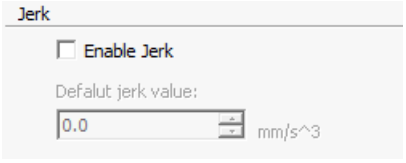
In the Acceleration/Deceleration dialog, you can define the acceleration speed and deceleration speed for your application.



- **Max acceleration**: used to limit the maximum acceleration for user-configured motion instructions.
- **Default acceleration**: the default value for the acceleration parameter in Motion Control Instructions (Page 906).
- **Default deceleration**: the default value for the deceleration parameter in Motion Control Instructions (Page 906).
- **Max deceleration**: used to limit the maximum deceleration for user-configured motion instructions.
  For PN IRT axis hardware/software limits, the deceleration used is the maximum deceleration configured for the axis.
  Motor acceleration and deceleration settings are determined by trials and errors. You should start by entering a large 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.



Torque Limit or Fixed Stop Detection

Requirement

- In the hardware topology network, the IRT network added PROFINET devices and configured Telegram 105.
- The device was selected on the hardware interface page.

Torque limiting

Configure the torque limit percentage of the drive in "Torque limiting".

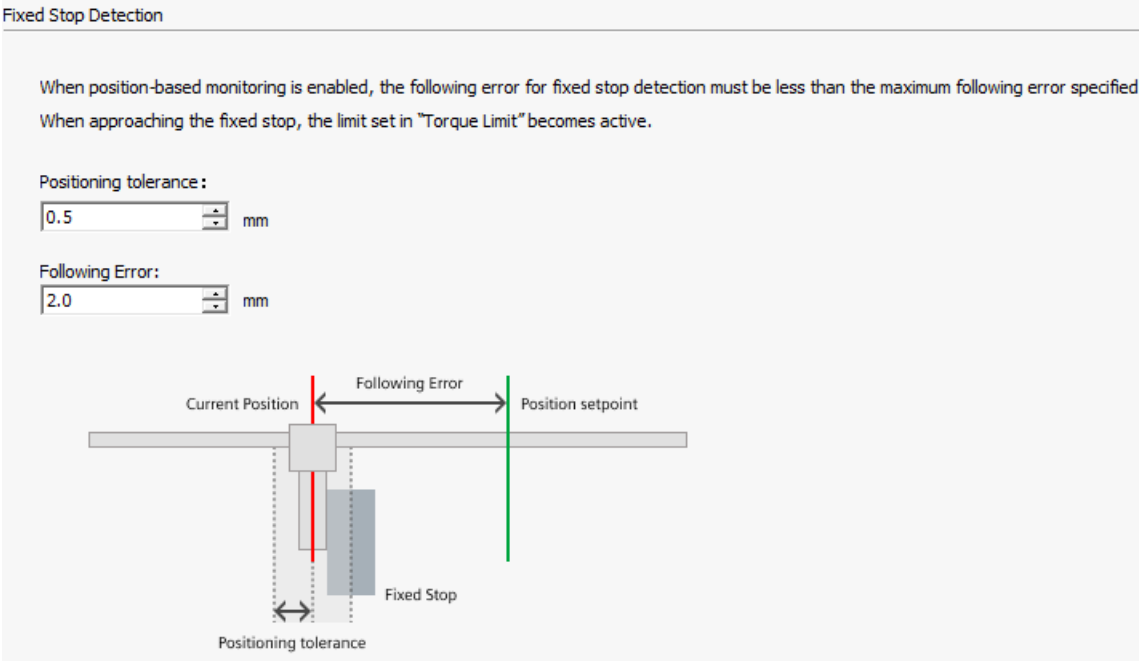
Position-based monitoring

When the torque/force limit is enabled, if the torque has already reached the threshold, the motor's closed-loop position control will be affected, causing the actual position to continuously deviate from the set position, and the following error to continuously increase.

To detect and respond to the above situations in a timely manner, you can enable position monitoring, including monitoring of following errors, standstill, etc.

If "Disable position-based monitoring" is checked, position monitoring will be disabled when TorqueLimiting is enabled, and will be restored when TorqueLimiting is disabled.

Fixed stop detection



Positioning tolerance	Exceeding the positioning tolerance is regarded as a breaking away or turning back of the fixed stop when exceeded. To detect the breaking away or turning back of the fixed stop, the position setpoint must be located outside the positioning tolerance. The configured positioning tolerance must be less than the configured following error.
Following error	The following error value from which the fixed stop detection starts to take effect. If the drive is stopped by a mechanical fixed stop during a motion job, the following error is increased. The accumulating following error serves as a criterion for fixed stop detection.

InClamping working mechanism

Entering InClamping state

The system continuously monitors whether the following error exceeds a preset threshold to determine whether the axis has contacted the fixed stop.

State behavior:

- Once the axis enters the InClamping state, the currently executing motion instruction will be aborted.
- The system will immediately record the current actual position of the axis.
- The axis will enter the Standstill state.

In this state, due to a certain deviation between the set position and the actual position, the system will generate a holding torque to firmly clamp the axis in the stop position.

If you attempt to execute a motion instruction in the same direction as the stop while in InClamping state, the instruction will fail with the error "Error executing FB in InClamping". It should be noted that even if the instruction reports an error, the axis will not enter the ErrorStop state, and the motor will continue to apply pressure to the stop to maintain the clamped state.

Normal breaking away from InClamping state

To properly break away from the InClamping state, a motion instruction opposite to the direction of the stop needs to be executed.

During this breakaway, the motion state of the axis will be managed and controlled by the currently invoked instruction.

When the actual position of the axis exceeds the set range of the positioning tolerance, the system determines that the axis has successfully broken away from the InClamping state.

Abnormal breaking away from InClamping state

If the actual position of the axis unexpectedly breaks away from the set range of the positioning tolerance without executing the reverse direction motion instruction, it is considered an abnormal breaking away from the InClamping state.

Abnormal breaking away may be caused by:

- The mechanical stop was moved by someone.
- The limit value is too large, causing the axis to exceed the stop limit.

If abnormal breaking away occurs, the system will immediately report an error: "Abnormal breaking away from InClamping". At this point, the axis will enter the ErrorStop state and needs to be reset by executing the MC_Reset instruction to recover.

Abnormal breaking away determination

When the set position (or target position) is outside the positioning tolerance, but the actual position deviates further from the stop direction and exceeds the set range outside the positioning tolerance, the system will determine it as an abnormal breaking away.

Note

When abnormal breaking away occurs, the motor may move rapidly due to the drastic change in the following error, which may lead to an overload error.

If the target position is still within the set range of the positioning tolerance, the system will not be able to detect any abnormal breaking away.

Active Homing

When the axis triggers the active homing operation, the axis will search for the homing signal according to the configured velocity and complete the homing command.

Mode selection

Select the homing mode. Active homing supports the following homing modes:

- Use the zero position signal from the servo drive
- Use the reference stop and the zero position signal from the servo drive
- Use a digital input as the homing signal
- Use the reference stop and a digital input as the homing signal

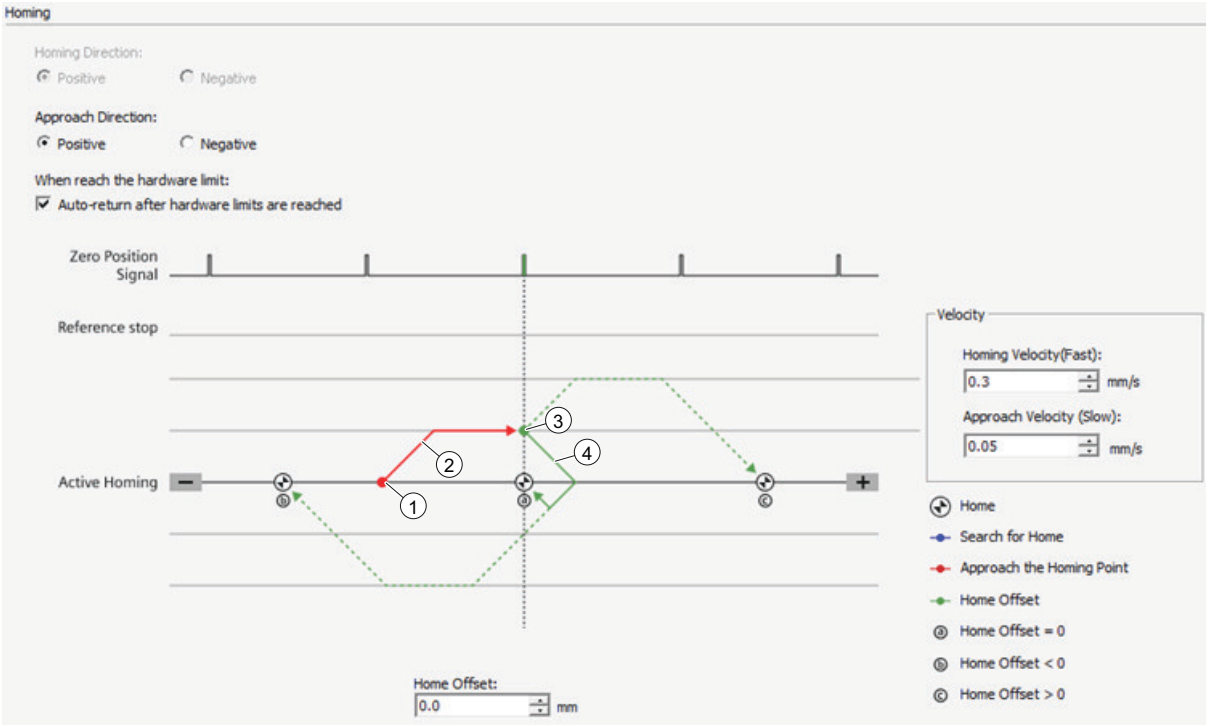
Input signal

- Inputs: Select the homing signal.
- Trigger level: The effective level of the homing signal.
- Home detection direction: A valid input signal has two edges, and one of the edges is designated as the homing signal.

Homing

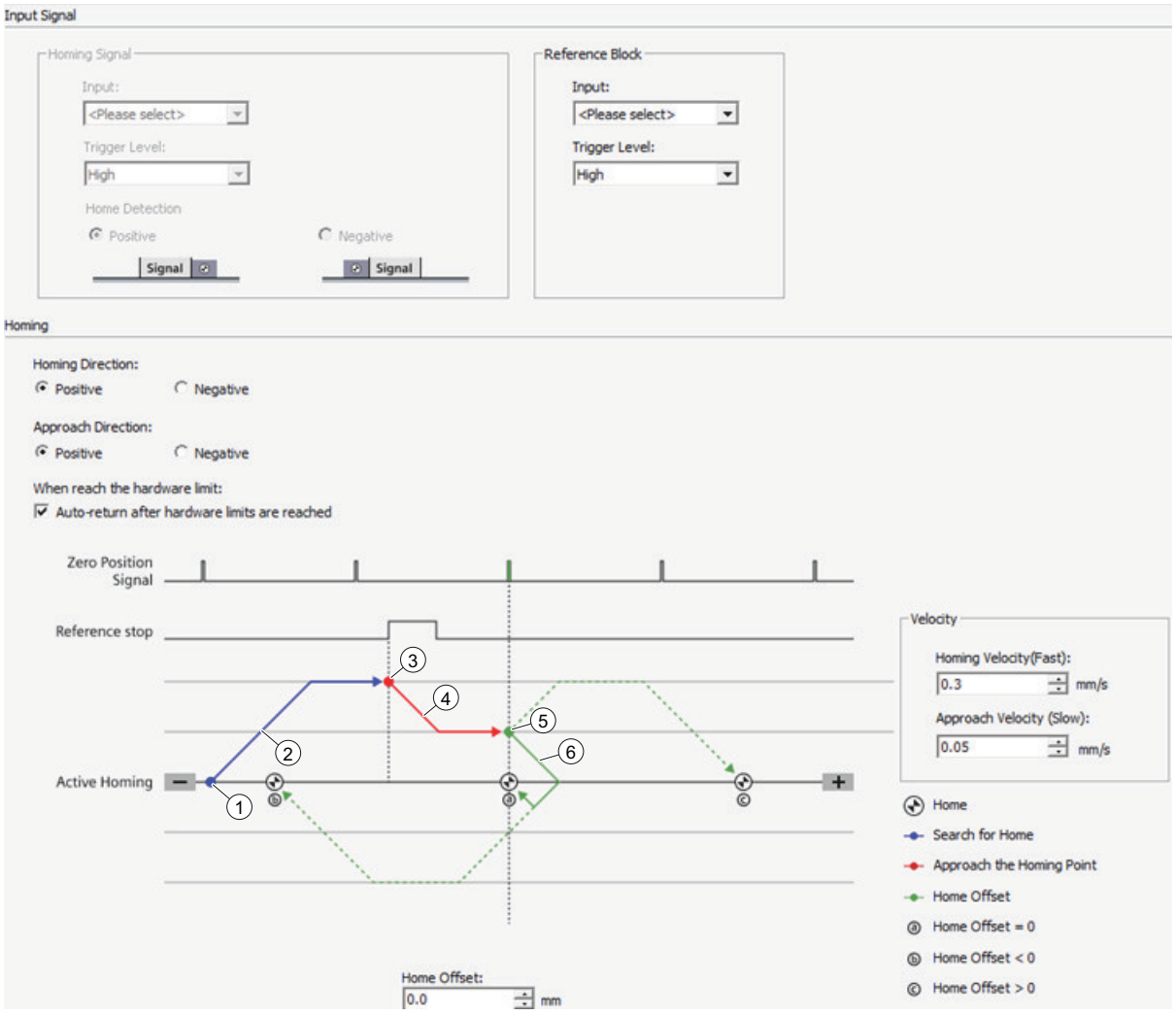
- Homing direction: Specifies the initial direction of motion for searching for the homing signal. When the program triggers active homing, the axis immediately moves in the "homing direction" to search for the homing signal.
- Approach direction: Specifies the final direction of the motion approaching the homing signal.
- Auto-return after hardware limits are reached: When you select this checkbox, if a hardware limit is triggered during the search for the homing signal, the axis will automatically reverse and continue the search. Otherwise, the axis will stop, and the instruction will report an error.
- Homing velocity: Specifies the velocity of the motion for searching for the homing signal. When the program triggers active homing, the axis immediately searches for the homing signal at the "Homing velocity".
- Approach velocity: Specifies the final velocity of the motion approaching the homing signal. After the axis finds the homing signal, it immediately continues to move at the approach velocity and approaches the homing signal again in the approach direction. The "Approach velocity" should be less than the "Homing velocity", and neither the "Homing velocity" nor the "Approach velocity" should be set too fast. Within an acceptable range, it is recommended to set a lower velocity value.
- Home offset: The offset of the final home position from the homing signal. When set to 0, the home position coincides with the homing signal.
- Zero position signal The zero position signal is typically generated by an incremental encoder, which produces a zero position signal once per revolution. In the homing process, the zero position signal can be used as an "Homing signal" with high precision.
- Homing signal: The homing signal serves as the reference position for the origin of the positioning system. It can be either an edge of the reference stop signal or a high-precision zero position signal. It is the key reference point for the system to determine the "origin". In the homing path diagram, the homing signal is typically the endpoint of the red line segment.

Example: Use the zero position signal from the servo drive



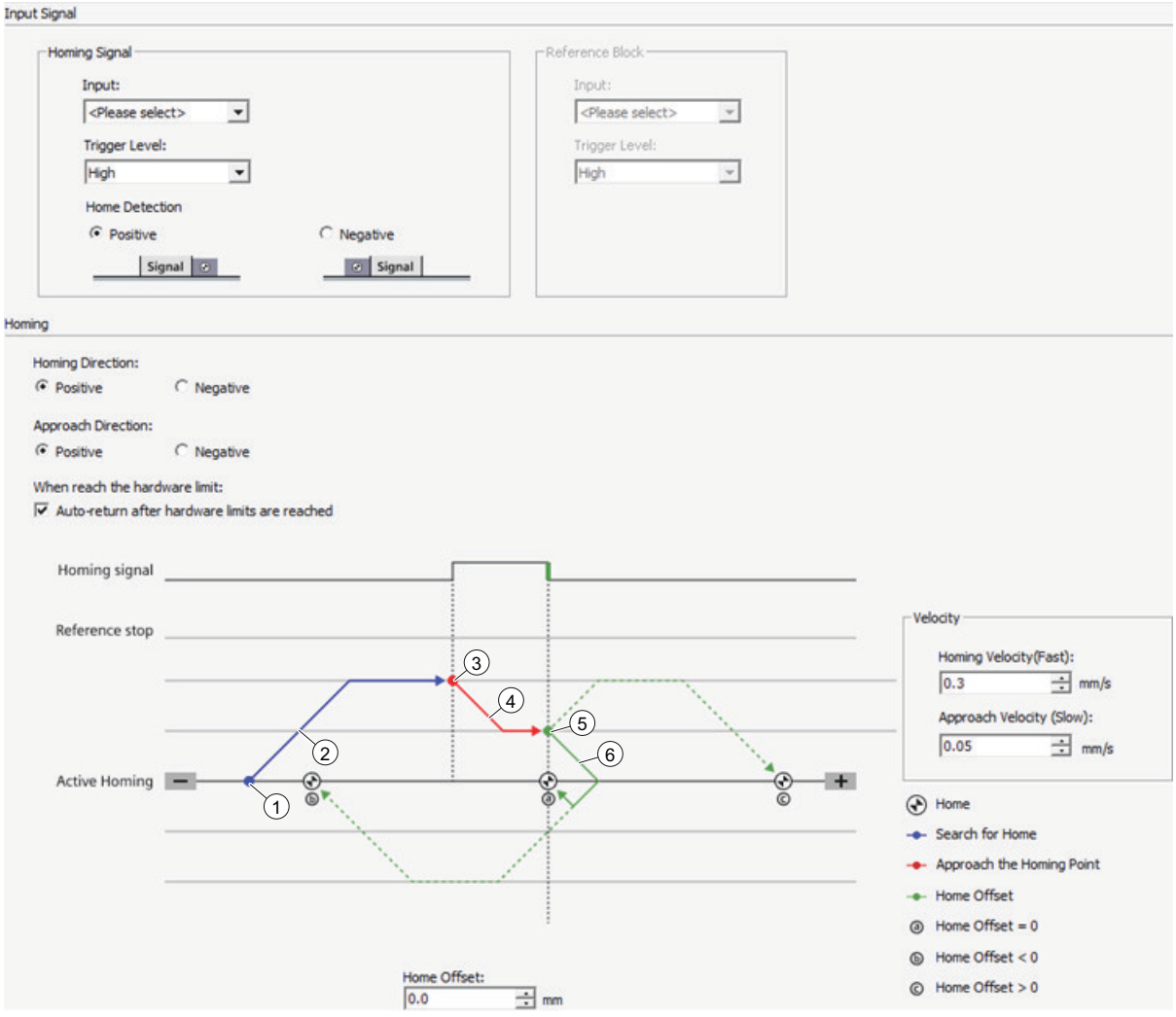
①	Start of active homing via the "MC_Home" motion control instruction
②	Search for the zero position signal in the approach direction at the approach velocity
③	Zero position signal is detected
④	Move to the home position at the homing velocity. If a home offset is set, the motor will move to the offset position.

Example: Use the reference stop and the zero position signal from the servo drive



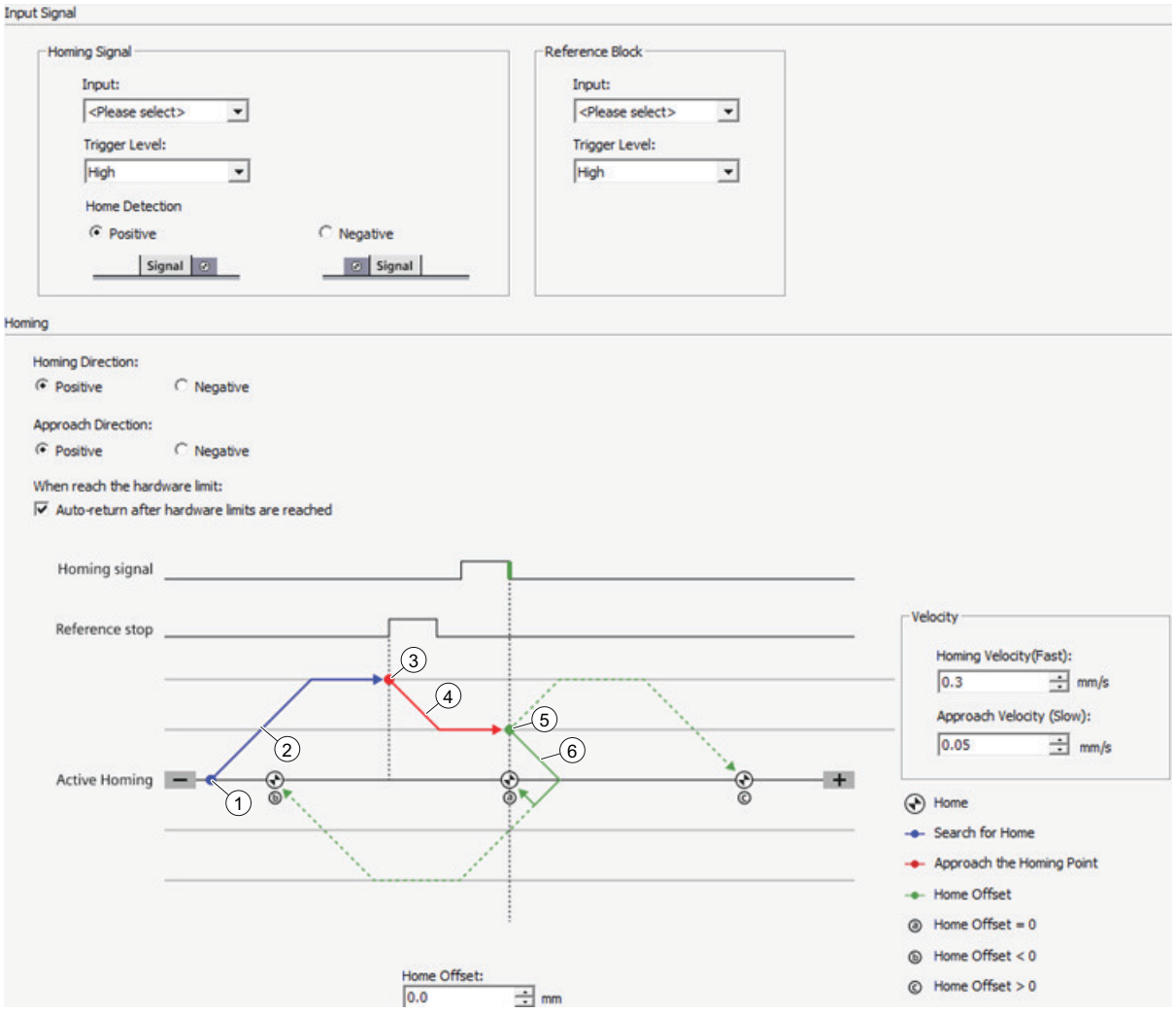
①	Start of active homing via the "MC_Home" motion control instruction
②	Search for the rising edge of the reference stop signal in the homing direction at the homing velocity
③	The rising edge of the reference stop is detected
④	Search for the zero position signal in the approach direction at the approach velocity
⑤	Zero position signal is detected
⑥	Move to the home position at the homing velocity. If a home offset is set, the motor will move to the offset position.

Example: Use a digital input as the homing signal



①	Start of active homing via the "MC_Home" motion control instruction
②	Search for the rising edge of the homing signal in the homing direction at the homing velocity
③	The rising edge of the homing signal is detected
④	Search for the falling edge of the homing signal in the approach direction at the approach velocity
⑤	The falling edge of the homing signal is detected
⑥	Move to the home position at the homing velocity. If a home offset is set, the motor will move to the offset position.

Example: Use the reference stop and a digital input as the homing signal



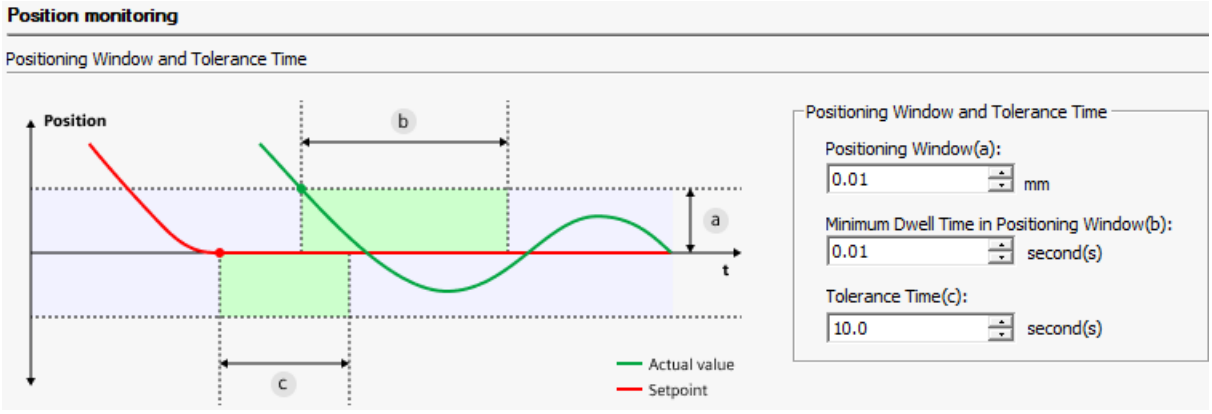
①	Start of active homing via the "MC_Home" motion control instruction
②	Search for the rising edge of the reference stop signal in the homing direction at the homing velocity
③	The rising edge of the reference stop is detected
④	Search for the homing signal in the approach direction at the approach velocity
⑤	Homing signal is detected
⑥	Move to the home position at the homing velocity. If a home offset is set, the motor will move to the offset position.

Position Monitoring

Position Monitoring

Configures the standard used to monitor the target position.

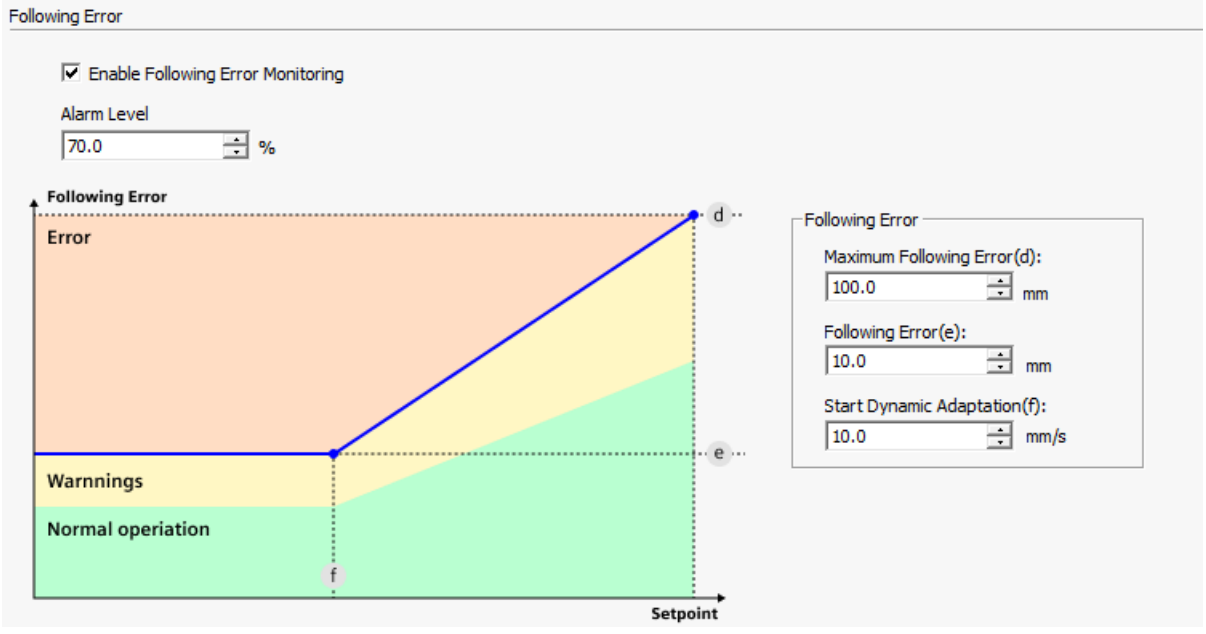
It monitors the behavior of the actual position at the end of the calculation. If the actual position reaches the positioning window within the tolerance time and remains in the positioning window for the minimum dwell time, a motion command is completed.



Positioning window (a)	Configure the size of the positioning window.
Minimum dwell time in positioning window (b)	Configures the minimum dwell time. The duration for which the actual position value remains within the "Positioning window" must be greater than the "Minimum dwell time within positioning window".
Tolerance time (c)	Configures the tolerance time. As soon as the setpoint velocity reaches the zero value, the actual position value must reach the "Positioning window" within the "tolerance time".

Following error

Configures the permissible deviation of the actual position of the axis from the set position. The following error can be dynamically adapted to the current velocity of the axis.

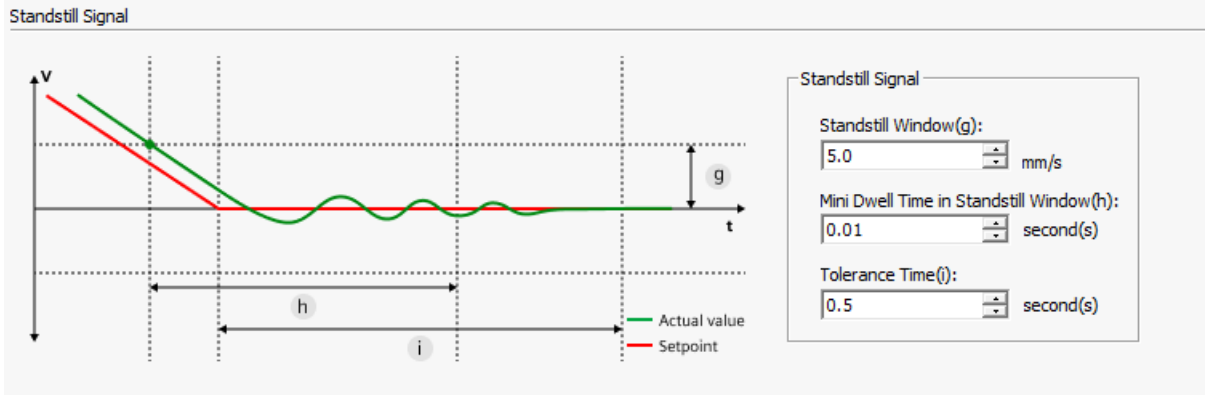


Enable following error monitoring	When you select this checkbox, the axis is stopped in the error range (orange).
Maximum following error (d)	Configures the following error that is permissible at maximum velocity.
Following error (e)	The permissible following error (without dynamic adjustment) when the velocity is less than the "Start dynamic adjustment".
Start Dynamic Adaptation (f)	Once the configured velocity is exceeded, the following error will be dynamically adjusted.
Alarm level	Configures a percentage of the current following error limit, above which an alarm should be output.

The following error is the difference between the setpoint position and the actual position value of the axis. The transmission times of the setpoint to the drive and of the actual value to the controller are taken into account in the calculation of the following error. The permissible following error depends on the setpoint velocity. When the velocity setpoint is less than "Start of dynamic adjustment", the tolerance range of the following error is constant; while when the velocity setpoint is higher than "Start of dynamic adjustment", the following error increases proportionally with the velocity setpoint. When the difference between the set position value and the actual position value exceeds the permissible deviation range of the following error, the axis will stop. In the initial stage of motion axis commissioning, the "Following error" can be set to a larger value to avoid frequent following error faults in the motion axis.

Standstill signal

Configures the standard for standstill detection.

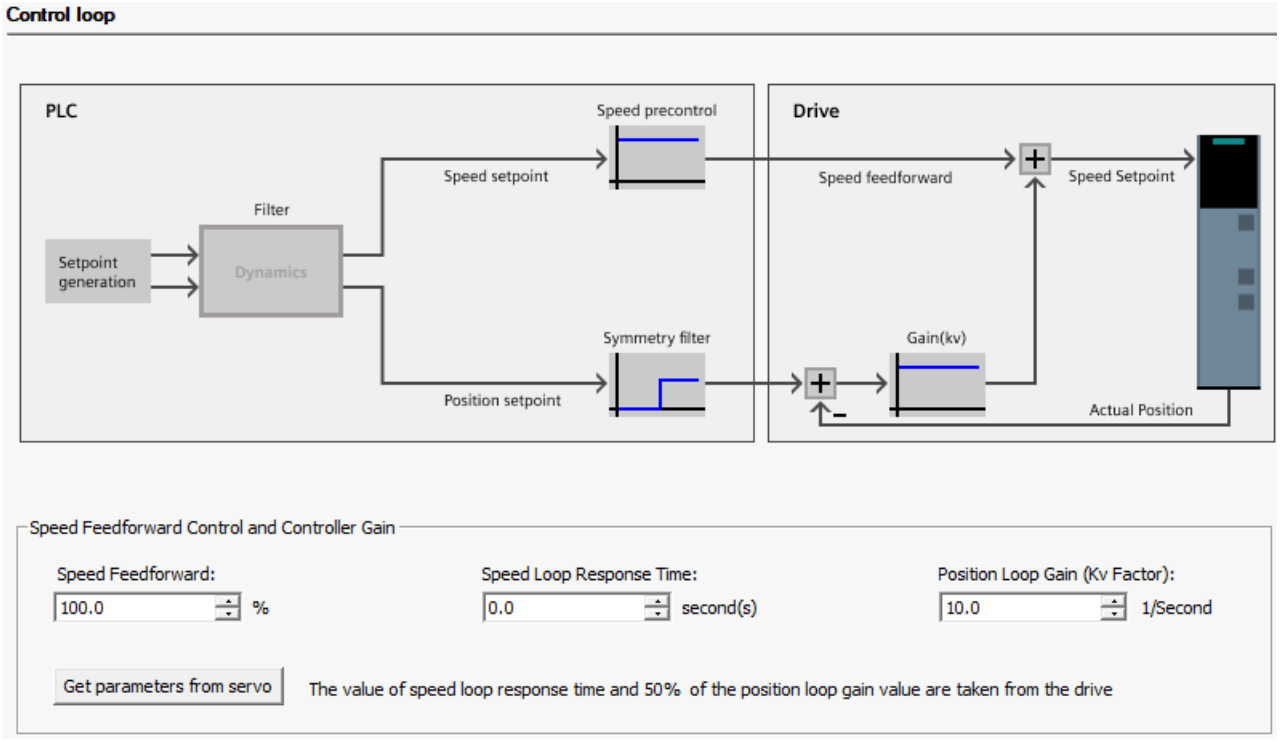


Stop window (g)	Configures the size of the stop window.
Minimum dwell time in stop window (h)	Configures the minimum dwell time in the stop window. The axis's actual velocity must be maintained within the "Stop window" for the "Minimum dwell time in stop window", and the axis is displayed as standstill.
Tolerance time (i)	Configures the tolerance time within which the position value must reach the stop window.

Control Loop

Configure the speed precontrol and position loop gain (Kv factor) for the position control loop.

After activating speed precontrol, the setpoint is delayed by the speed loop response time before the control deviation is applied. This prevents an overshoot or a leading of the actual value compared with the position setpoint. The speed loop response time is a simplified substitute model for the dynamic characteristics of the speed control loop. The speed loop response time is included in the balancing filter.



Speed precontrol	Modify the speed precontrol percentage for the control loop. "Speed precontrol" improves the dynamic response characteristics of the system, but excessively large setting values can cause the position control system to overshoot.
Speed loop response time	Configure speed loop response time (Tvtc).
Position loop gain (Kv factor)	• Positioning accuracy and stop control • Uniformity of motion • Positioning time The better the mechanical condition of the axis is (high stiffness), the higher the Kv factor can be configured. This reduces the following error, and a higher dynamic response is achieved.

Get parameters from servo

You can use this function to obtain the servo parameters of the PN IRT axis control loop.

Note

This function is currently only supported on the S200.

Prerequisites

- The motor has been successfully connected to the PLC, and the axis has been successfully configured.
- Ensure the servo is optimized before acquiring parameters from the servo.

Procedure

To configure position control in the drive for an axis, follow these steps:

1. Select the "Get parameters from servo" option.
2. Enter the servo's username and password to log in (when the servo-side user management function is enabled).

Speed precontrol	Custom, not retrieved from the servo, default value is 100%.
Speed loop response time	Retrieve servo parameter r5277: Precontrol symmetrical time estimation, unit has been converted.
Position loop gain (Kv factor)	Retrieve the servo parameter r5276: The estimated value of kv factor is 50% of r5276, and the units have been converted.

Note

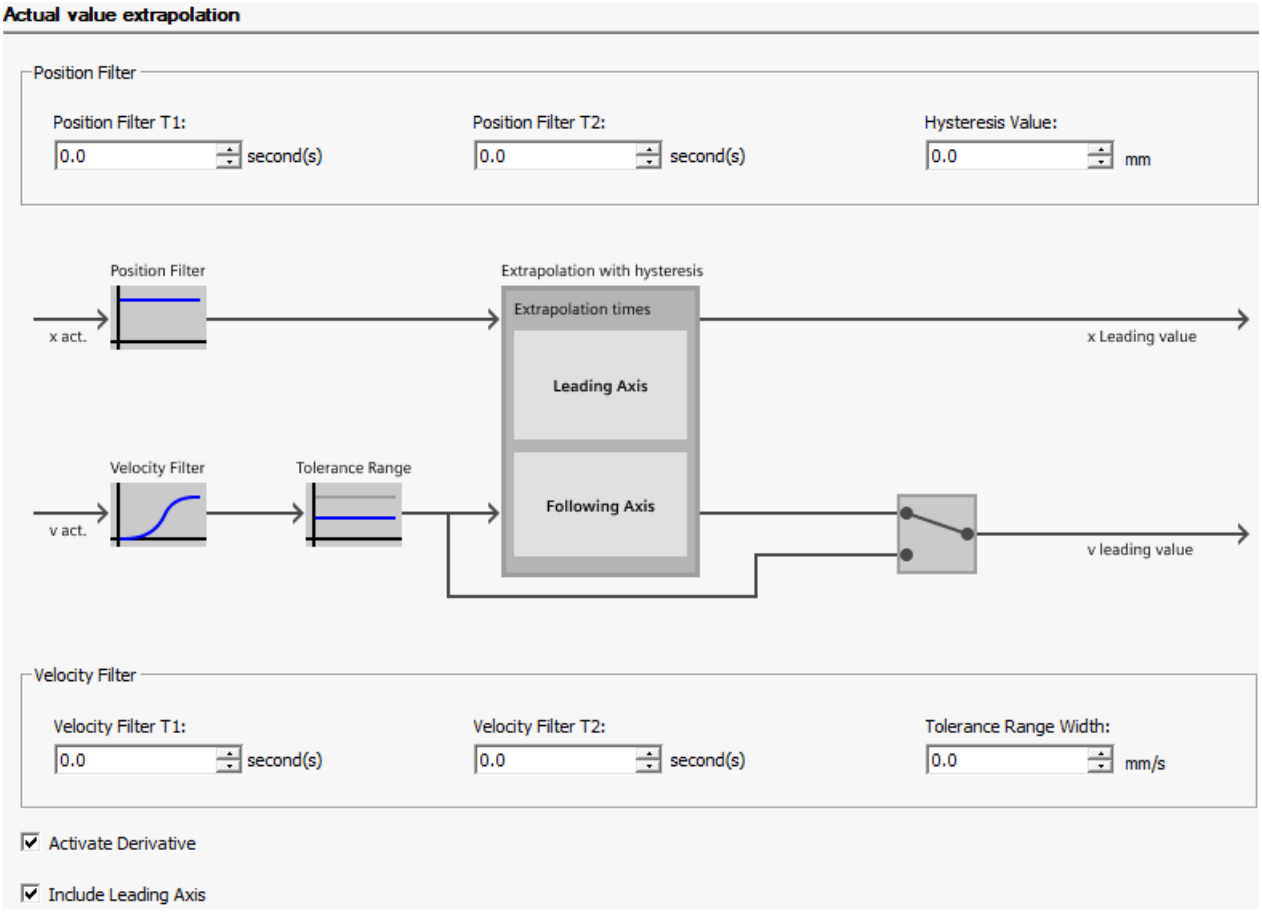
If the device name is configured incorrectly or the network card interface is selected incorrectly, the system will prompt you that the device is not found, and you need to reselect the device. After clicking "Yes", you can reselect the device in the pop-up dialog box.

Note

If the device has not yet been configured with an IP address, the system will display a dialog box to guide you through setting the IP address. After successful setup, the IP address will be updated synchronously in the hardware topology network and the actual device.

Active Value Extrapolation

Configure the properties of the extrapolation for an actual value coupling for synchronous operation in "Actual value extrapolation". The value set here only takes effect when the actual value of the current axis is used as the leading value.



Position filters T1 and T2

T1 and T2	Enter the time constants of the PT2 filter for smoothing the position.
Hysteresis value	Enter a value for the application of the hysteresis function to the extrapolated actual value of the position. The specification is made in the configured length unit.

Velocity filter

Velocity filters T1 and T2	Enter the time constant of the PT2 filter used to smooth the actual velocity, and the tolerance bandwidth of the smoothed actual velocity.
Tolerance band	For optimized application of the tolerance band, enter the same bandwidth for the tolerance band as the width of the noise signal.

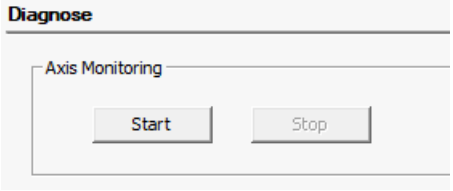
Activate derivative	When you select this checkbox, the leading value velocity will be obtained by differentiating the extrapolated leading value position. When you clear this checkbox, the filtered actual velocity is applied.
Include leading axis	When you select this checkbox, the leading axis-dependent extrapolation time is included in the calculation of the effective extrapolation time. When you clear this checkbox, the leading axis-dependent extrapolation time is not included in the calculation of the effective extrapolation time.

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 state

The status of the axis is shown in the dialog. The details would be displayed in "ErrorCode (Page 1015)", "Position", "Velocity", and "Direction" dialogs.

Axis state	Description
Error	An axis error occurred. Refer to the ErrorID and ErrorInfo parameters in the Motion Control instructions for details
Disable	The axis is disabled and cannot be controlled via Motion Control
Standby	The axis is at a standstill state
Discrete motion	The axis is moving in Discrete state
Continuous	The axis is moving in Continuous state
Synchronous motion	The axis is moving in Synchronized state
Homing in progress	The axis is returning to the home.
Stopping	The axis is Stopping
ErrorStop	The axis is experiencing an error condition and is currently in the ErrorStop state
Homed	The axis has returned to its home position
Hardware positive limit switch	The hardware positive limit switch is reached or exceeded
Hardware negative limit switch	The hardware negative limit switch is reached or exceeded
Software positive limit switch	The software positive limit switch is reached or exceeded

Axis state	Description
Software negative limit switch	The software negative limit switch is reached or exceeded
Reference stop	Reference stop
Homing signal	Digital input
The axis is clamped at a fixed stop	Clamping
Configuration error	Configuration error
Following error exceeds limit	Following error exceeds limit
Following error warning	Following error warning
Configuration warning	Configuration following error warning

Servo status

The servo status and name are shown in the dialog.

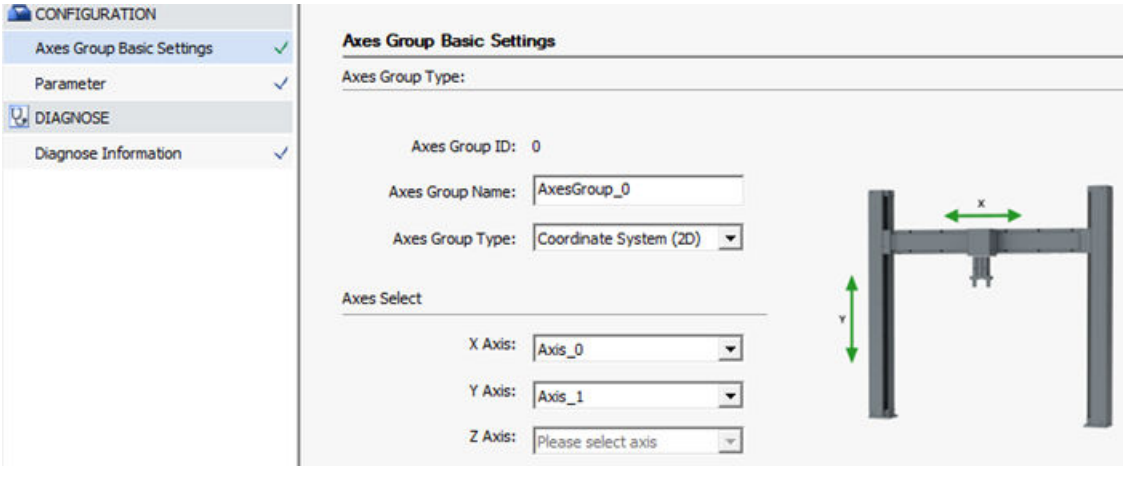
14.4.8 Configuring Axes Group

14.4.8.1 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. Do not use keywords as variable names. 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.

Parameters

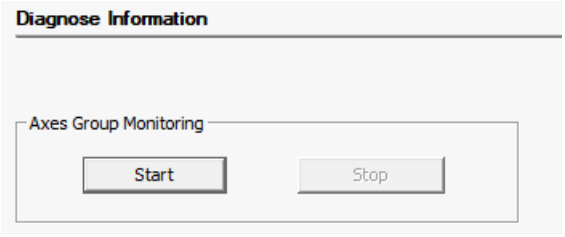
Set the parameters of the Axes Group, such as maximum velocity, maximum acceleration, maximum deceleration.

14.4.8.2 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 "Start" button, and disable by click on the "Stop" button.



State of axes group

The status of the axes group is shown in the dialog. The details would be displayed in "ErrorCode (Page 1015)" 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

14.4.9 Configuring the Cam

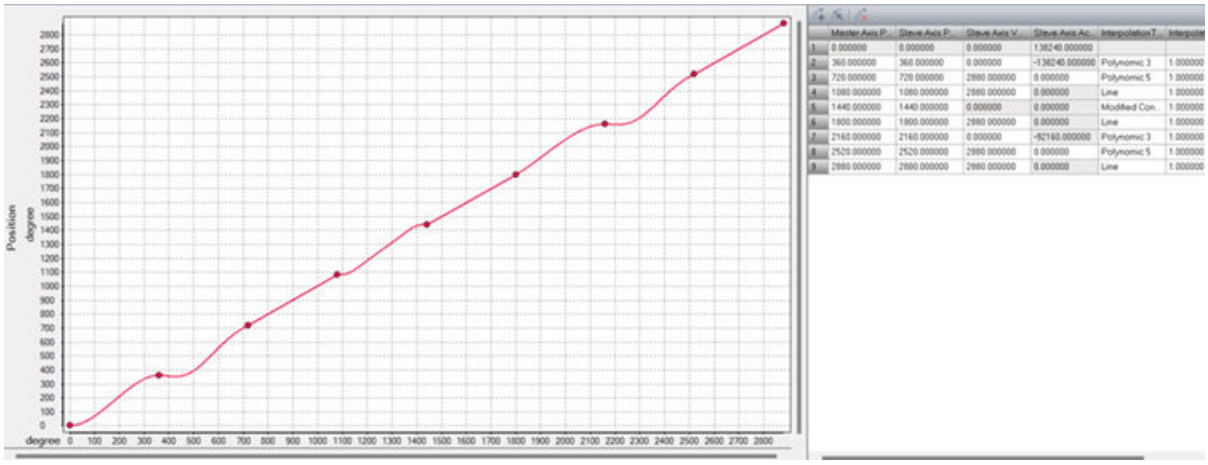
Compared to traditional cam mechanisms, the use of an electronic cam offers greater flexibility, precision and efficiency. An electronic cam controls the slave axis in synchronization with the master axis, using pulse trains, according to a pre-defined electronic cam table.

14.4.9.1 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.

Parameters	Description
Master axis phase	Set the phase of master axis
Slave axis position	Set the position of slave axis

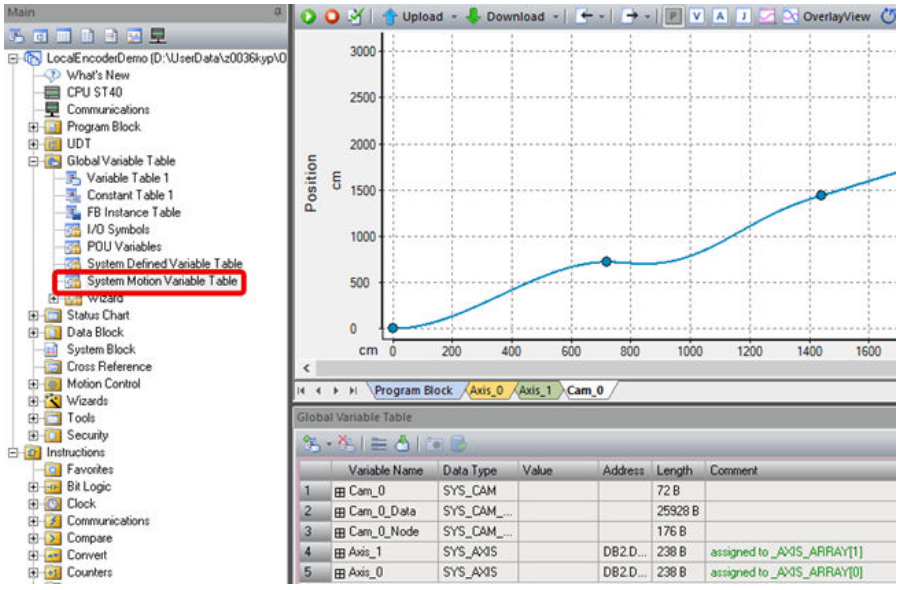
Parameters	Description
Slave axis velocity	• When the interpolation type is constant, or modified constant velocity, slave velocity is set to 0 and cannot be edited. • When the interpolation type is line, slave velocity is created automatically when selected. • When the interpolation type is cubic polynomial 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".

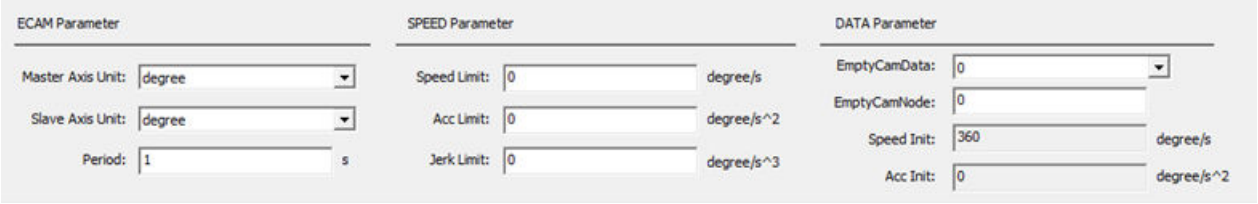
Note

The values of "Cam_x_Data" and "Cam_x_Node" can be changed by program segments or status chart.



14.4.9.2 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 units	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.

14.4.9.3 View the curve

Zoom/Move curves

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.

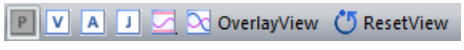
Move the curve: You can move the curve around by clicking and holding down the right mouse button in the curve area.

Tools for viewing curves

You can use the tools in the toolbar to view the curve.

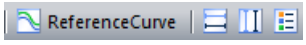


	In this mode, you can move the timing diagram by dragging the mouse
	Zooms in on the center of the canvas in 10% increments. The ranges of the time axis and value axis are reduced every time the button is clicked. The curves are displayed larger
	Zoom out from the center of the canvas in 10% increments. The ranges of the time axis and value axis are increased every time the button is clicked. The curves are displayed smaller
	On the timing diagram, press and hold the left mouse button, then drag from the top-left to the bottom-right to draw a rectangular selection box. After releasing the left mouse button, the system will automatically zoom in on the selected area



	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.



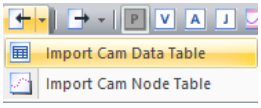
	Show or hide reference curves.
	Show or hide horizontal reference lines.
	Show or hide vertical reference lines.
	Show or hide vertical legend.

14.4.9.4 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 table you want to import.



Note

Note the following when import cam node 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 cam node table has a maximum point limit of 361, and the total number of points across all cam node 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.

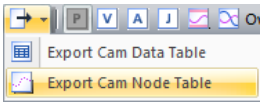
Note

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

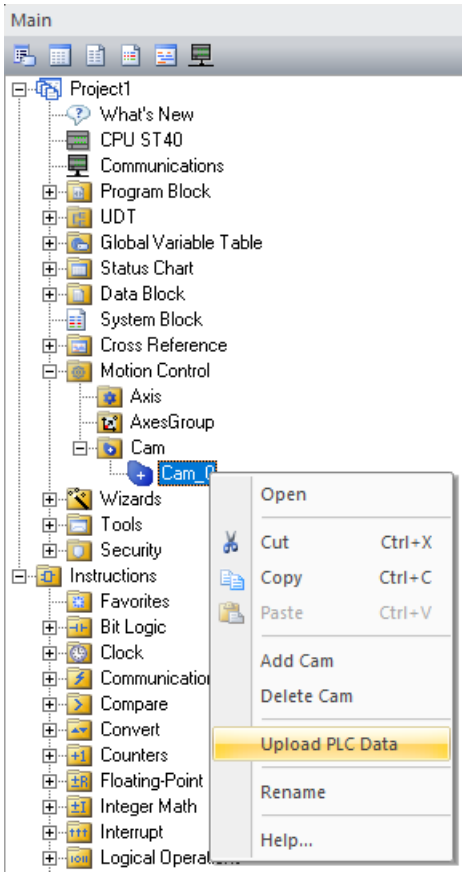
1. To export cam configuration or cam data, click the down arrow beside the export button, and then select the table you want to export.



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.



14.5 Commissioning (for PTO axis and PN IRT axis only)

14.5.1 Control Panel

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.

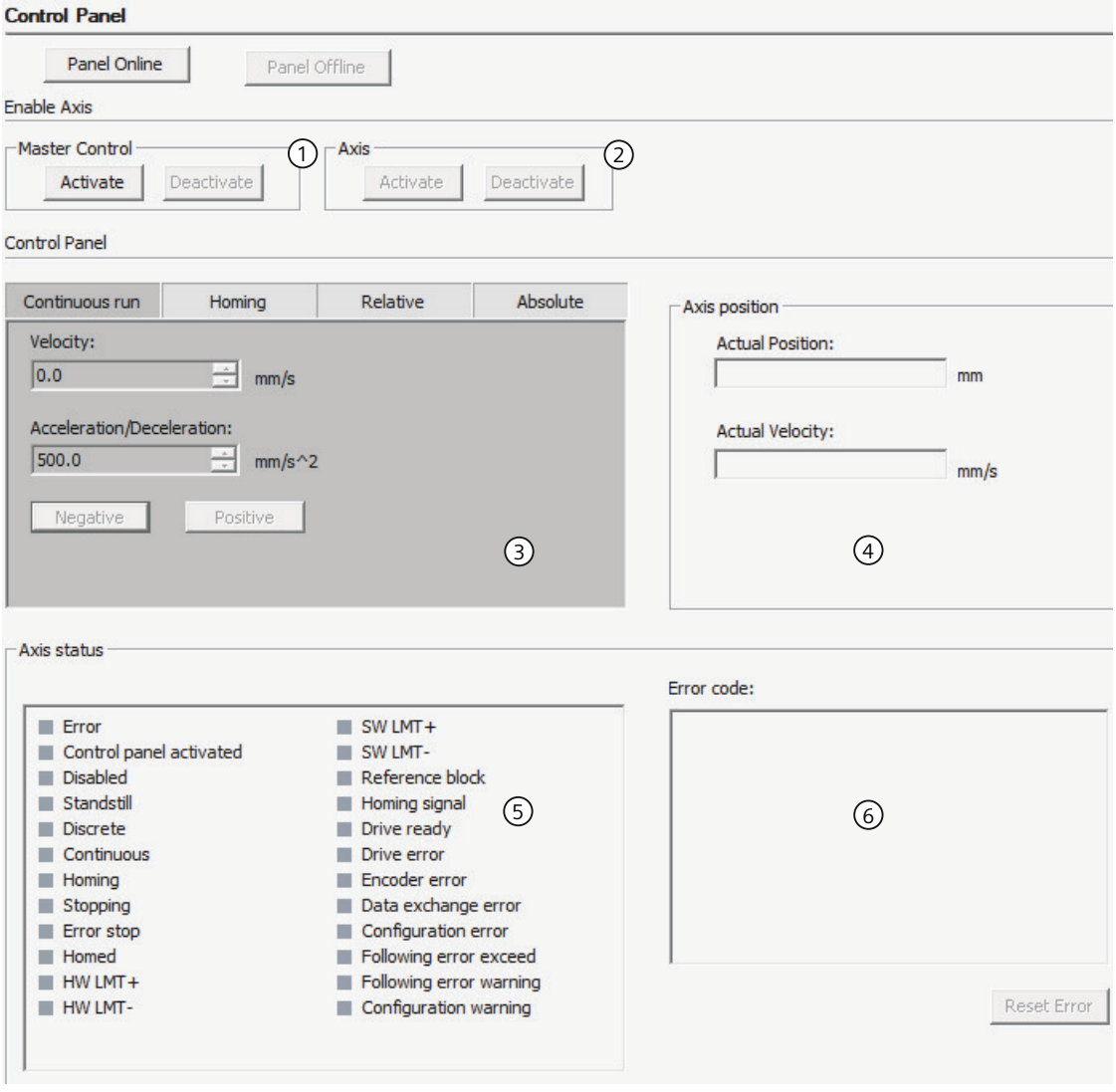
Precondition

Before using the Control Panel, please ensure that the following operations have been completed:

- You have configured the axis and 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 "RUN".

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 with the CPU and take over the axis master control.
• Deactivate: disable the axis control in the Control Panel.

② • Activate: enable the axis, change the axis status from "Disabled" to "StandStill".
• Deactivate: disable the axis, change the axis status from "StandStill" to "Disabled".

③ Control Panel

④ Real-time axis moving status

⑤ Show the current axis status

⑥ Show the error codes (Page 1015) 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.

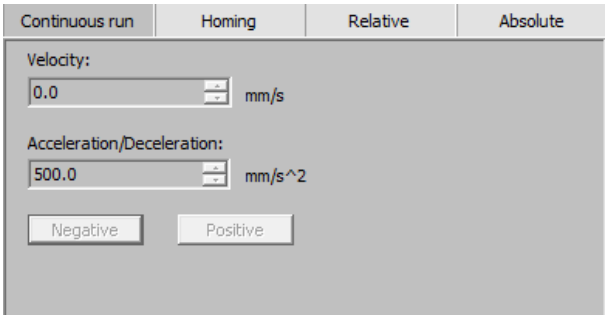
- If no CPU is connected, a pop-up window prompts you to connect the CPU.
- If the CPU is STOP and the currently controlled axis is an IRT axis, a pop-up window prompts you to set the operating state of the CPU to "RUN".
- If the program that is currently open is not the same as the program in the CPU, 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 operation", "Homing", "Relative motion", and "Absolute motion" motions. The real-time position and velocity of the axis can be monitored from the "Axis position" dialog.

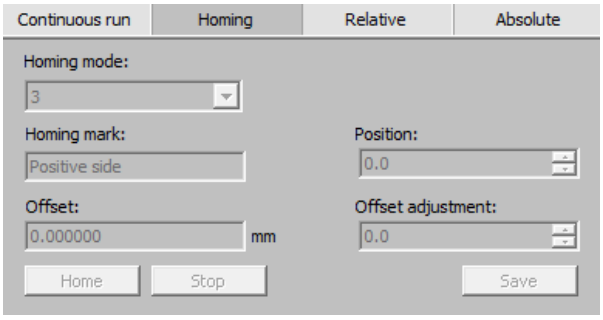
continuous operation



Enter the velocity and the acceleration/deceleration

- 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

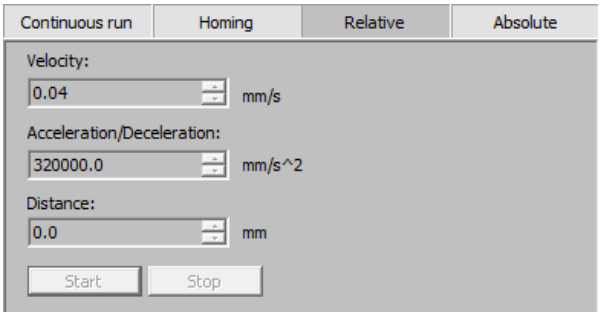


Homing mode/Homing mark/Offset are set in Homing (Page 829). You can adjust the Offset and click "Save".

- Click "Home", the triggered MC_Home instruction passes the value of offset_adjust to the firmware for debugging.
- Click "Stop" to stop moving back to home position.

Relative motion

Absolute motion



Enter the velocity and the acceleration/deceleration

Set the distance for relative movement and position for absolute moment

Click "Start" to start the movement. Click "Stop" to stop the movement.

Note

When an axis is in motion (Continuous\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 (Page 1015) in the Control Panel.

14.5.2 Parameter tuning

This panel is used to tune the position control loop parameters of the IRT axis. Tuning must be performed axis by axis, and only one axis can be run at a time.

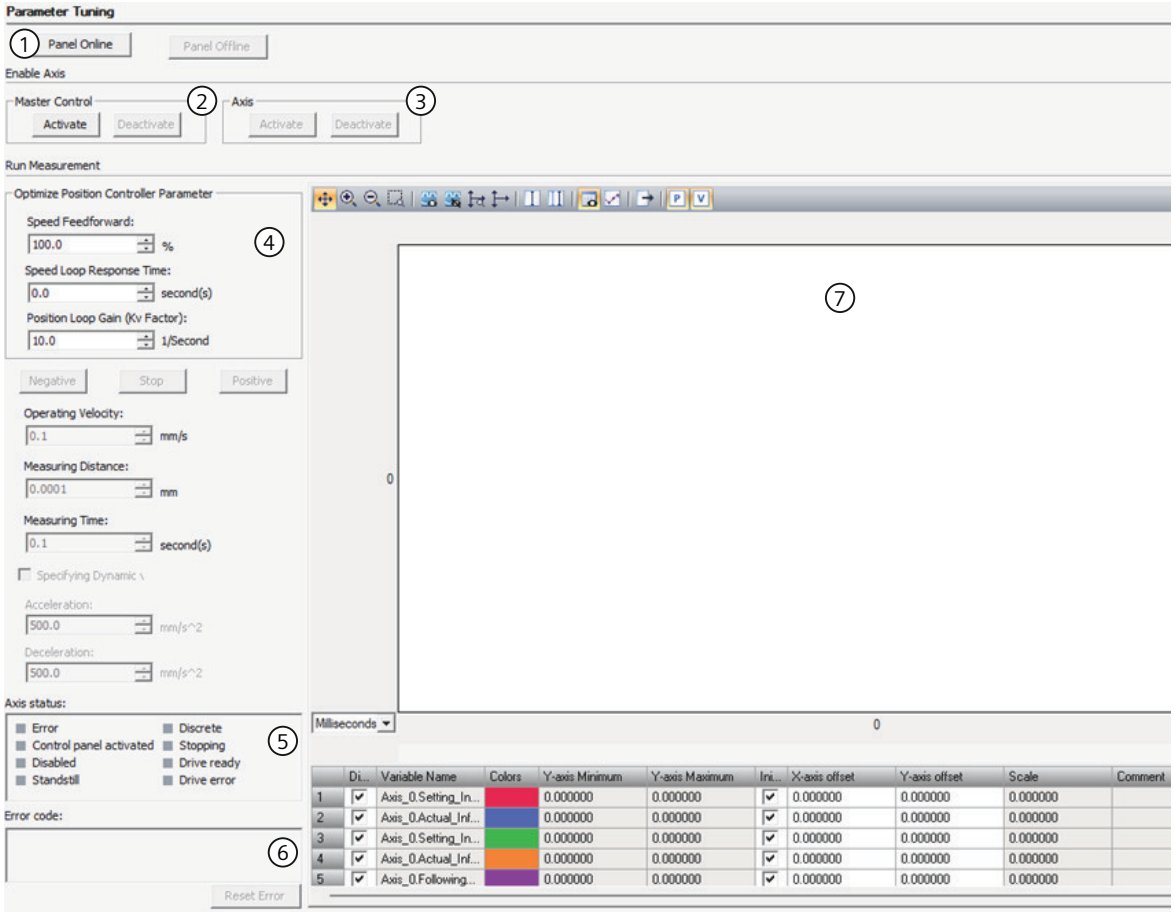
Precondition

Before using "Parameter tuning", ensure that the following steps have been completed:

- The axis configuration has been downloaded to the CPU, and the servo axis has been successfully connected.
- The CPU must be in RUN mode.
- The axis object is deactivated in the user program (MC_Power.Enable = FALSE).
- The axis object has no errors.

Enable parameter tuning

1. Click the tab of the IRT axis you want to debug at the bottom of the program editor.
2. Click "Parameter tuning".



① Panel online status

②
- Activate: Establish an online connection with the CPU and take over the axis master control.
- Deactivate: disable the axis control in the Control Panel.

③
- Activate: enable the axis, change the axis status from "Disabled" to "StandStill".
- Deactivate: disable the axis, change the axis status from "StandStill" to "Disabled".

④ Run measurement[2]

⑤ Show the current axis status

⑥ Show the error codes (Page 1015) that occur while the axis is in motion[3].

⑦ Trace[4]

[1] The online/offline status of the control panel, as well as the button status in the master control activation area and axis command area, is kept synchronized between the control panel and the parameter tuning page.

[2] Before the axis is activated, the input fields and buttons in the 'Run measurement' section will remain uneditable.

[3] If the activated parameter tuning panel reports an error on the axis, you can view the axis status and error code online. Click the "Reset error" button to clear the axis error. After the error is cleared, the axis status will be automatically updated and displayed in the status indicator.

[4] During the tuning process, the available tracking ID range is 129 to 143. The tracking data generated during the tuning process will be automatically stored in SLOT 4.

Activate parameter tuning

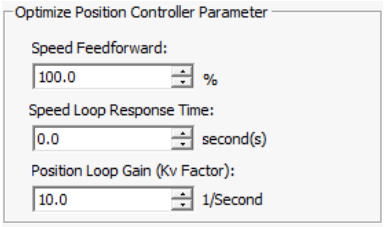
Click the "Activate" button in ③ to activate parameter tuning. If the specified activation conditions are not satisfied, a dialog box will appear, providing step-by-step instructions on the necessary actions to be taken.

- If no CPU is connected, a pop-up window prompts you to connect the CPU.
- If the axis is in the STOP state, a pop-up window prompts you to set the operating state of the CPU to RUN.
- 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.

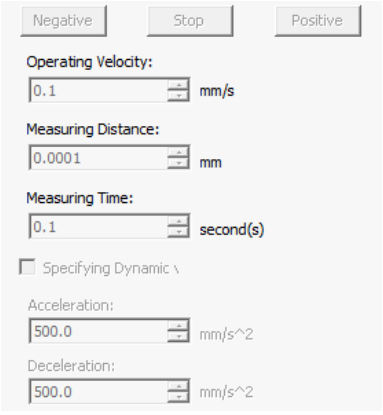
Tuning position control loop parameters

Click the Activate button in ③ to activate the motor drive, the axis state will change from "Disabled" to "Standstill".

Tuning the controller



Run measurement



The forward and reverse buttons are only enabled when the axis is activated and in Standstill. The stop button will always remain enabled when the axis is in any run state (e.g., Discrete, Continuous, Stopping, or Homing). The behavior of this stop button is the same as that of the MC_Stop function.

- Operating velocity: The axis operating velocity used for parameter tuning. Its value range is (0, maximum velocity].
- Measuring distance: During the tuning process, the axis completes a relative movement distance at the set operating velocity.
- Measuring time: This refers to the duration of data trace collection.
- Specifying dynamic value: You can specify dynamic values for the tuning process. When this feature is enabled, the axis will run according to the acceleration and deceleration set by the user. If no dynamic value is specified, the system will use the dynamic parameters set in the axis configuration by default.

Note

Trace supports recording up to 32,768 points. With an IRT refresh cycle of 1 ms, the system will adjust the sampling frequency according to the measuring time you set in order to capture as much data as possible within the maximum number of points limit. (If the number of points exceeds 32,768, the sampling frequency will be reduced so that the number of points does not exceed 32,768).

The following are the methods for calculating the sampling frequency and sampling points:

- Original sampling points = Measuring time / IRT cycle (rounded up)
- Factor by which the sampling frequency increase = Original sampling points / 32,768 + x (if the original sampling points are a multiple of 32768, then x = 0; otherwise, x = 1).
- Final sampling frequency = Factor by which the sampling frequency increase * IRT cycle
- Final sampling points = Measuring time / Final sampling frequency (rounded up)

14.6 General mechanism

PLCopen standardized the motion control function blocks and defined the language programming interface per IEC 61131-3 (JIS B 3503) specification. The S7 200 SMART G2 motion control function conforms to the motion control function defined by PLCOpen, with some extensions. In addition to functions such as single-axis positioning, electronic cam function, and multi-axis interpolation control, the basic steps of the start instructions are also defined.

By using PLCopen motion control function blocks, the dependence on hardware can be reduced, and the reusability of user programs can be improved. It can also reduce the costs of training and support.

Instruction execution

In function block-based motion control programming, the execution of each motion instruction depends on its corresponding function block instance. Each motion instruction corresponds to a unique function block instance when it is used in the program. Each function block instance stores the input parameters and output results data of the motion instruction. When a motion instruction is invoked within a program organization unit, it is executed as follows:

1. Input parameter passing: The input parameters provided when calling the instruction (such as target position, velocity, enable signal, etc.) will be passed and stored in the corresponding input pin data of the FB Instance.
2. Instruction triggering and execution: Once the input conditions are met, the internal logic of the FB Instance will be triggered, and the corresponding motion control task will begin to be executed.
3. Output status feedback: The execution status, current position, error code, or completion flag of the instruction are updated and stored in the output pin data of the FB Instance in real time. This output data can be read and used by other parts of the program.

Multiple instances of the same motion instruction can be created.

General mechanism of output pins

Note

The following table describes the general mechanism of the FB instance output pins. If any particular instruction exhibits unusual behavior, always refer to the description of that instruction.

Output mutual exclusion	• In Execute mode: Outputs Busy, Done, Error, and CommandAborted are mutually exclusive: For a given instruction, only one of them can be TRUE. If Execute is TRUE, then one of these outputs must be TRUE. Only one of the outputs Active, Error, Done, and CommandAborted will be set at a time. • In Enable mode: The outputs Status and Error are mutually exclusive. For a given instruction, only one of them can be TRUE.
Output status	• In Execute mode: The Done, Error, ErrorID, and CommandAborted outputs are reset on the falling edge of Execute. However, the falling edge of Execute does not stop or even affect the execution of the actual instruction. If Execute is reset before the instruction completes, the response output is retained for at least one scan cycle. • In Enable mode: The outputs Valid, Enabled, Busy, Error, and ErrorID are reset simultaneously on the falling edge of Enable, so they are not delayed by one scan cycle before resetting.
Behavior of Done output	When the instruction is successfully completed, the Done output is set. When multiple instructions are executed sequentially on the same axis, the following rules apply: If a motion on one axis is interrupted by another motion on the same axis before reaching its final goal, the Done output of the first instruction will not be set.
Behavior of Busy output	• In Execute mode: Each instruction can have a Busy output, indicating that the function block is not yet complete, and a new output value can be expected. Busy is set on the rising edge of Execute and reset when one of the Done, Aborted, or Error outputs is set. • In Enable mode: Each instruction can have a Busy output, indicating that the function block is working and new output values can be expected. Busy is set on the rising edge of Enable and remains set as long as the instruction is performing any action. Busy and Error can both be TRUE. During an error, the instruction continues to be in a working state. Once the error is resolved, Error is automatically reset.
Active output	Active is set when an instruction takes over the motion control of the corresponding axis. Taking over motion control means that an interpolation (IPO) is executed. Active lags behind Busy by at least one POU cycle because the Active output pin is only refreshed when the instruction is scanned.

14.6.1 Instruction restart

Basic description

Preconditions

- The motion control FB instance has been triggered and started by a rising edge signal and is currently in the Busy state (i.e., it is either in the Execute or Buffer state).

Execution process and results

The FB instance supports repeated triggering of the execution instruction, and all input parameters are valid.

The currently running FB parameters will be updated. Meanwhile, the CommandAborted output pin will not be set.

Note

When the Execute pin of the MC_TouchProbe instruction is set high, if the axis ID is changed during this period, re-triggering the instruction will perform the probe function for the new axis ID.

Examples

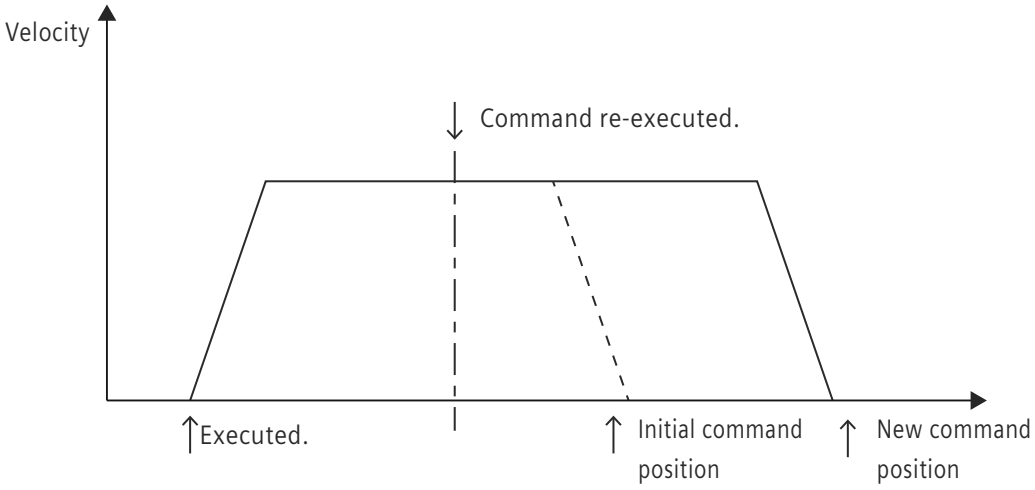
Preconditions

- The initial position of the axis is 0.

Execution process and results

Execution conditions	Results
1. MoveRelative is triggered by a rising edge signal. The motion parameters are set as follows: Distance 100, Velocity 5, Acceleration 5, Deceleration 5, and Jerk 0. 2. When the axis reaches position 50 during the aforementioned motion, MoveRelative receives another rising edge trigger signal. At this point, the new motion parameters are set as follows: Distance 150, Velocity 5, Acceleration 5, Deceleration 5, and Jerk 0。	• The axis first accelerates from a stationary state to a velocity of 5 with an acceleration of 5. • When the axis reaches Position 50, the motion command restarts. The axis then moves at a constant speed of 5, exceeding the initial Target position 100, and finally stops at Position 150.

Restart instruction during motion at constant velocity



Motion instruction restart and abort

The motion instruction restart is similar in behavior to the Abort instruction. Both will re-plan the motion based on the axis's current real-time motion state (including current velocity, acceleration, and deceleration) and combined with the newly specified motion parameters (i.e., the updated target velocity, acceleration, deceleration, and target displacement). Therefore, the changes in the motion curves for both are basically consistent.

However, there is a difference between the two at the reference point of the target displacement.

- Motion instruction restart: Its target displacement is always calculated relative to the original starting point of the previous motion.
- Interrupts: Planning is based on the current axis position at the time the instruction is triggered.

Instructions and parameters that support single instance restart

The instruction is re-executed by re-triggering the function block instance only after the updatable parameters (i.e., the parameters listed below) have been modified. If a non-updatable parameter is modified, the system will ignore the trigger request even if the function block instance is re-triggered.

Single axis

Instructions	Position/Distance	Velocity	Acceleration	Deceleration
MC_MoveAbsolute	Y	Y	Y	Y
MC_MoveRelative	Y	Y	Y	Y
MC_MoveVelocity		Y	Y	Y
MC_Stop				Y
MC_Halt				Y

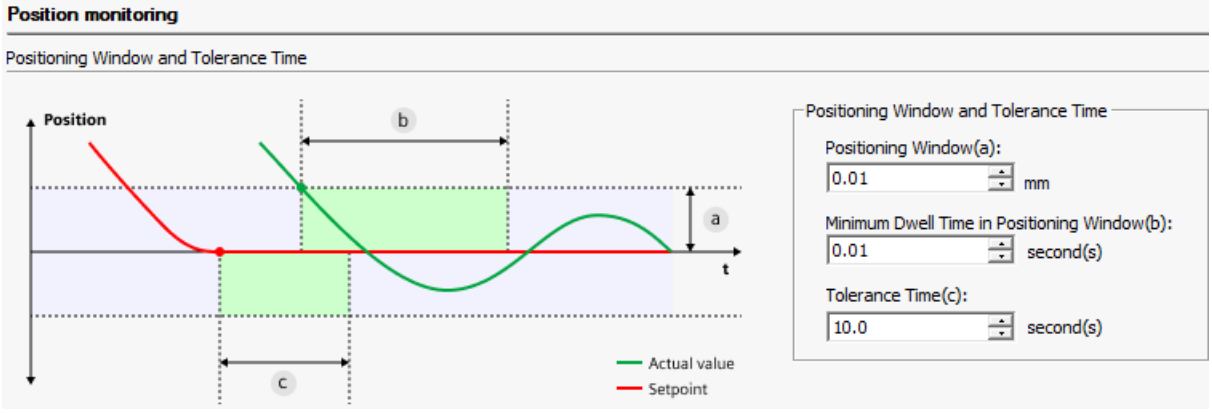
Cam and gear instructions

- MC_CamIn: Only the CamTable on the pins can be modified.
- MC_GearIn: Only the pins RatioNumerator, RatioDenominator, Acceleration, and Deceleration。 can be modified

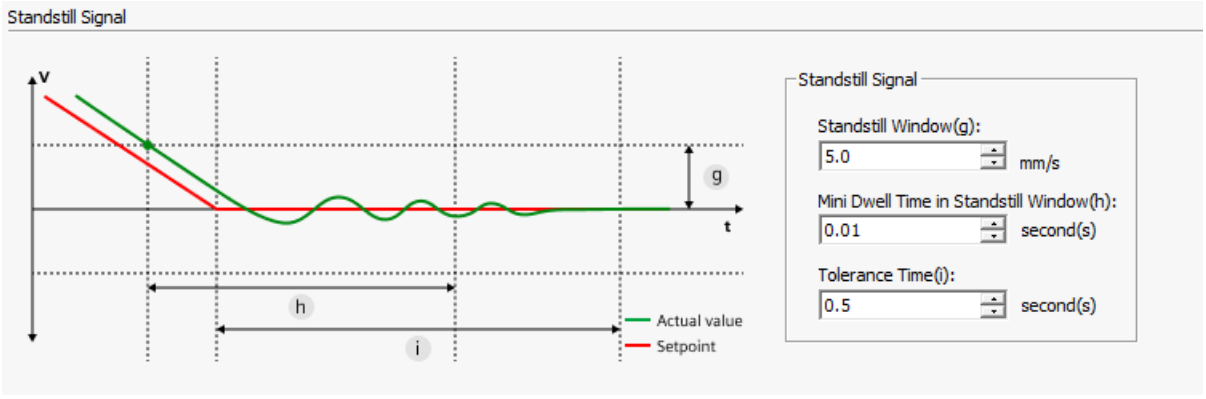
Ineffective cases

Even parameters that support restarting may not take effect under certain conditions.

- Updates to the target position are ignored if they occur after the given value has been reached. Taking the IRT axis as an example, as shown in the figure below, if the axis has already entered segment C for position monitoring, the restart for target position update will be ignored.



- For velocity control instructions, such as MC_Stop, if the given velocity has already reached 0, then a restart will not be initiated. Taking the IRT axis as an example, once velocity monitoring is enabled, it will no longer respond to restarts.

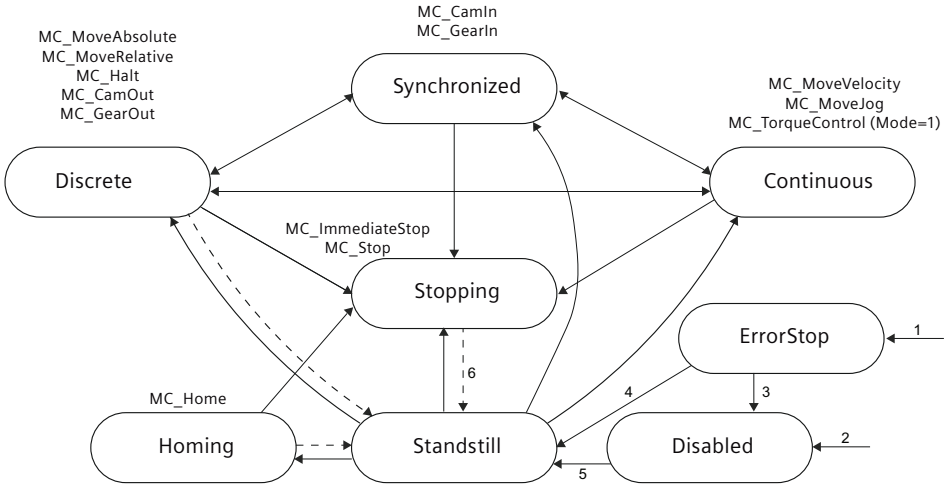


14.6.2 State Machine

14.6.2.1 Single axis state machine

Single axis state

The motion of the axis after starting the corresponding motion control instruction for the axis is shown in the figure below.
The motion control instructions are executed sequentially, and the axis is in one 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_ImmediateStop).Done=TRUE and MC_Stop(MC_Immediate-Stop).Execute=FALSE

Status		Description
Servo OFF	Disabled	Axisxis is in the state of servo off, stop, or ready to start.
	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 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 MC_Stop is Execute after stopping with the TRUE 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.

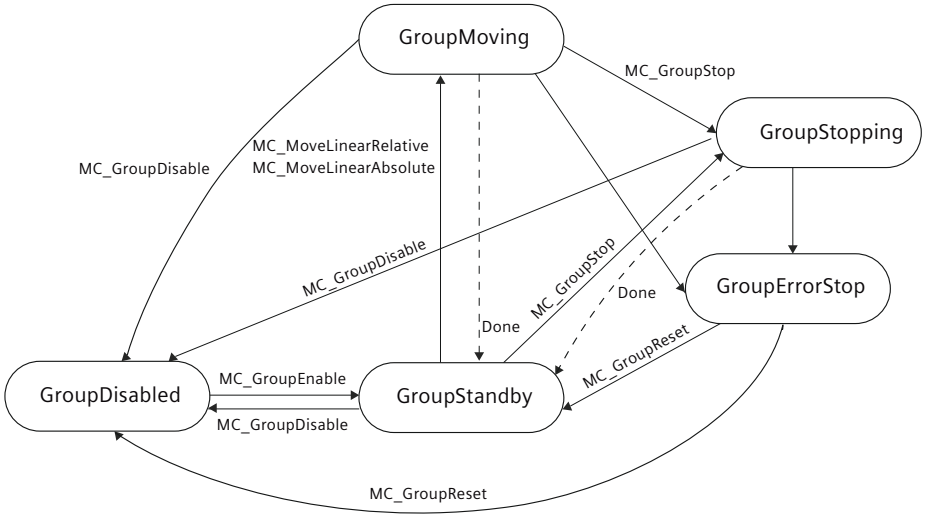
Note

All motion control instructions cannot be called in an interrupt routine.

14.6.2.2 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.



GroupErrorStop	Enter this state when the axes group' fault detection logic detects a fault
GroupStandby	• 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
GroupDisabled	• 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_GroupReset • When calling MC_GroupDisable while axes group is moving, stopping or in standby, axes group enters the state GroupDisabled
GroupMoving	Call the axis group motion instructions when the axis group is in the GroupStandby state
GroupStopping	When calling MC_GroupStop while axes group is in the GroupStandby or GroupMoving 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.

14.6.3 Buffered mode

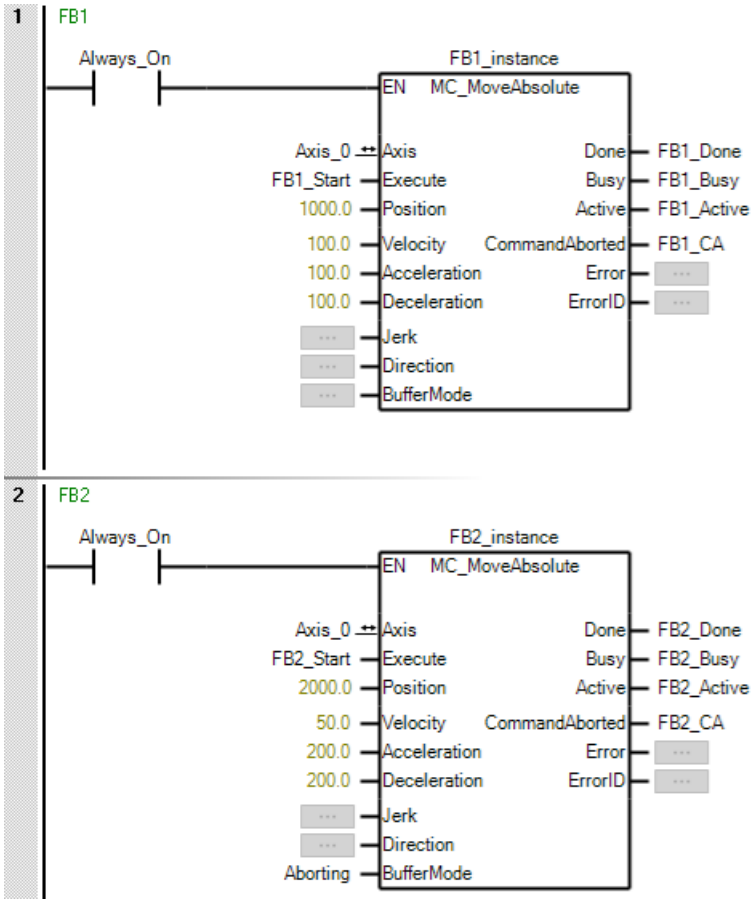
14.6.3.1 Single-axis command buffered mode

The following table gives an overview of single-axis command buffered mode.

S7-200 SMART 3.0 CPU and S7-200 SMART G2 CPU can buffer up to 8 instructions.

Instructions	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	NO	--
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	YES	InVelocity
MC_CamIn	YES	YES	EndOfProfile
MC_CamOut	NO	NO	--
MC_GearIn	YES	Yes	InGear
MC_GearOut	NO	NO	--
MC_PhasingRelative	NO	NO	--
MC_ImmediateStop	NO	NO	--
MC_MoveJog	NO	NO	--

Example: Aborting motion



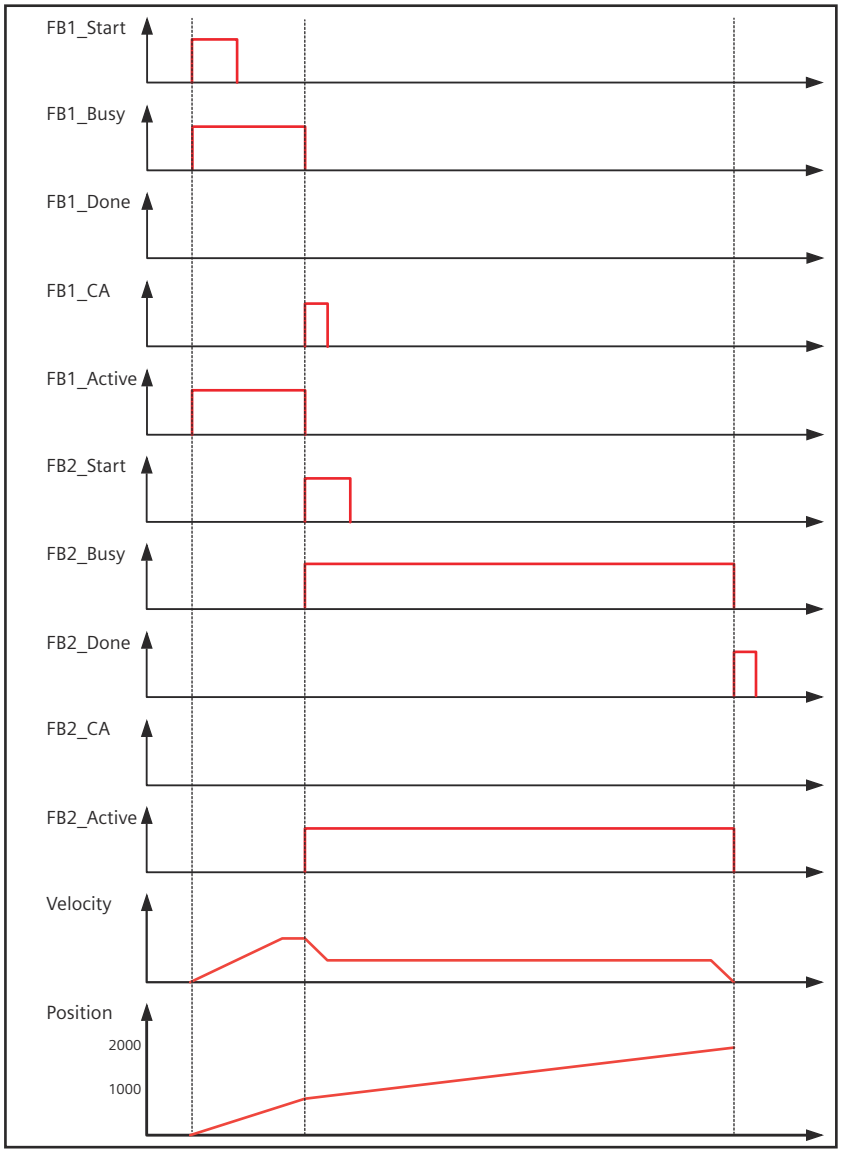
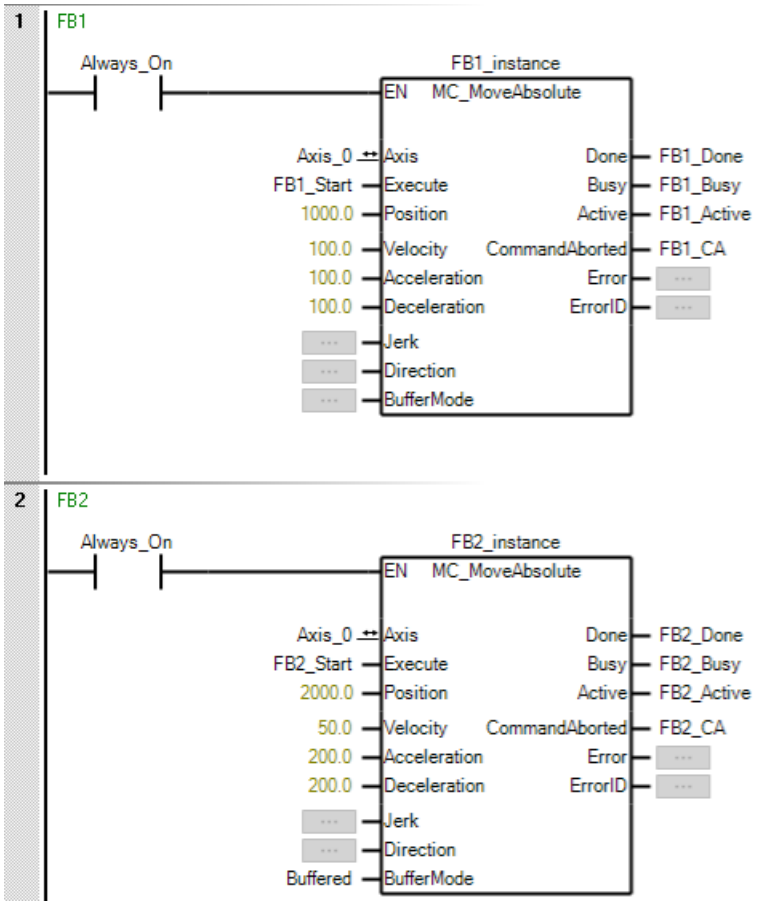


Figure 14-2 Timing diagram for example above with FB2 interrupting FB1 (Executing FB2 immediately from the breakpoint)

Example: Buffered motion



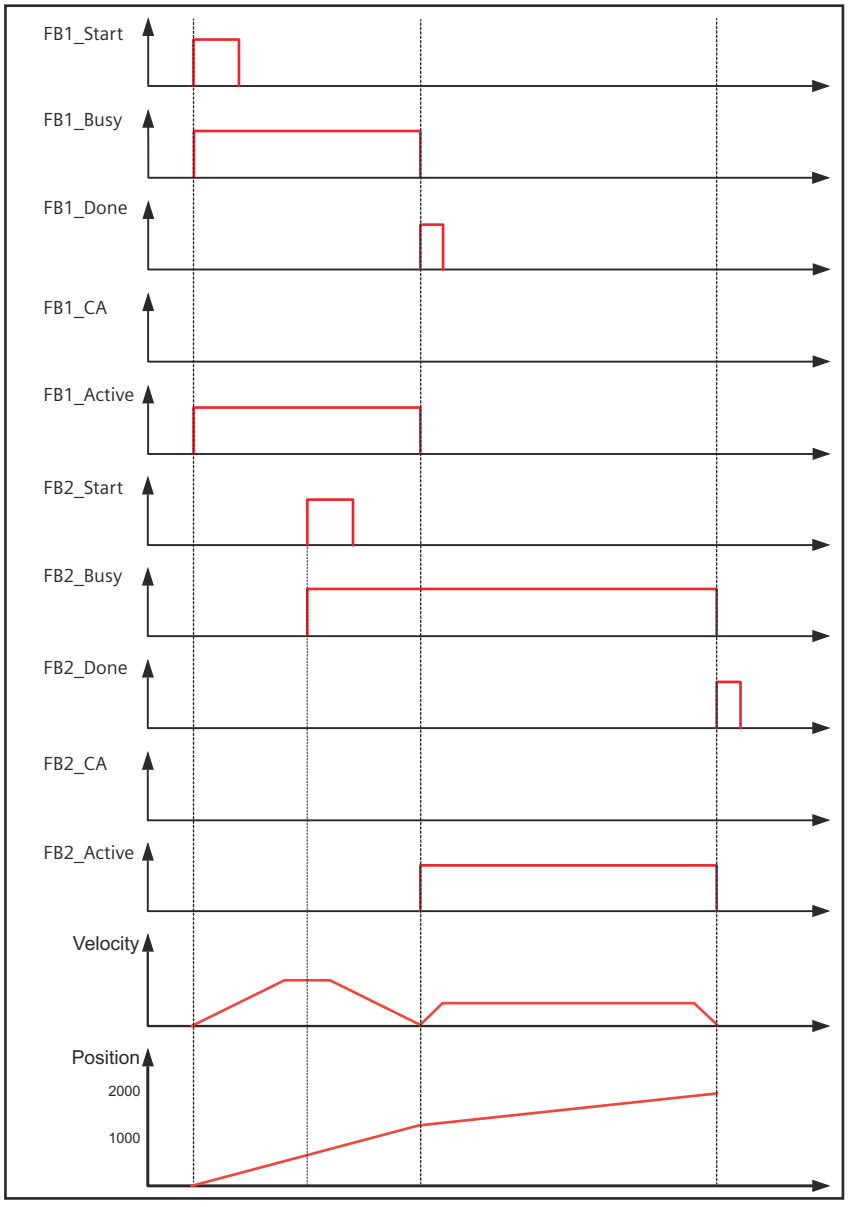


Figure 14-3 Timing diagram for example above in Buffered Mode (Stopping to velocity 0 and starting FB2 at that point without delay)

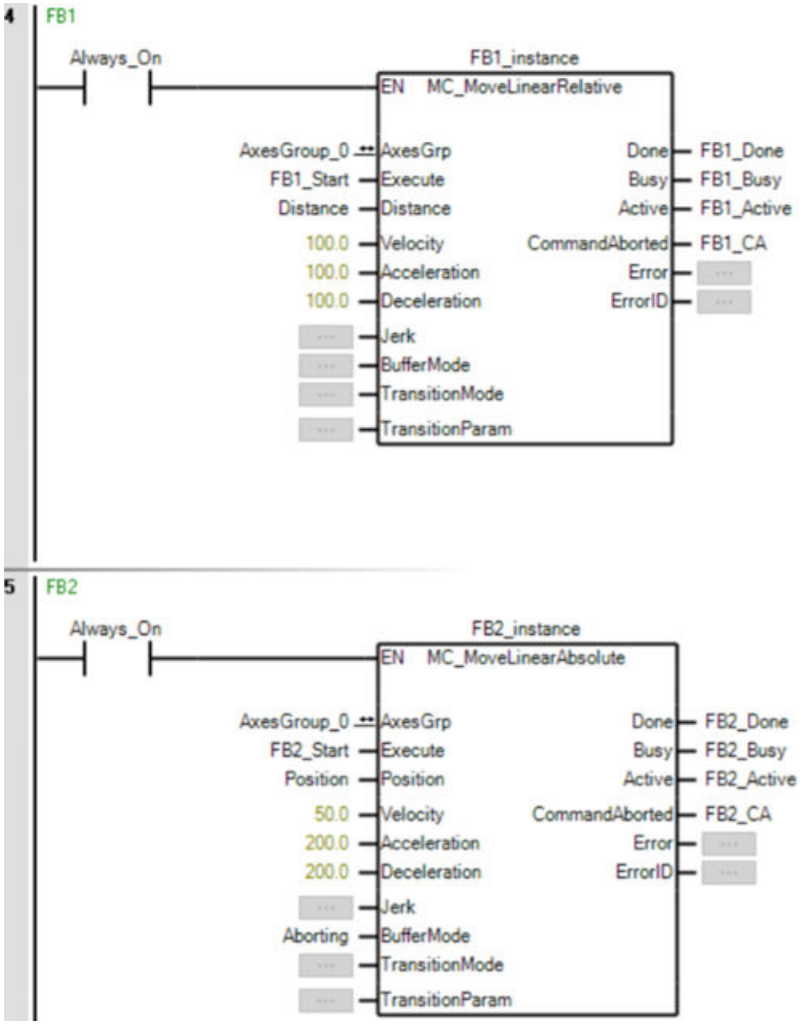
14.6.3.2 Axes group command buffered mode

The following table gives an overview of axes group command buffered mode.

S7-200 SMART 3.0.1 CPU and S7-200 SMART G2 CPU can buffer up to 8 instructions.

Instructions	Can be specified as a buffered command	Can be followed by a buffered command	Relevant signal to activate the next buffered instruction
MC_GroupEnable	NO	NO	--
MC_GroupDisable	NO	NO	--
MC_GroupReset	NO	NO	--
MC_GroupStop	NO	NO	--
MC_MoveLinearAbsolute	YES	YES	Done
MC_MoveLinearRelative	YES	YES	Done

Example: Aborting motion



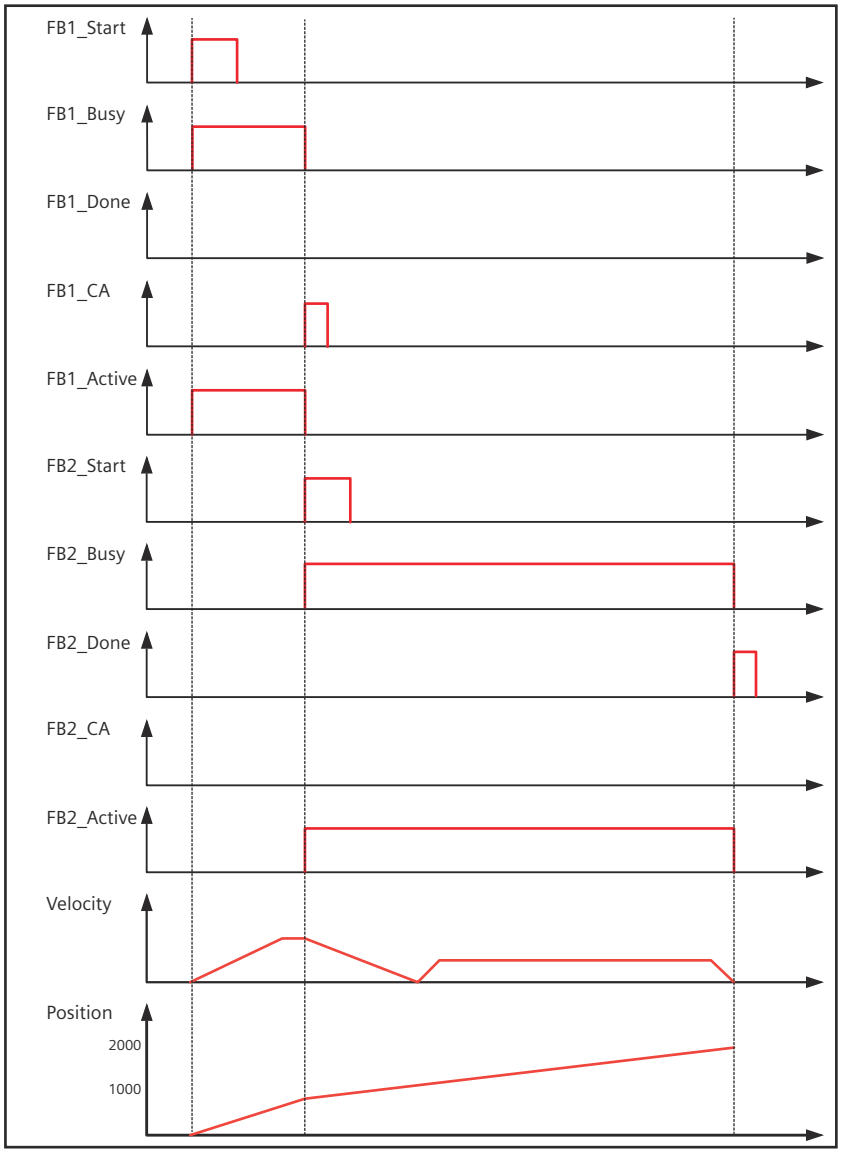
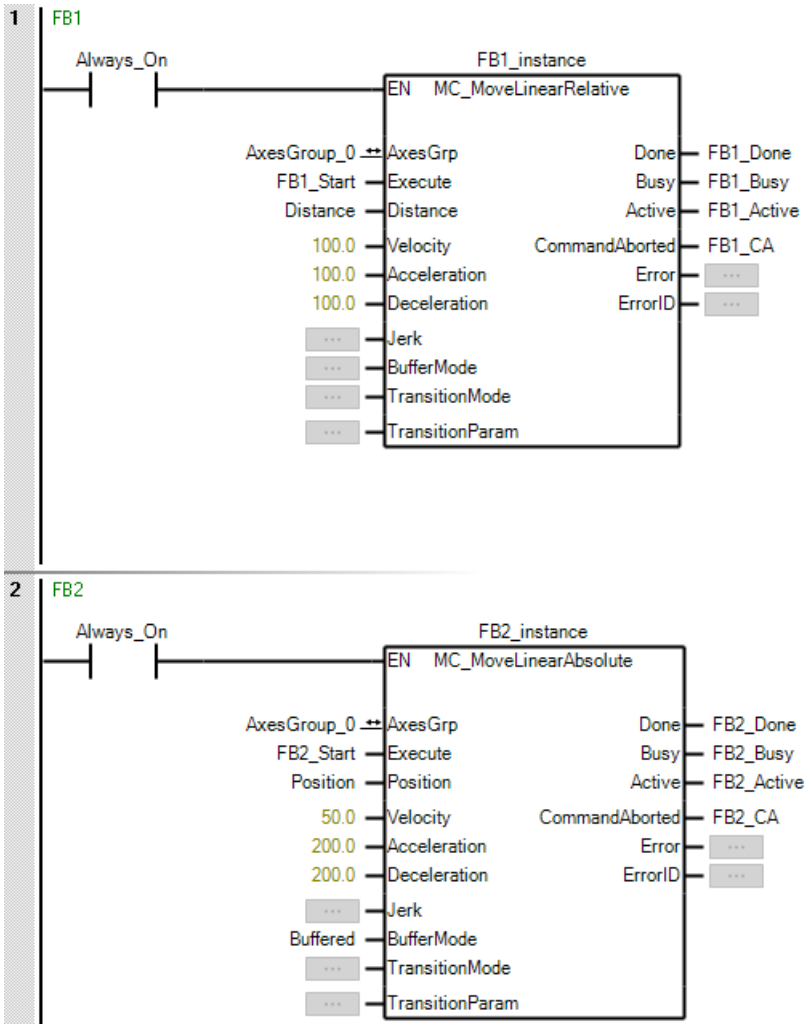


Figure 14-4 The timing diagram of FB2 interrupting FB1 in the example above (the axis decelerates to a stop as per FB1, and moves to the target coordinate as per FB2 parameters)

Example: Buffered motion



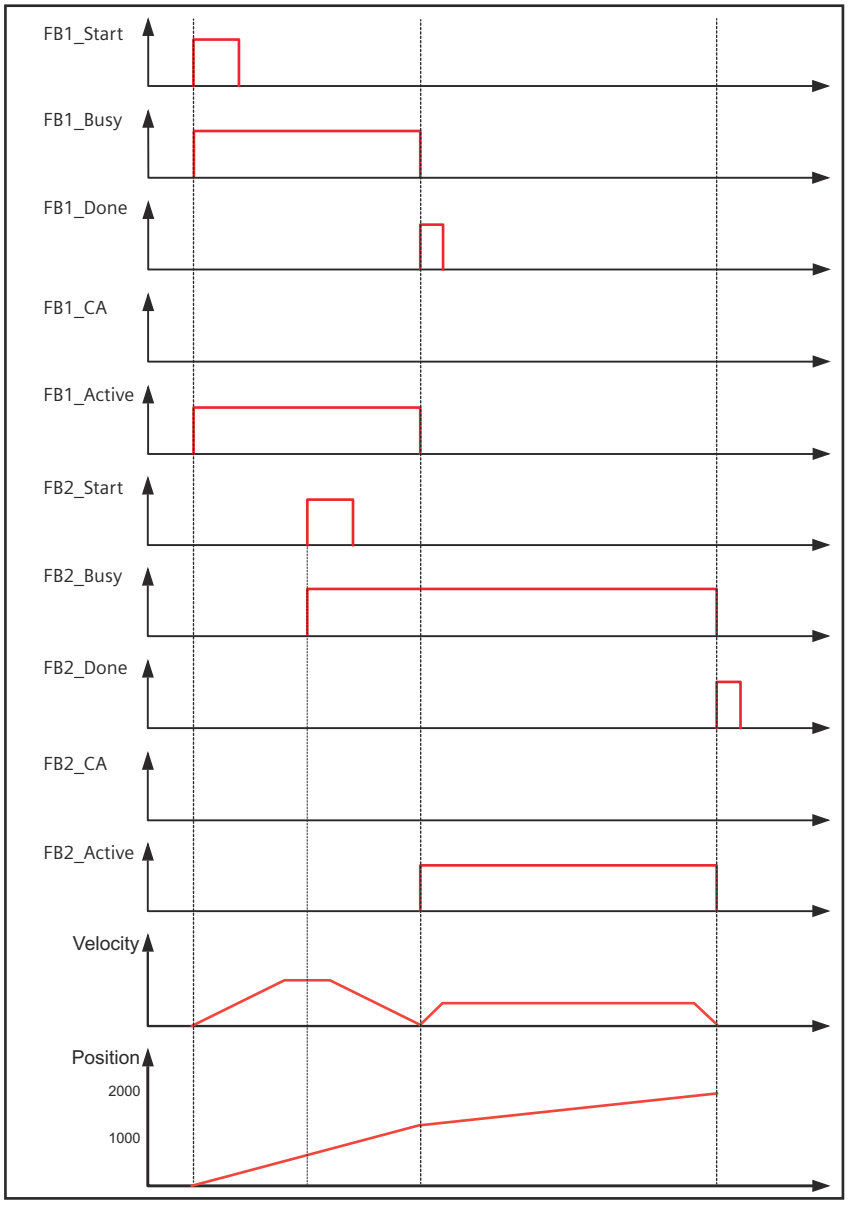


Figure 14-5 Timing diagram for example above in Buffered Mode (Stopping to velocity 0 and starting FB2 at that point without delay)

14.6.4 Overview of variable

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 variable
  Variables used for handling servo drives, virtual servo drives, and encoders.
  - STEP 7-Micro/WIN SMART V3 created variable names: Axis_0 ~ Axis_15

  Note

  Use MC type array to set MC instructions

  When adding an axis, the system will assign it to the corresponding axis array based on the axis number. For example, Axis_1 corresponds to axis number 1, and this axis will be assigned to AXIS_ARRAY[1].

  When using a "FOR" loop to execute MC instructions, make sure you use an MC type array to set these instructions.

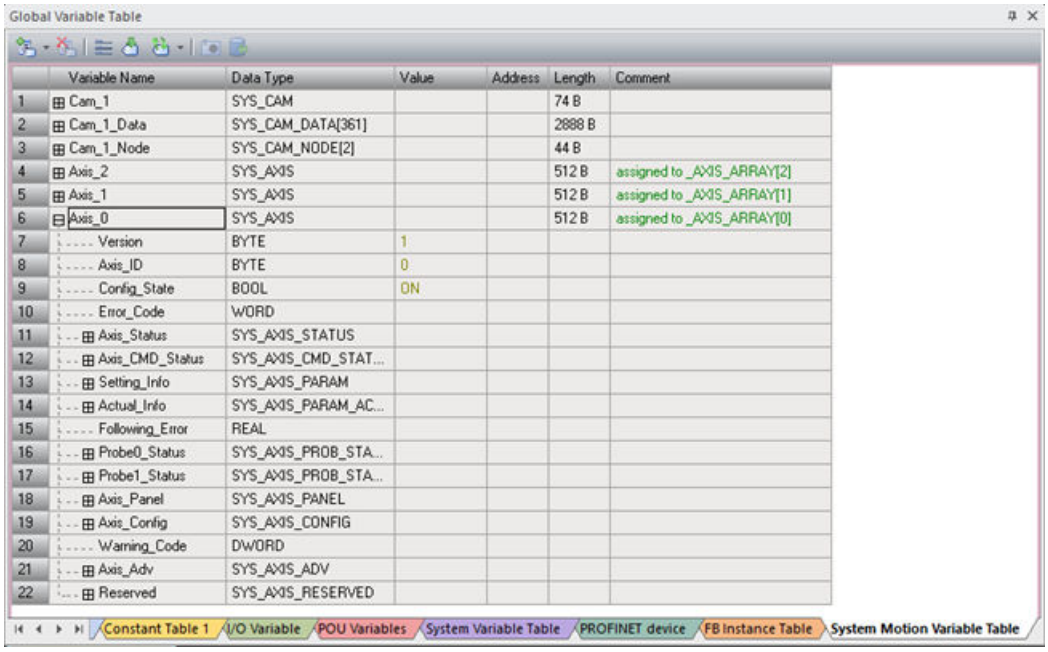
  Because each MC instruction needs to be associated with an instance. If you do not use an MC type array for the setting, the program's execution results can be unpredictable.

  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



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
Hold	No

14.6.5 Variable overview table

Motion control 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).

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.

Variables	Data type	Description
Version	BYTE	TO version number
Axis_ID	BYTE	Axis number
Config_State	BOOL	Axis configuration status
Error_Code	WORD	Axis error code

Variables			Data type	Description
Axis_Status.			SYS_AXIS_STA-TUS	
	ENC_Enable		BOOL	For encoder axis use only • 0: Not enabled • 1: Enable
	Error		BOOL	It becomes TRUE when there is a PN communication error/ motion error/configuration error.
	ControlPane-lActive		BOOL	Indicates whether the control panel is active • 0: Is Inactive • 1: Is active
	Disabled		BOOL	It becomes TRUE when the axis is not enabled and has no fault.
	Standstill		BOOL	When it becomes TRUE, it means: • The axis has been enabled • The axis has no fault • No motion instructions were activated on the axis.
	Discrete		BOOL	It becomes TRUE while performing a discrete motion.
	Continuous		BOOL	It becomes TRUE while performing continuous motion.
	Synchronized		BOOL	It becomes TRUE when performing synchronized motion.
	Homing		BOOL	It becomes TRUE when performing homing.
	Stopping		BOOL	It becomes TRUE when decelerating to a stop.
	ErrorStop		BOOL	It becomes TRUE when the system detects an axis-related error. In this state, the axis may be enabled or disabled, and can be controlled by the MC_Power instruction. However, as long as the error is not cleared, the axis will remain in the ErrorStop state. The core objective of the ErrorStop state is to bring the axis to a stop as safely as possible. Once in this state, the system will not accept any subsequent motion instructions until a reset operation is successfully performed.
	Homed		BOOL	It becomes TRUE when the home is determined, and the MC_home instruction has been executed.
	LMTpHW_State		BOOL	It becomes TRUE when the positive hardware limit is reached.
	LMTmHW_State		BOOL	It becomes TRUE when the negative hardware limit is reached.
	LMTpSW_State		BOOL	It becomes TRUE when the positive software limit is reached.
	LMTmSW_State		BOOL	It becomes TRUE when the negative software limit is reached.
	RPS_State		BOOL	It becomes TRUE when the home point search signal is reached.
	STP_State		BOOL	It becomes TRUE when the emergency stop signal is reached (Only for PTO axis).
	RefBlock_State		BOOL	Whether the Reference stop has been reached • 0: No • 1: Yes

Variables			Data type	Description
	DIInput_State		BOOL	Whether the digital input point has been reached • 0: No • 1: Yes
	InClamping		BOOL	The drive is kept at the fixed stop (clamping). The axis position is within the positioning tolerance.
	VelLimit		BOOL	During synchronization, the slave axis exceeds its set maximum velocity.
	Reserved4		BOOL	Reserved
	Reserved5		BOOL	Reserved
	Drive_Ready		BOOL	When it becomes TRUE, it means: • Cyclic communication between the controller and the drive has been established • The drive has no errors • The drive is configured correctly.
	Drive_Err		BOOL	It becomes TRUE when a drive error occurs.
	Encoder_Err		BOOL	It becomes TRUE when an encoder error occurs.
	Data_Exchange_Err		BOOL	During operation, PN disconnection or drive SOL consecutive timeouts changes to TRUE.
	Config_Err		BOOL	When the following combined conditions are met, the axis will enter an error state (become TRUE) and wait for the PLC to re-download the configuration: • Failed to read drive parameters • The system failed to read the parameters after three retries. • MW has not been configured with offline parameters
	Following_Err_Exceed		BOOL	It becomes TRUE when the following error exceeds the preset threshold.
	Following_Err_Warning		BOOL	When the following error exceeds the preset threshold, the system will trigger an out-of-tolerance warning. The triggering conditions for this warning are related to the system configuration. At this point, the axis will not enter an error state, nor will the Error bit be set.
	Config_Warning		BOOL	It becomes TRUE when the system detects that the online parameters of the drive are inconsistent with the offline parameters, and the system will not set the Error bit.
Axis_CMD_Status.			SYS_AXIS_CMD_STATUS	
	Power_CMD		BOOL	It becomes TRUE while the MC_Power instruction is being executed.
	MoveVelocity_CMD		BOOL	It becomes TRUE while the MC_MoveVelocity instruction is being executed.
	MoveRelative_CMD		BOOL	It becomes TRUE while the MC_MoveRelative instruction is being executed.
	MoveAbsolute_CMD		BOOL	It becomes TRUE while the MC_MoveAbsolute instruction is being executed.
	Reset_CMD		BOOL	It becomes TRUE while the MC_Reset instruction is being executed.

Variables			Data type	Description
	Home_CMD		BOOL	It becomes TRUE while the MC_Home instruction is being executed.
	Stop_CMD		BOOL	It becomes TRUE while the MC_Stop instruction is being executed.
	Halt_CMD		BOOL	It becomes TRUE while the MC_Halt instruction is being executed.
	ImmediateStop_CMD		BOOL	It becomes TRUE while the MC_ImmediateStop instruction is being executed.
	Reserved_CMD		BOOL	Reserved
	TouchProbe_CMD		BOOL	It becomes TRUE while the MC_TouchProbe instruction is being executed.
	AbortingTrigger_CMD		BOOL	It becomes TRUE while the MC_AbortingTrigger instruction is being executed.
	GearIn_CMD		BOOL	It becomes TRUE while the MC_GearIn instruction is being executed.
	GearOut_CMD		BOOL	It becomes TRUE while the MC_GearOut instruction is being executed.
	MoveJog_CMD		BOOL	It becomes TRUE while the MC_MoveJog instruction is being executed.
	CamIn_CMD		BOOL	It becomes TRUE while the MC_CamIn instruction is being executed.
	CamOut_CMD		BOOL	It becomes TRUE while the MC_CamOut instruction is being executed.
	PhasingRelative_CMD		BOOL	It becomes TRUE while the MC_PhasingRelative instruction is being executed.
	TorqueRange_CMD		BOOL	It becomes TRUE while the MC_TorqueRange instruction is being executed.
	TorqueControl_CMD		BOOL	It becomes TRUE while the MC_TorqueControl instruction is being executed.
	TorqueLimiting_CMD		BOOL	It becomes TRUE while the MC_TorqueLimiting instruction is being executed.
	Reserved		BOOL	Reserved
Setting_Info.			SYS_AXIS_PARAM	
	Pos		REAL	Position setpoint
	Vel		REAL	Velocity setpoint
	Acc		REAL	Acceleration setpoint
	Decel		REAL	Deceleration setpoint
	Jerk		REAL	Jerk setpoint
	Reserved		REAL	Reserved
	Torque		REAL	Torque setpoint
Actual_Info.			SYS_AXIS_PARAM_ACTUAL	
	Pos		REAL	Actual position
	Vel		REAL	Actual velocity
	Acc		REAL	Actual acceleration

Variables			Data type	Description
	Decel		REAL	Actual deceleration
	Reserved0		REAL	Reserved
	Torque		REAL	Actual torque
	Reserved1		REAL	Reserved
Following_Error			REAL	Following error of the current axis
Probe0.Status.			SYS_AXIS_PROB_STATUS	
	Enable		BOOL	Whether probe 0 is used • 0: No probes used • 1: Probe used
	RisingEdgeDone		BOOL	
	FallingEdgeDone		BOOL	
Probe1.Status.			SYS_AXIS_PROB_STATUS	
	Enable		BOOL	Whether probe 1 is used • 0: No probes used • 1: Probe used
	RisingEdgeDone		BOOL	
	FallingEdgeDone		BOOL	
Axis_Panel.			SYS_AXIS_PANEL	
	Target_Pos		REAL	Position setpoint
	Target_Vel		REAL	Velocity setpoint
	Target_Acc		REAL	Acceleration setpoint
	Target_Dir		BYTE	Direction setpoint
	Home_Rps_Offset		REAL	Homing offset
	Error_Code		WORD	Control panel motion error
	Reserved0		REAL	Reserved
	Reserved1		DWORD	Reserved
Axis_Config.				
	Name		STRING	Name of the axis
	Axis_Type		BYTE	Axis type • Virtual axis: 0 • PTO Axis: 1 • Encoder Axis: 2 • PN IRT axis: 3

Variables			Data type	Description
	Engineer_Sys		BYTE	• cm: 0 • mm: 1 • Inch: 2 • Degree: 3 • pulse: 4
	Pulse_Of_One-Motor		DWORD	The number of pulses in one rotation of the PLS/REV motor/ encoder
	Units_Revolu-tion		REAL	The movement of the work table in one rotation
	Units		STRING	Units
	Gear_Numera-tor		DWORD	Gear numerator
	Gear_Denomi-nator		DWORD	Gear denominator
	Linear_Rotary		BYTE	• Rotary: 0 • Linear: 1
	Modu-lo_StartVal		REAL	Modulo start value
	Modu-lo_Length		REAL	Modulo length
	Reserved0		REAL	Reserved
	Posi-tionLmt_SW.		SYS_AX-IS_SOFTLMT	
		SwLimit_En	BOOL	Enable software limit • 0: Software limit not enabled • 1: Software limit enabled
		MaxLi-mit_Resp	BYTE	Software positive limit response (decelerated stop/immediate stop mode)
		MinLi-mit_Resp	BYTE	Software negative limit response (decelerated stop/immediate stop mode)
		MaxLimit_Pos	REAL	Software positive limit position
		MinLimit_Pos	REAL	Software negative limit position
	Dynamic_De-fault.		SYS_AXIS_DY-NAMIC_DE-FAULT	
		Default_Acc	REAL	Default acceleration
		Default_Decel	REAL	Default deceleration
		Default_Jerk	REAL	Default jerk
		Reserved0	REAL	Reserved
		Reserved1	REAL	Reserved
	Dynamic_Lim-it.		SYS_AXIS_DY-NAMIC_LIMIT	
		Max_Vel	REAL	Max Velocity
		Max_Acc	REAL	Maximum acceleration
		Max_Decel	REAL	Maximum deceleration

Variables			Data type	Description
		Reserved0	REAL	Reserved
		Reserved1	REAL	Reserved
	Reserved1		REAL	Reserved
	Reserved2		REAL	Reserved
	Reserved3		REAL	Reserved
	Reserved4		REAL	Reserved
Reserved			DWORD	
Axis_Config_Adv.			SYS_AXIS_CONFIG_ADV	Extended parameters
	Pos_Monitoring.		SYS_AXIS_POS_MONITOR	
		Tolerance_Time	REAL	**Positioning window tolerance time** The tolerance time is the maximum permissible duration from when the slave axis speed reaches the setpoint of 0 until its actual position enters the positioning window. The maximum permissible time interval from reaching the velocity setpoint of zero to entering the positioning window.
		MinDwell_Time	REAL	Minimum dwell time in positioning window
		Window	REAL	Positioning window
		Reserved0	REAL	Reserved
	Following_Err.		SYS_AXIS_FOLLOWING_ERR	
		Enable	BOOL	• Following error monitoring not enabled: 0 • Following error monitoring enabled: 1
		Reserved1	BYTE	
		Min_Value	REAL	Permissible following error when the velocity is below the Dynamic_Vel value.
		Max_Value	REAL	The maximum permissible following error upon reaching maximum speed.
		Dynamic_Vel	REAL	Min_Value is allowed when the velocity is below this value and remains constant.
		Warning_Level	REAL	Warning limit as a percentage value relative to the maximum permissible following error (velocity-dependent in accordance with the characteristic curve).
		Reserved2	REAL	
	Standstill_Signal.		SYS_AXIS_STANDSTILL	
		Vel_Threshold	REAL	Dwell time startup speed threshold If velocity is below this threshold, then the minimum dwell time begins.
		MinDwell_Time	REAL	Specifies the minimum dwell time
		Tolerance_Time	REAL	Reserved

Variables			Data type	Description
	Pos_Ctrl.		SYS_AX-IS_POS_CTRL	
		Kv	REAL	Proportional gain of the closed loop position control ("Kv"> 0.0)
		Kpc	REAL	Velocity precontrol for position control. Below are the recommended settings: • Isochronous drive interface via PROFIdrive: 100.0% • Non-isochronous drive interface via PROFIdrive: 0.0 to 100.0% • Analog drive interface: 0.0 to 100.0%
		Vtc	REAL	
	Extrapolation.		SYS_AXIS_EX-TRAPOLATION	
		Differentia-tion	BOOL	Activate derivative • When this checkbox is selected: The leading value velocity results from the differentiation of the extrapolated leading value position. • When this checkbox is cleared: The filtered actual velocity is applied.
		Leading_Axis	BOOL	Include leading axis • When this checkbox is selected: The leading axis-dependent extrapolation time will be included in the calculation of the effective extrapolation time. • When this checkbox is cleared: The leading axis-dependent extrapolation time will not be included in the calculation of the effective extrapolation time.
		Reserved4	BYTE	Reserved
		Pos_FilterT1	REAL	Position filter time constant T1
		Pos_FilterT2	REAL	Position filter time constant T2
		Vel_FilterT1	REAL	Velocity filter time constant T1
		Vel_FilterT2	REAL	Velocity filter time constant T2
		Vel_Tolerance	REAL	Tolerance band width for velocity
		Hysteresis	REAL	Hysteresis of the extrapolated actual position value
		Reserved5	REAL	Reserved
	Torque_Lmt		SYS_AXIS_TOR-QUELMT	
		PosLmt_En	BOOL	• 0: Disable Position-Based Monitoring • 1: Keep Postion-Base Monitoring Enabled
		Reserved6	BYTE	Reserved
		Torque_Force	REAL	TorqueLimit Percentage
		Reserved7	BYTE	Reserved
	Clamping		SYS_AX-IS_CLAMPING	
		Pos_Tolerance	REAL	Positioning tolerance
		Followin-gErr_Dev	REAL	Following error

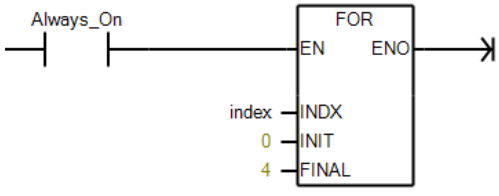
Variables			Data type	Description
	Ctrl_Mode		SYS_AX-IS_CTRL_MODE	
		Pos_Ctrl_Mode	BOOL	
		Speed_Ctrl_Mode	BOOL	
		Tor-que_Ctrl_Mode	BOOL	

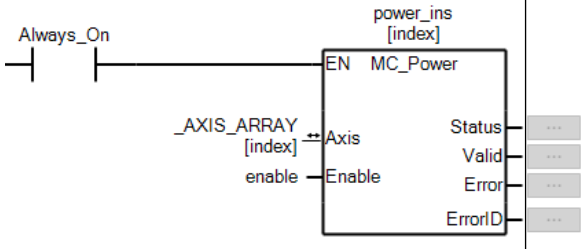
14.6.6 Achieve multi-axis control with arrays

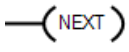
You can achieve multi-axis control by calling array.

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 14-1 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: 		

14.7 Motion Control Instructions

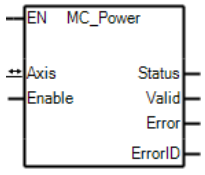
14.7.1 Basic Instruction

14.7.1.1 MC_Power

The MC_Power instruction is primarily used to enable and disable axes. It is a prerequisite for all axis motion operations; only after MC_Power successfully enables an axis can other motion control instructions (such as MC_MoveAbsolute, MC_MoveRelative, etc.) function properly.

The subsequent motion instructions should be triggered based on the status of the MC_Power.

Parameters

LAD/FBD	STL	Description
	**FBCALL MC_Power, INSTANCE, Axis, Enable, Status, Valid, Error, ErrorID**	Switch the axis to operational status

Parameters	Declaration	Data type	Initial value	Description
EN	IN	BOOL	-	Power flow
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Enable	IN	BOOL	-	• TRUE: Enable • FALSE: Not enabled
Status	OUT	BOOL	False	Becomes TRUE upon entry into the operational state
Valid	OUT	BOOL	False	TRUE: The MC_Power FB instance has been successfully bound to the axis
Error	OUT	BOOL	False	An error has occurred with the instruction
ErrorID	OUT	WORD	0	Error code (Page 1015)

Note

Disabling the axis while the axis is in the error stop state will keep the error stop state.

Override response

- The execution of "MC_Power" (including "enable" and "disable") cannot be aborted by a Motion Control instruction
- Disabling the axis (input parameter "Enable" = FALSE) aborts all Motion Control instructions for the associated axis immediately

Preconditions

Note

Single instance control

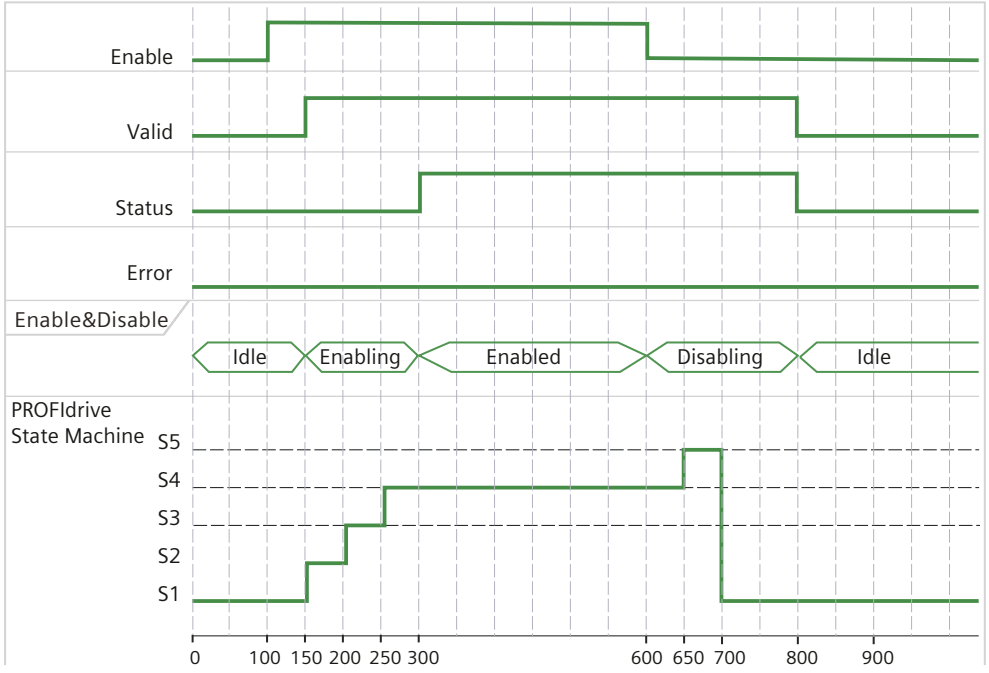
Each axis can only be enabled by one MC_Power FB instance. If an axis has been successfully enabled by an MC_Power FB instance, it cannot be enabled by another MC_Power FB instance.

The PN IRT axis must meet the following conditions:

- Communication has been established between the controller and the drive.
- The axis must be correctly configured and in the "Drive Ready" state.

Enable axis

When the Enable parameter is switched from FALSE to TRUE, MC_Power first performs a series of internal checks, such as verifying drive status and communication status, to ensure that the operating environment is ready. If all checks pass, MC_Power will send an enable signal to the driver. At this point, the Valid output will become TRUE, and after the axis is enabled, the Status output will become TRUE.



S1 Powering on inhibited: (drive off, brake closed if necessary)

S2 Ready to power on: (Drive is ready to receive power)

S3 Powered on: (Drive powered on, release brake if necessary)

S4 In operation: (Drive released, brakes released if necessary)

S5 Shutting down: (Drive brakes according to a predefined ramp)

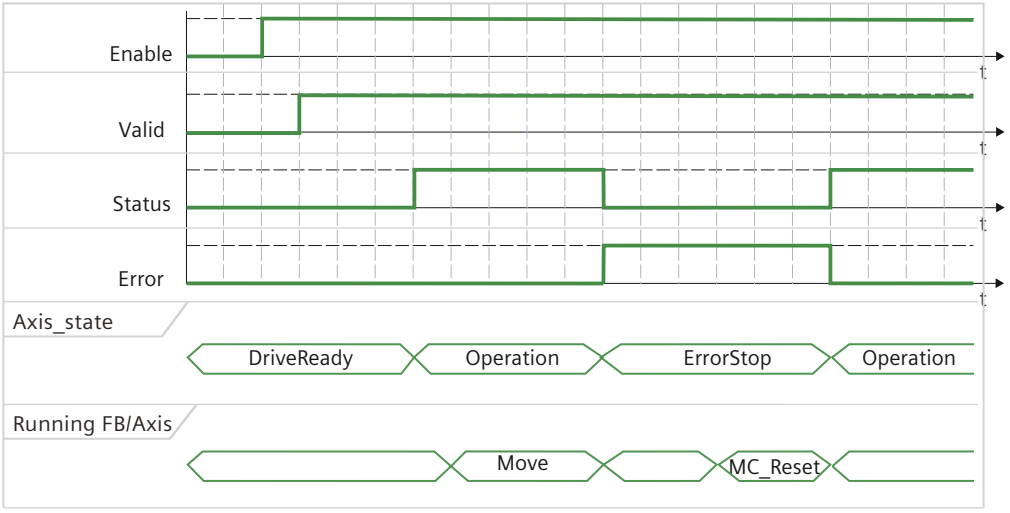
Disable axis

If no errors occur on the axis, the Status output of MC_Power will remain TRUE, indicating that the axis is in a normal enabled state.

Once an error is detected with the axis object, the MC_Power FB instance immediately updates its output:

- The Status output will become FALSE.
- The Error output will become TRUE.

To clear the error state in MC_Power and allow the axis to re-enter the enabled state, you need to use MC_Reset (Page 913).



14.7.1.2 MC_TouchProbe

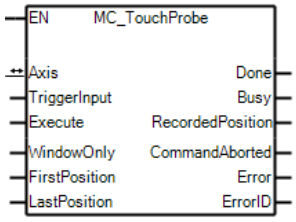
The MC_TouchProbe instruction is used to capture and record the current position of the axis when the probe is triggered.

Note

For the PN IRT axis, TouchProbe only supports drive mode, where the probe position detection is performed by the drive, and the probe must be configured at the drive end. PTO and encoder only support controller mode, directly reading the bus cycle feedback position received by the PLC.

When configuring the Siemens servo drive S200, refer to the operating instructions for the S200 for detailed configuration guidance on the probe function.

Parameter

LAD/FBD	STL	Description
	**FBCALL MC_TouchProbe, INSTANCE, Axis, TriggerInput, Execute, WindowOnly, FirstPosition,LastPosition,Done, Busy, RecordedPosition, CommandAborted, Error, ErrorID**	This instruction configures the touch probe.

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	-	The instruction is triggered on the 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 running[1]
RecordedPosi-tion	OUT	REAL	0.0	Capture the position where trigger event occurred
CommandA-borted	OUT	BOOL	False	The current instruction is aborted by another instruction
Error	OUT	BOOL	False	An error has occurred with the instruction.
ErrorID	OUT	WORD	0	Error code (Page 1015)

[1] When the MC_TouchProbe instruction is in the Busy state, it will continue to detect even if the axis is out of the [FirstPosition, LastPosition] range and the probe has not been triggered, without reporting an error.

Note

Using MC_TouchProbe Command

- If two MC_TouchProbe instructions have the same axis number and probe ID, the latter will report an error and exit, while the execution of the former remains unaffected.
- PTOs and encoders require configuring probes. Every single axis has a range of [I0.0~I1.7]. 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.

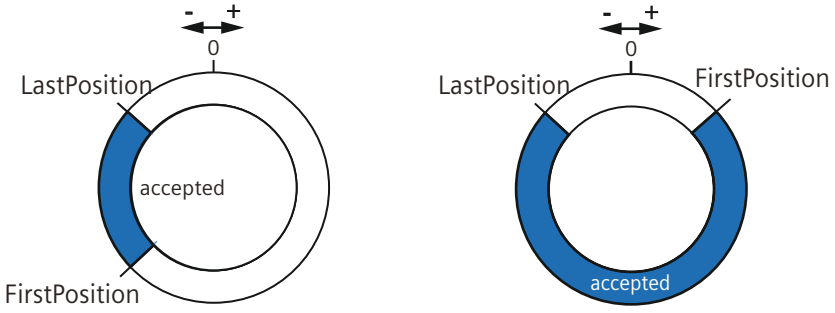
Note

When the Execute pin of the MC_TouchProbe instruction is set high, if the axis ID is changed during this period, re-triggering the instruction will perform the probe function for the new axis ID.

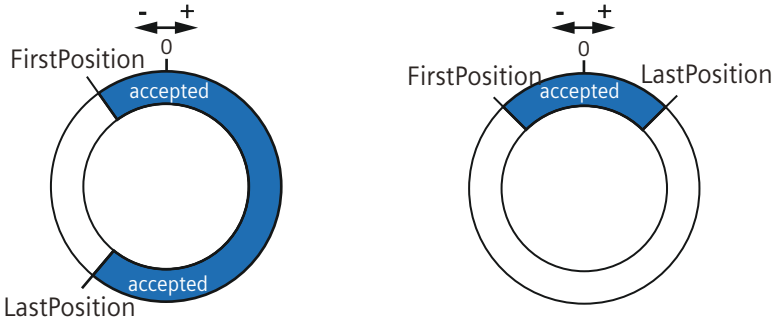
Linear axis and modulo axis

- Linear axis: The TouchProbe function will only detect the probe signal and output a valid result if the current position of the axis is within the range defined by FirstPosition and LastPosition.
- Modulo axis: FirstPosition and LastPosition will perform a modulo operation on the cycle to determine the effective detection range within a cycle.

  A. FirstPosition < LastPosition



  B. FirstPosition > LastPosition



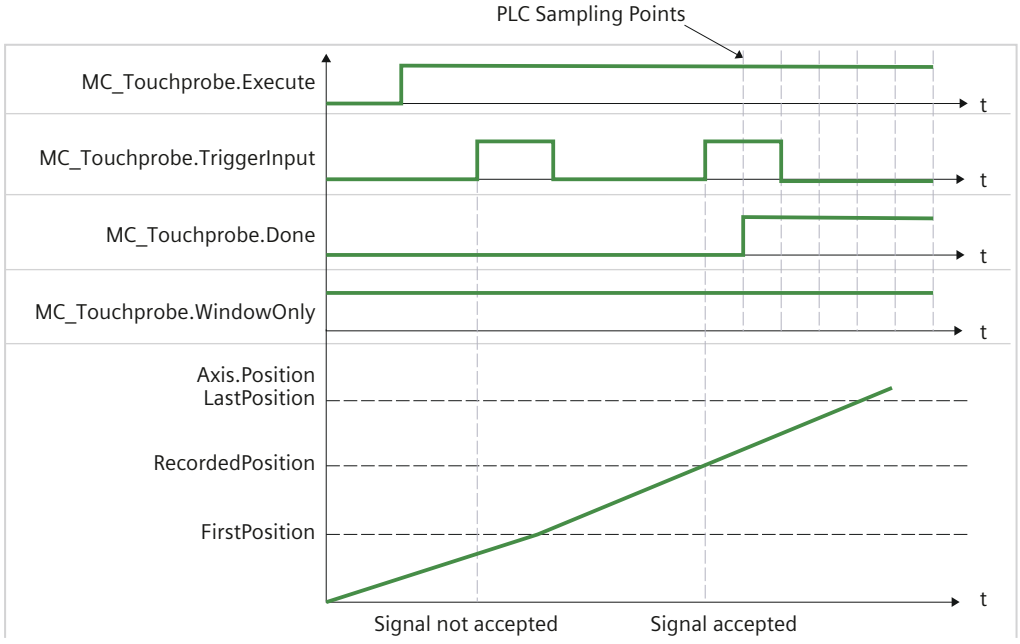
Preconditions

- The axis has been configured correctly.
- The axis is enabled and homed.

- The drive's probe function is successfully configured for the PN IRT axis.
- TriggerInput must be configured correctly. First add a variable for TriggerInput in the variable table, then set its data type as MC_TRIGGER_REF.

MC_TRIGGER_REF structure element variable	Description
ProbeID	• Associate ProbeID = 0: associated with axis configuration probe0 • Associate ProbeID = 1: associated with axis configuration probe1
TriggerEdge	• TriggerEdge = 0: rising edge of the probe is valid • TriggerEdge = 1: falling edge of the probe is valid

Execution



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.

14.7.1.3 MC_AbortTrigger

The MC_AbortTrigger instruction cancels or aborts an active MC_TouchProbe operation.

- IRT axis: Since the AbortTrigger instruction defined by the underlying protocol applies to all TouchProbe instances on a specified axis, rather than a single instance, MC_AbortTrigger will terminate all currently active TouchProbe operations on that axis.
- PTO axis: The PTO's AbortTrigger only interrupts probes with a specific probeID on the specified axis.

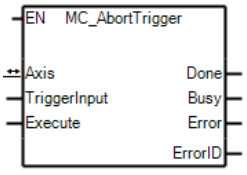
Note

If both MC_TouchProbe and MC_AbortTrigger are enabled before the next IPO, the following execution process applies: If MC_AbortTrigger executes first, an error will be reported; if MC_TouchProbe executes first, MC_AbortTrigger will cancel the MC_TouchProbe operation.

Note

MC_TouchProbe does not support continuous triggering mode; it only supports single triggering. Once the probe is successfully detected, the Done bit of the MC_TouchProbe function block will be set to TRUE.

Parameter

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).

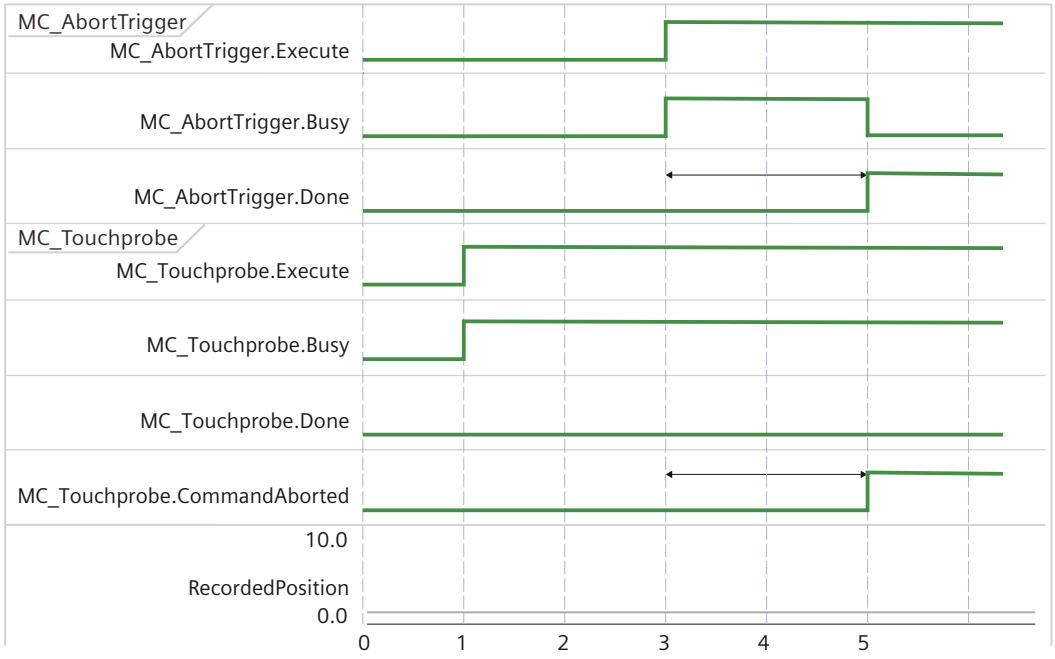
Parameter	Declaration	Data type	Initial value	Description
EN	IN	BOOL	-	Power flow
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
TriggerInput	IN_OUT	MC_TRIG-GER_REF	-	Reference to the trigger signal source
Execute	IN	BOOL	-	The instruction is triggered on the rising edge.
Done	OUT	BOOL	False	Trigger functionality aborted
Busy	OUT	BOOL	False	The instruction is running
Error	OUT	BOOL	False	An error has occurred with the instruction
ErrorID	OUT	WORD	0	Error code (Page 1015)

Preconditions

- The axis has been configured correctly.
- The drive's probe function is configured.
- There is a MC_TouchProbe instruction running on the axis.

Execution

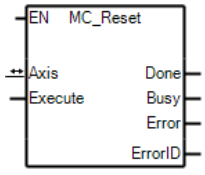
Even if the Execute input of MC_AbortTrigger is set to interrupt MC_TouchProbe, the CommandAborted output of MC_TouchProbe will not immediately go high.



14.7.1.4 MC_Reset

The main function of MC_Reset is to clear the error state of the axis.

Parameters

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 instances.

Parameters	Declaration	Data type	Initial value	Description
EN	IN	BOOL	-	Power flow
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Execute	IN	BOOL	-	The instruction is triggered on the rising edge.
Done	OUT	BOOL	False	The job has ended
Busy	OUT	BOOL	False	The instruction is running
Error	OUT	BOOL	False	An error has occurred with the instruction
ErrorID	OUT	WORD	0	Error code (Page 1015)

Preconditions

- The axis has been configured correctly.

Restart axis

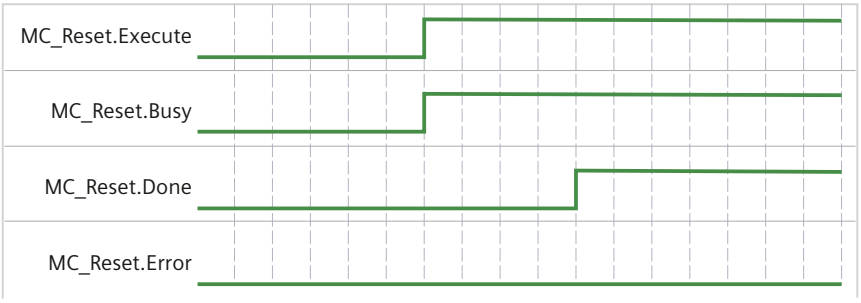
To restart a certain axis, follow these steps:

1. Check the requirements indicated above.
2. Perform the restart with a rising edge at parameter "Execute".

If the "Done" parameter shows the value "TRUE", it means that the restart job has been completed.

Note

If the error is not eliminated, it will re-enter the error stop state even after the restart is completed. (Such as network connectivity issues, servo/motor connection issues)



Override response

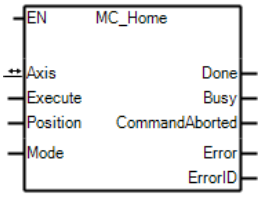
- The MC_Reset instruction cannot be aborted or buffered by any other Motion Control instruction.
- The new MC_Reset instruction does not abort or buffer to any other active Motion Control instructions.

14.7.1.5 MC_Home

MC_Home unifies the actual mechanical position of the axis with the axis position in the S7-200 SMART program.

Parameters

Table 14-2 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.

Parameters	Declaration	Data type	Initial value	Description
EN	IN	BOOL	-	Power flow
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Execute	IN	BOOL	-	The instruction is triggered on the rising edge.
Position	IN	REAL	0.0	Specifies the position after homing After the instruction ends, the axis position must be determined in conjunction with the Mode.
Mode	IN	BYTE	-	Homing mode: • 0: Direct home (Absolute) • 1: Direct home (Relative) • 3: Active homing[1]
Done	OUT	BOOL	False	Home point is set or found successfully
Busy	OUT	BOOL	False	The instruction is running
CommandAborted	OUT	BOOL	False	The current instruction is aborted by another instruction
Error	OUT	BOOL	False	An error has occurred with the instruction
ErrorID	OUT	WORD	0	Error code (Page 1015)

[1] Mode 3 only supports PTO and IRT axes. Before executing active homing, the parameters must be correctly configured in the axis "Active homing" page, and the relevant PTO and IRT axes must be in a stationary state (Axis_Status.StandStill = TRUE).

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.

homing mode

Mode	Description	Axis
Mode 0	Direct home (Absolute) The current position of the axis is set to the value of the parameter "Position".	Applicable to all axes
Mode 1	Direct home (Relative) The current position of the axis will be offset based on the value of the parameter "Position".	Applicable to all axes
Mode 3	Active Homing After completion of the motion, the axis is positioned at the value of the "Position" parameter.	Applicable to PTO axes and IRT axes

Maintain homed state

The table below illustrates the maintenance and changes of various axis homed states in different scenarios.

Behavior	PTO axis		Virtual axis		Encoder Axis	
	Homed state	Axis position	Homed state	Axis position	Homed state	Axis position
MC_Power OFF	Missing	Hold	Hold	Hold		
CPU STOP -> RUN	Missing	Reset	Hold	Hold	Hold	Varies with actual position
MC_Home re-executed but not yet completed	Missing	Varies with actual position	Missing	Hold	Missing	Varies with actual position
PLC power off	Missing	Reset	Missing	Reset	Missing	Reset
Program download	Missing	Reset	Missing	Reset	Missing	Reset

Behavior	PN IRT axis (Incremental encoder and single-turn absolute encoder)		PN IRT axis (Multi-turn absolute encoder)	
	Homed state	Axis position	Homed state	Axis position
MC_Power OFF	Hold	Hold	Hold	Hold
CPU STOP -> RUN	Hold	Hold	Hold	Hold
MC_Home re-executed but not yet completed	Missing	Varies with actual position	Missing	Varies with actual position
PLC power off/servo disconnection	Missing	Incremental encoder: Undetermined value. Single-turn absolute encoder: Varies with actual position. However, when its internal count overflows, the reported position data becomes unreliable.	Hold	Hold
Program download	Missing	Hold	Hold	Hold

Preconditions

Absolute direct homing (Mode 0) and relative direct homing (Mode 1)

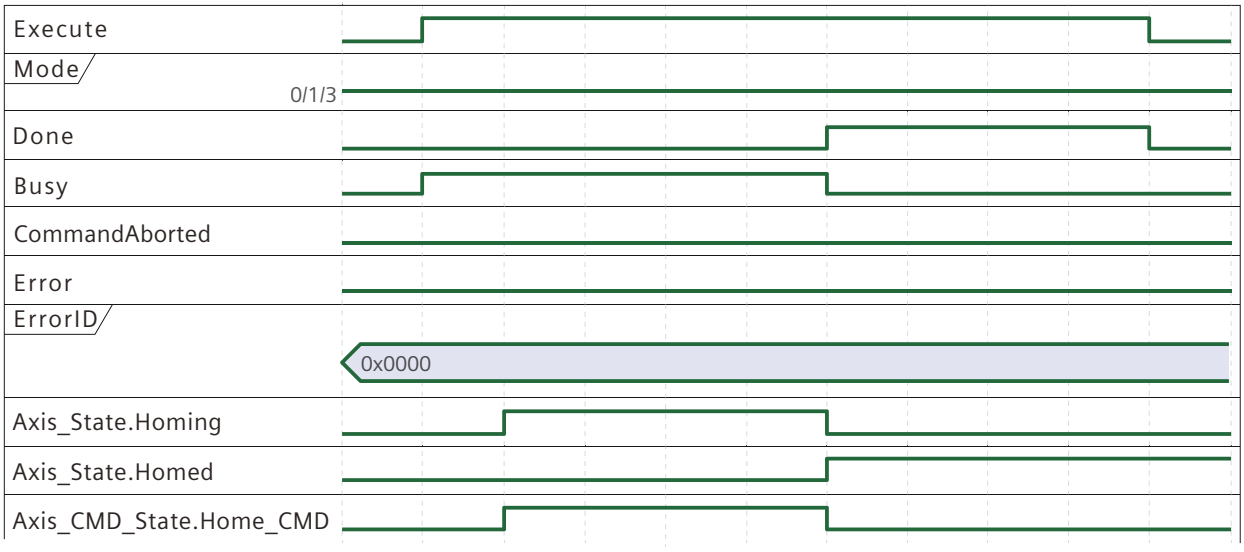
- The axis has been configured correctly
- The axis is enabled.
- The axis has not been enabled by the axis group
- Axis condition requirements
  - Encoder Axis: It must be enabled (Axis_Status.ENC_Enable = TRUE)
  - Other axes: It must be disabled or in a standstill (Axis_Status.Disabled =TRUE or Axis_Status.StandStill = TRUE)

Active homing (Mode 3)

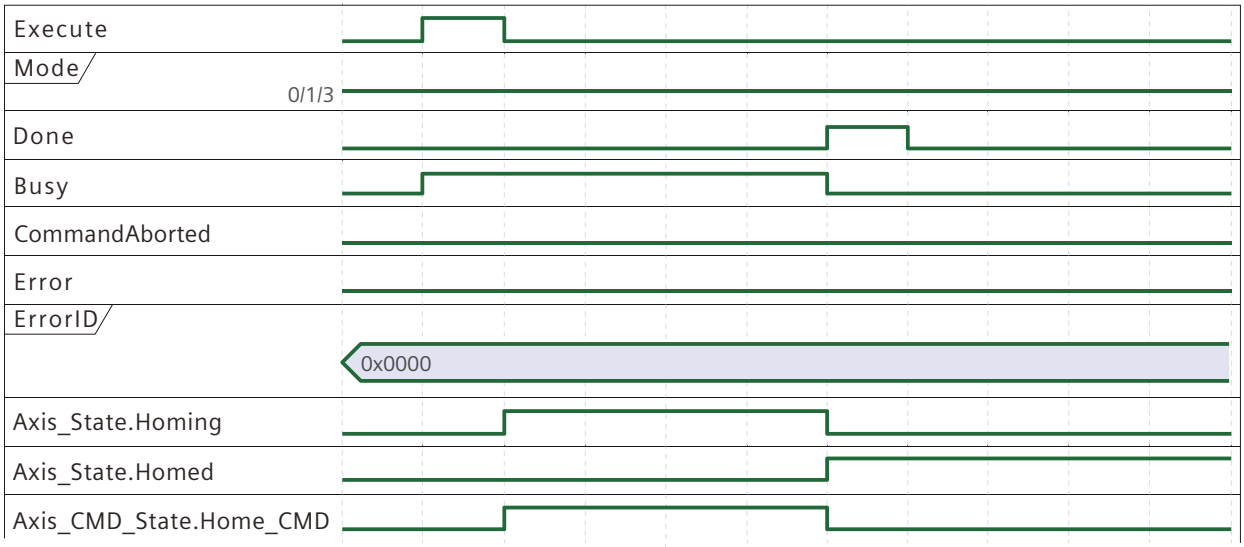
- Supports PTO and IRT axes
- The axis has not been enabled by the axis group
- The axis must be in a standstill (Axis_Status=StandStill).

Execute MC_Home when home is not determined

- The trigger signal remains high during the homing process

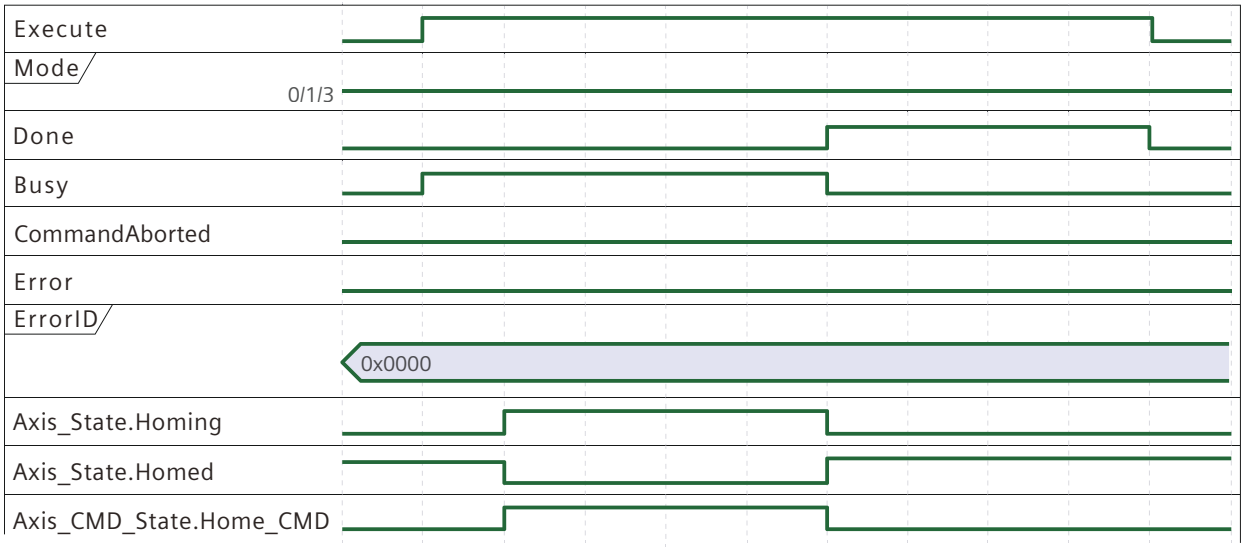


- The trigger signal goes low before the homing process ends

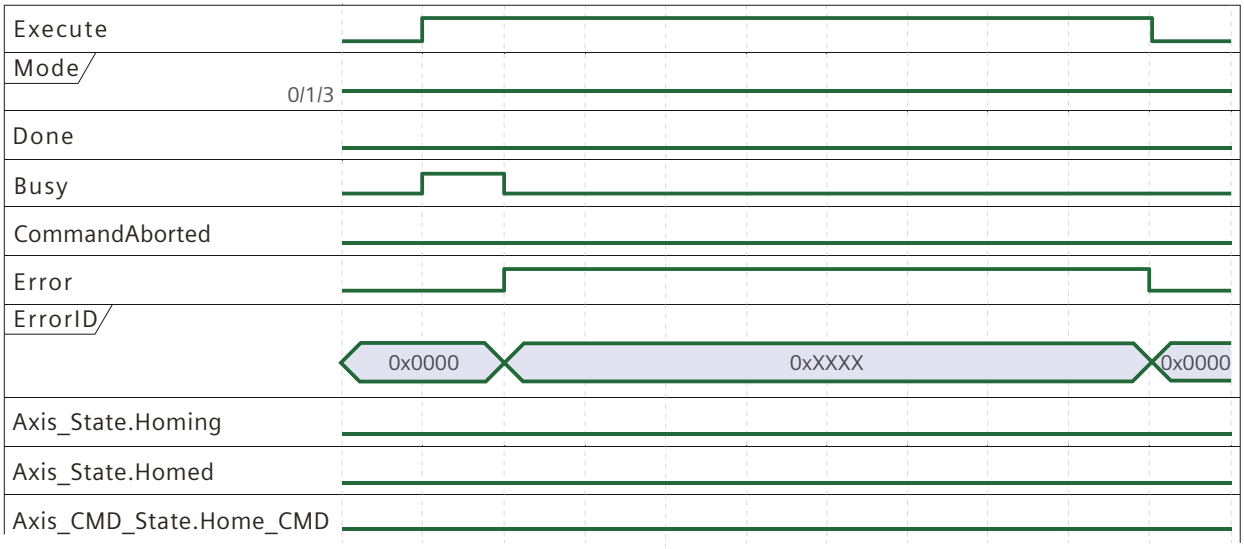


Execute MC_Home when home is determined

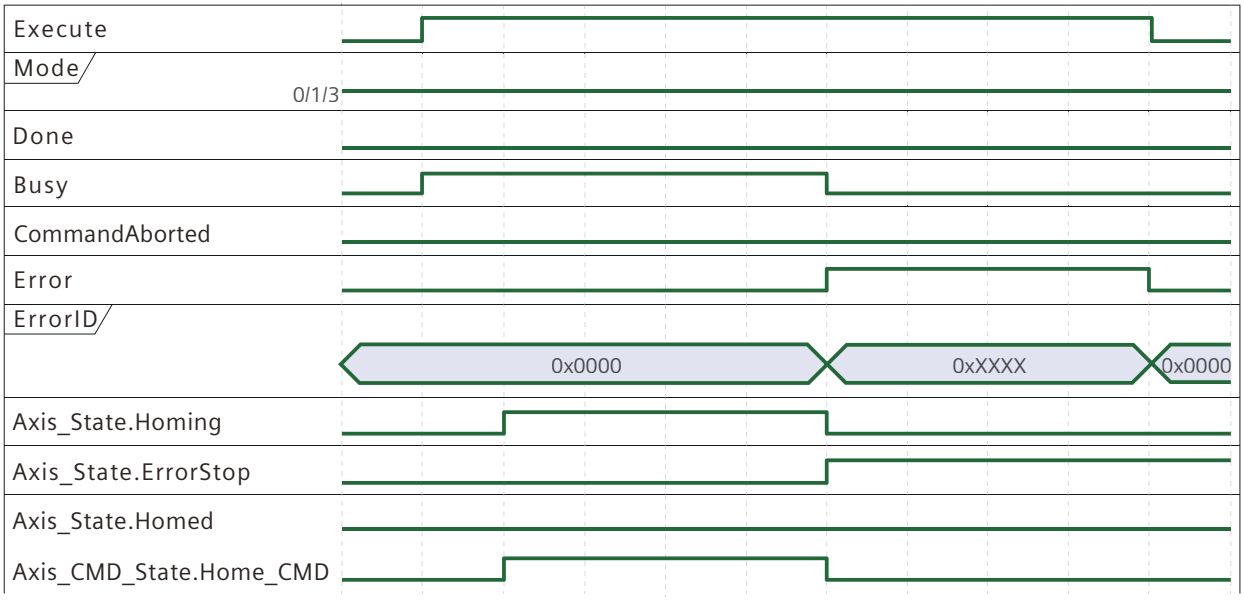
If MC_Home is executed repeatedly, that is, executed in the Axis_State.Homed state, Axis_State.Homed will be pulled low first.



Execution conditions for MC_Home are not met



Error executing MC_Home



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 MC_Home instruction cannot be buffered to any Motion Control instruction.

See also

Homing (Page 829)

14.7.1.6 MC_Stop

This instruction applies to IRT axes, PTO axes, and virtual axes. It is used to instruction a controlled motion stop and transition the axis to the Stopping state. This instruction interrupts the execution of any active motion FBs. Any motion function blocks, with the exception of MC_ImmediateStop, are unable to perform motion operations on the axis while it is in the Stopping state. Once the axis speed reaches zero (for IRT axes, the speed is considered zero when the user-defined standstill criteria are met), the Done output is set to TRUE immediately. The axis remains in the Stopping state as long as Execute is TRUE or the speed has not yet reached zero. The axis transitions to the Standstill state when Done is TRUE and Execute is FALSE.

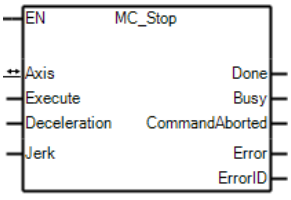
Note

As long as the Execute signal remains TRUE, the axis remains in the Stopping state.

Note

MC_STOP is not effective for torque control mode (MC_TorqueControl Mode 1).

Parameter

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 the execution of any ongoing instructions (with the exception of MC_ImmdiateStop). 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**.

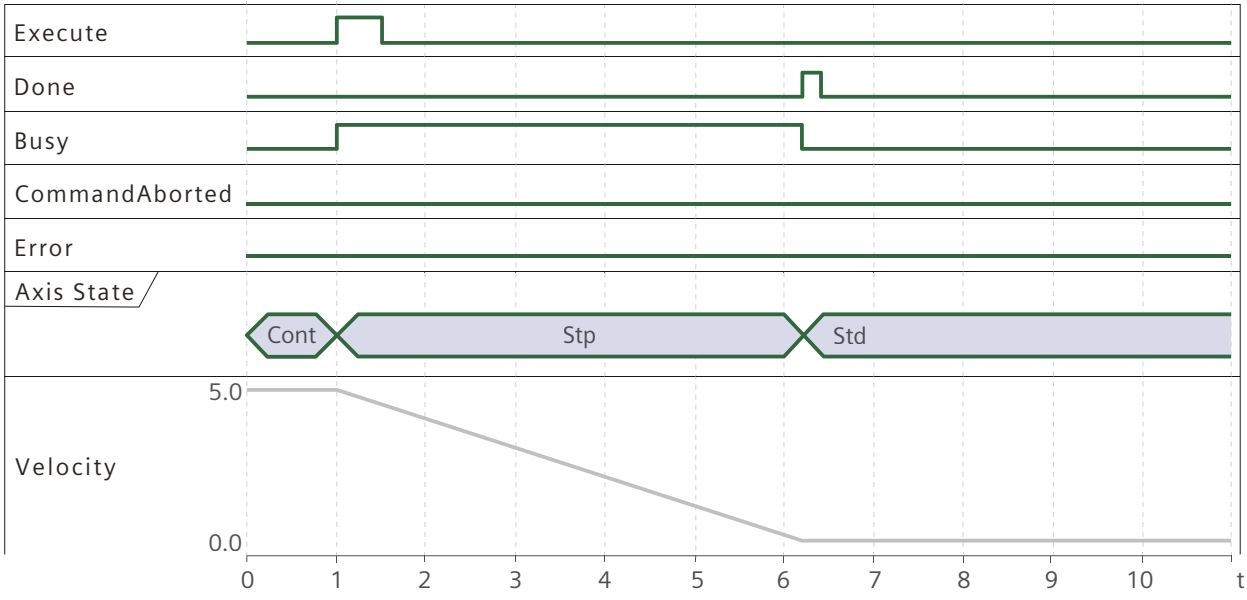
Parameter	Declaration	Data type	Initial value	Description
EN	IN	BOOL	-	Power flow
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Execute	IN	BOOL	-	The instruction is triggered on the rising edge.

Parameter	Declaration	Data type	Initial value	Description
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	Value of deceleration
Jerk	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	Value of jerk
Done	OUT	BOOL	False	• Zero velocity reached • The user-defined Stopping criteria for the IRT axis have been met.
Busy	OUT	BOOL	False	The instruction is running
CommandAborted	OUT	BOOL	False	The current instruction is aborted by another instruction • This pin is set to TRUE if the Stopping state is interrupted by an MC_Power or MC_ImmediateStop instruction before it has been cleared.
Error	OUT	BOOL	False	Signals that an error has occurred within the instruction
ErrorID	OUT	WORD	0	Error code (Page 1015)

Precondition

- The axis has been configured correctly.
- The axis is enabled.

Execution



Override response

- Motion control instructions that can abort the MC_Stop instruction:

Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_Power MC_ImmediateStop	

- A new MC_Stop instruction can abort the following motion control instructions:

Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_Home MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_MoveJog MC_CamIn MC_GearIn MC_CamOut MC_GearOut MC_PhasingRelative	

14.7.1.7 MC_Halt

Table 14-3 MC_Halt

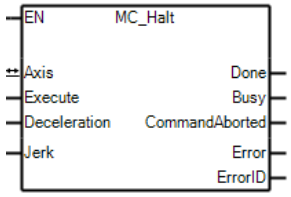
LAD/FBD	STL	Description
	**FBCALL MC_Halt, INSTANCE, Axis, Execute, Deceleration, Jerk, Done, Busy, CommandAborted, Error, ErrorID**	This instruction is used to command a controlled stop of axis motion. When this instruction is executed, the axis enters the "Discrete" state and smoothly decelerates from its current position and direction to zero speed, using the configured "Deceleration" and "Jerk" parameters.

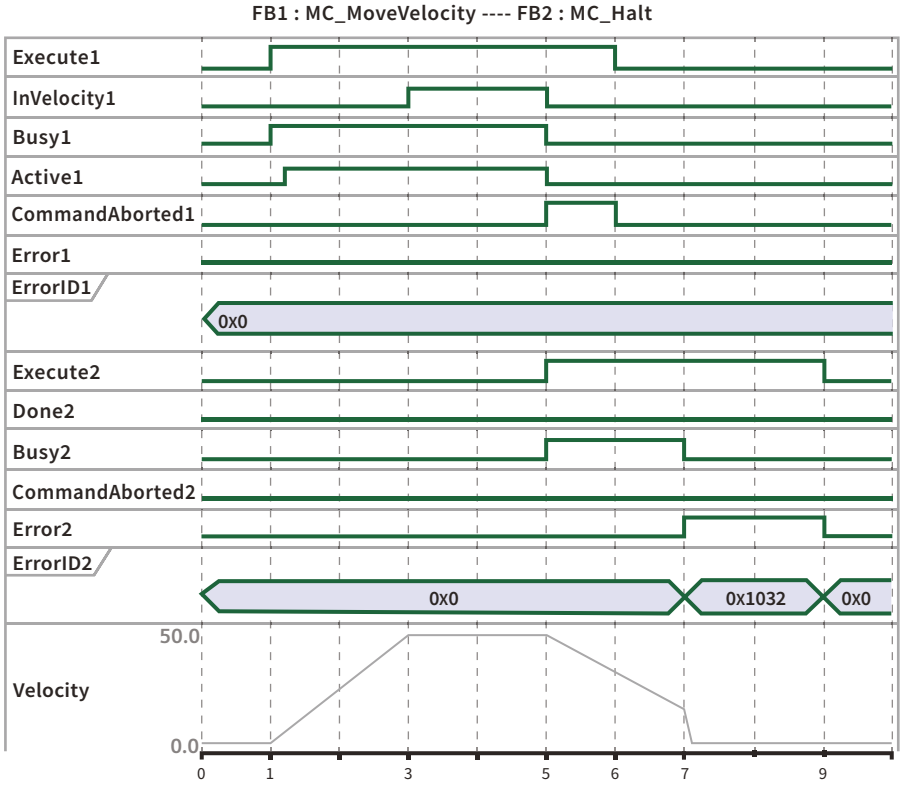
Table 14-4 Parameters for MC_Halt

Parameter	Declaration	Data type	Initial value	Description
EN	IN	BOOL	-	Power flow
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Execute	IN	BOOL	-	The instruction is triggered on the rising edge.
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	Deceleration value, optional • > 0.0: The specified value is used.

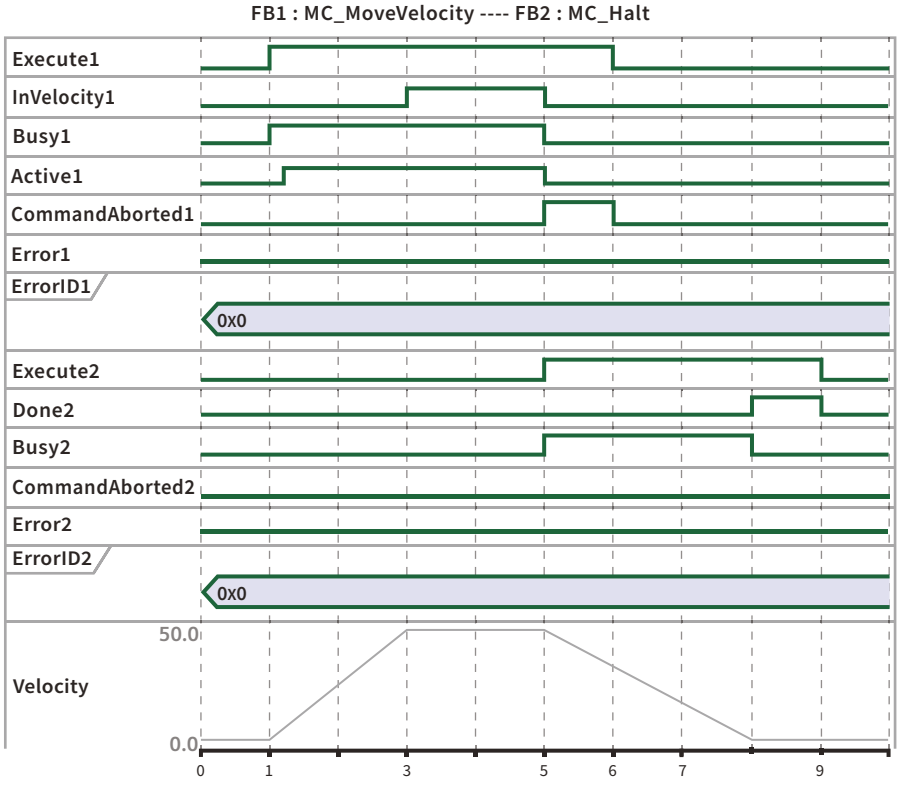
Parameter	Declaration	Data type	Initial value	Description
Jerk	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	Jerk value, optional • > 0.0: S-shaped velocity profile • =0: Trapezoid velocity profile
Done	OUT	BOOL	False	True： Zero velocity reached. For IRT axes, the setting of Done also indicates that the StandStill conditions have been met.
Busy	OUT	BOOL	False	The instruction is running
CommandAborted	OUT	BOOL	False	The current instruction is aborted by another instruction
Error	OUT	BOOL	False	An error has occurred with the instruction
ErrorID	OUT	WORD	0	Error code (Page 1015)

Timing diagrams

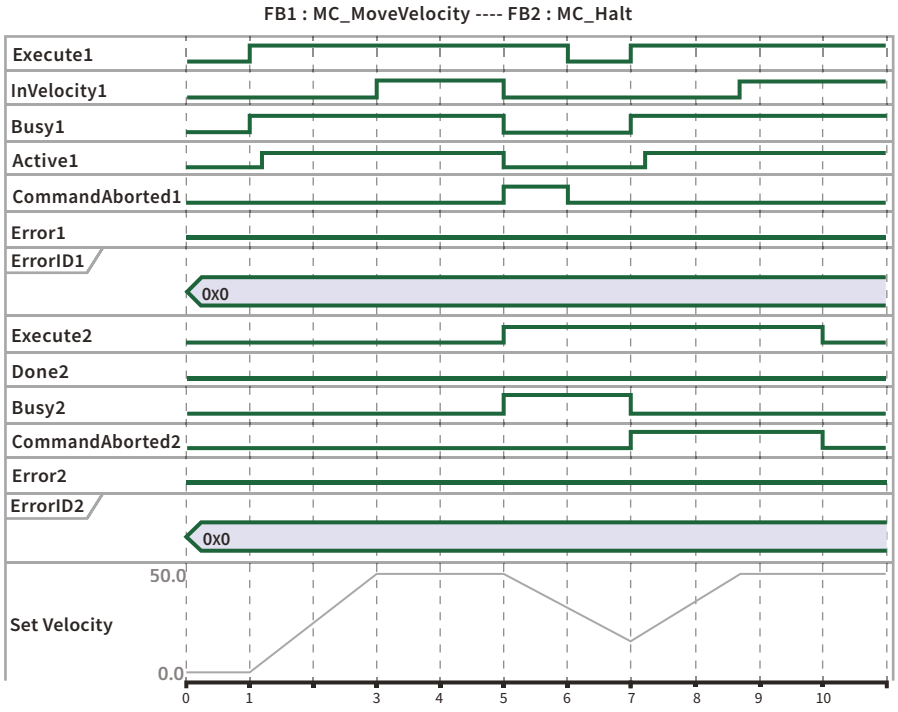
- During the deceleration-to-stop process of the MC_Halt instruction, if an error such as an axis error occurs, the system immediately interrupts the current deceleration and takes appropriate error-handling measures based on the error type. For example, when a hardware limit error is encountered, the controller prioritizes the limit protection response, stops further axis motion, and reports the corresponding error status.



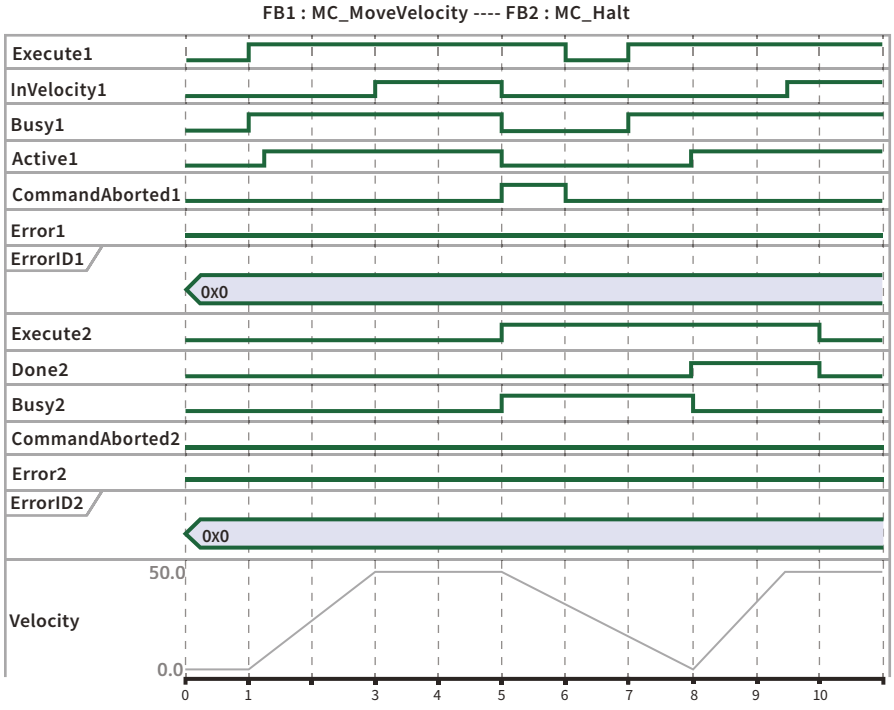
- When MC_MoveVelocity is triggered and a T-curve acceleration/deceleration profile is performed, the axis will enter the Continuous motion state once it has accelerated to the configured maximum speed (InVelocity). If MC_Halt is triggered at this point, the axis will begin decelerating according to the configured deceleration and Jerk parameters until it reaches zero speed, ultimately entering the Standstill state.


FB1 : MC_MoveVelocity ---- FB2 : MC_Halt
Execute1
InVelocity1
Busy1
Active1
CommandAborted1
Error1
ErrorID1
0x0
Execute2
Done2
Busy2
CommandAborted2
Error2
ErrorID2
0x0
Velocity
50.0
0.0
0
1
3
5
6
7
9

- During the deceleration phase of the MC_Halt instruction, if an MC_MoveVelocity instruction in Aborting mode is received, the current deceleration process of MC_Halt is interrupted. The axis immediately responds to the new MC_MoveVelocity instruction and executes acceleration or deceleration according to its T-curve parameters. At this point, the motion status and target velocity of the axis will be adjusted according to the newly issued MC_MoveVelocity instruction.



- During the deceleration-to-stop process of the MC_Halt instruction, if an MC_MoveVelocity instruction is issued in Buffered mode, it will be buffered in the instruction queue by the system. At this point, the axis continues to follow the MC_Halt deceleration curve until it comes to a complete stop. Once the deceleration process concludes and the Standstill state is entered, the system immediately retrieves the previously buffered MC_MoveVelocity instruction from the queue and executes the acceleration based on its T-curve parameters. This ensures a smooth transition between movements and avoids abrupt changes, satisfying the requirements for motion continuity and safety.



Override response

- Motion control instructions that can abort or buffer the MC_Halt instruction:

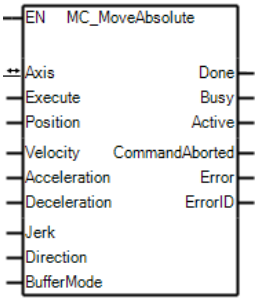
Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_Halt	MC_MoveAbsolute
MC_Stop	MC_MoveRelative
MC_ImmediateStop	MC_MoveVelocity
MC_MoveAbsolute	MC_CamIn
MC_MoveRelative	MC_GearIn
MC_MoveVelocity	
MC_MoveJog	
MC_CamIn	
MC_GearIn	

- A new MC_Halt instruction can abort the following motion control instructions:

Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_MoveJog MC_CamIn MC_GearIn MC_GearOut MC_CamOut MC_PhasingRelative	

14.7.1.8 MC_MoveAbsolute

The MC_MoveAbsolute instruction defines a target position using absolute coordinates and controls the specified axis to perform the intended trace motion.

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.

Parameter	Declaration	Data type	Initial value	Description
EN	IN	BOOL	-	Power flow
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Execute	IN	BOOL	-	The instruction is triggered on the rising edge.
Position	IN	REAL	-	Specify the target position in absolute coordinates
Velocity	IN	REAL	-	Target velocity The value range: Any real number
Acceleration[1]	IN	REAL	Axis_Config. Dynamic_Default. Default_Acc	Acceleration value, optional The value range: > 0.0
Deceleration[1]	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	Deceleration value, optional The value range: > 0.0
Jerk[1]	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	Jerk value, optional The value range: >= 0.0

Parameter	Declaration	Data type	Initial value	Description
Direction[1]	IN	BYTE	0	Direction of axis motion; this parameter is only effective when the axis is in rotation mode. • 0: Positive direction • 1: Negative direction • 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	Once the axis reaches the target position, the Done output is set to True
Busy	OUT	BOOL	False	The instruction is running
Active	OUT	BOOL	False	The instruction is currently controlling the axis
CommandAborted	OUT	BOOL	False	The instruction is aborted by another instruction.
Error	OUT	BOOL	False	An error has occurred with the instruction
ErrorID	OUT	WORD	0	Error code (Page 1015)

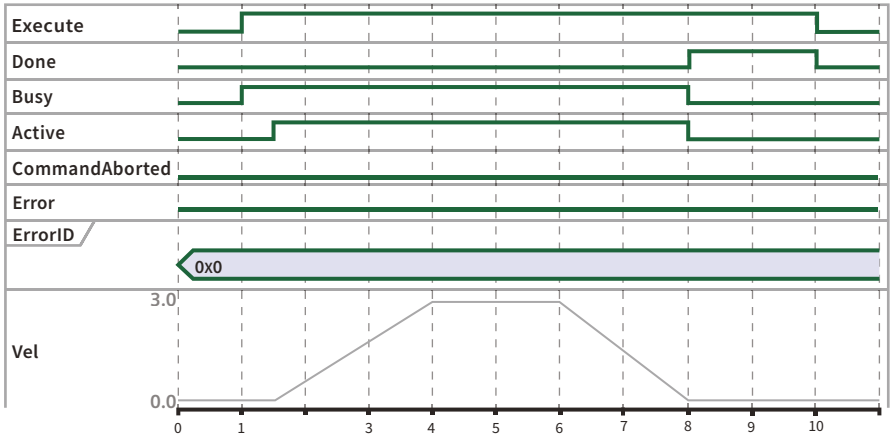
[1] When you don't assign values to these parameters, the system uses the default initial values.

Preconditions

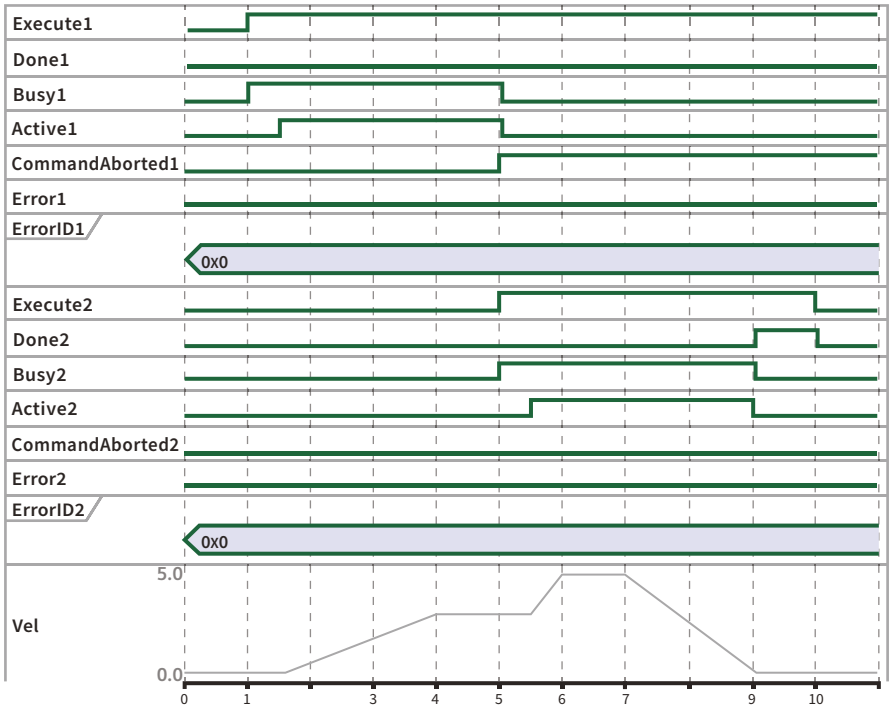
- Homed = 1

Timing diagrams

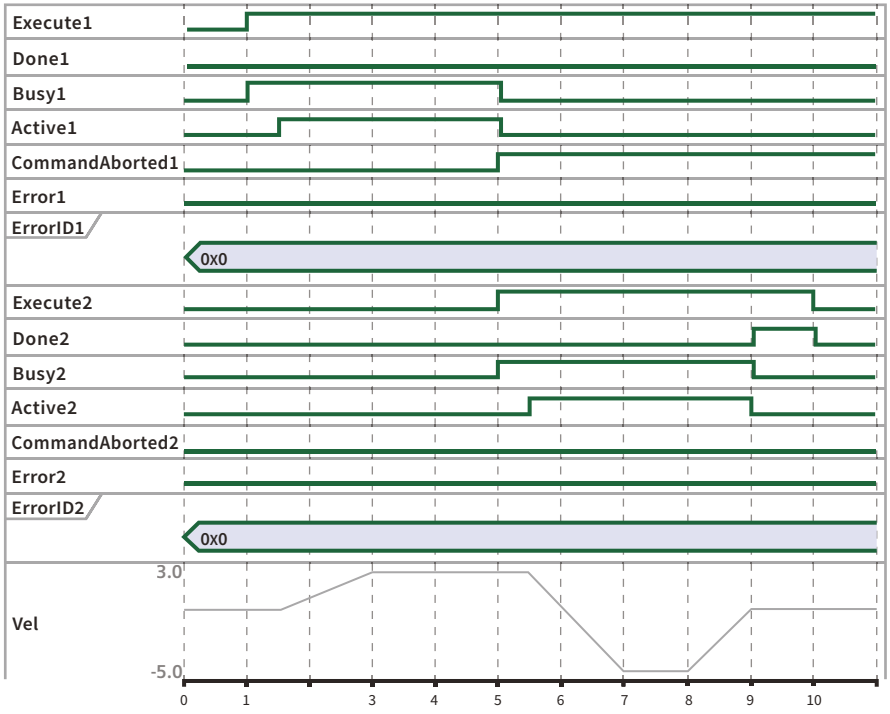
- When the axis state is in Standstill and Homed, the MC_MoveAbsolute instruction is triggered.



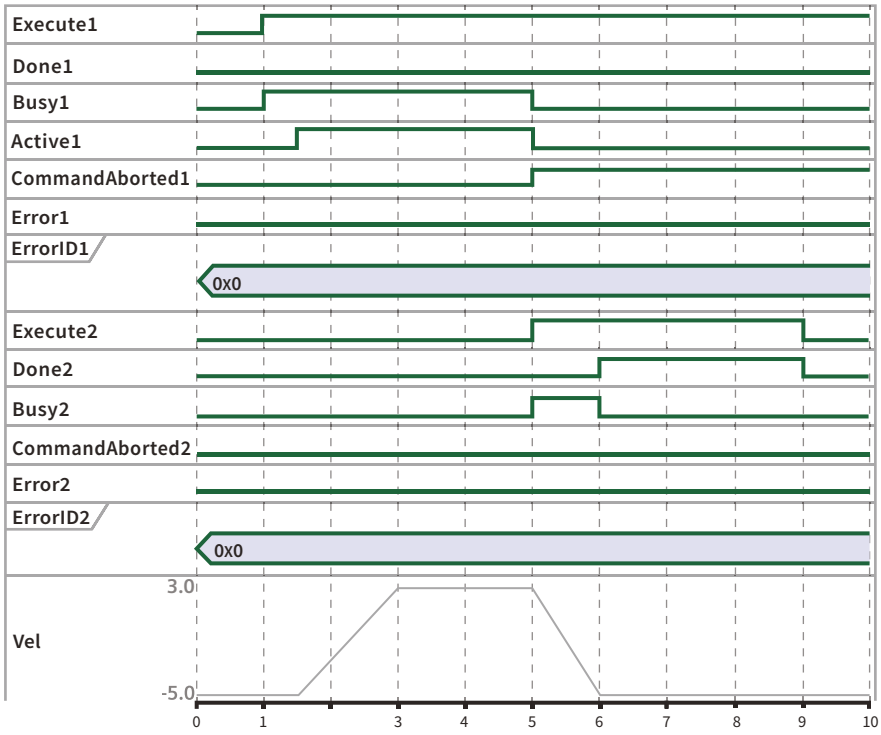
- When the trapezoidal curve is in the constant velocity phase, if a new MC_MoveAbsolute instruction is received with the target position aligned with the current direction of motion and BufferMode is set to 0 (No buffering, aborting), the current motion is immediately interrupted in forward direction. The axis regenerates and executes a new motion profile based on the new target position.



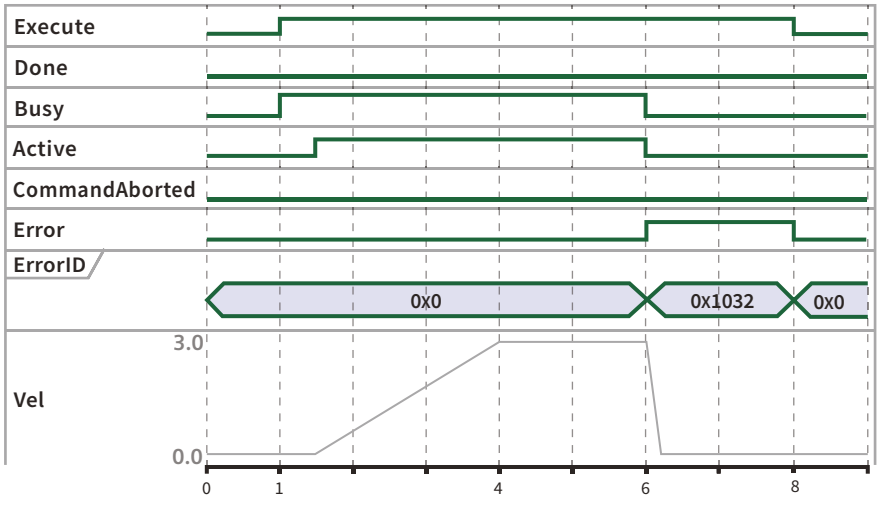
- When the trapezoidal curve is in the constant velocity phase, if a new MC_MoveAbsolute instruction is received with a target position in the opposite direction and BufferMode is set to 0 (No buffering, aborting), the current motion is immediately interrupted in the reverse direction. The axis regenerates a motion profile to reverse direction, decelerating until it comes to a stop based on the new target position.



- When the trapezoidal curve is in the constant velocity phase, if an MC_Stop instruction is received, the current motion is immediately interrupted, and the axis executes a stop movement according to the preset deceleration.



- When the trapezoidal curve is in the constant velocity phase, if an execution error occurs, the current motion is forcibly interrupted. The axis immediately enters the error-handling routine, reporting the relevant fault status and error information, and the motion is stopped to ensure system safety.



Override response

- Motion control instructions that can abort or buffer the MC_MoveAbsolute instruction:

Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_Halt MC_Stop MC_ImmediateStop MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_MoveJog MC_CamIn MC_GearIn	MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn

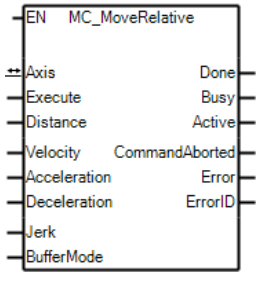
- A new MC_MoveAbsolute instruction can abort or buffer the following motion control instructions:

Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_MoveJog MC_CamOut MC_GearOut MC_CamIn MC_GearIn MC_PhasingRelative	MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn

14.7.1.9 MC_MoveRelative

MC_MoveRelative is used to move the specified axis a defined distance relative to its current position. It supports configuring the specified axis as a PTO axis, Virtual axis, or IRT axis, and is compatible with both Linear axes and Rotary axes.

Parameter

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.

Parameter	Declaration	Data type	Initial value	Description
EN	IN	BOOL	-	Power flow
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Execute	IN	BOOL	-	Start the instruction at rising edge
Distance	IN	REAL	-	Target relative distance
Velocity	IN	REAL	-	Target velocity The value range: Any real number
Acceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Acc	Acceleration value, optional The value range: > 0.0
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	Deceleration value, optional The value range: > 0.0
Jerk [1]	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	Jerk value, optional The value range: >= 0.0
BufferMode	IN	BYTE	0	Defines the chronological sequence of the instruction. • 0: Aborting mode: The previous motion function block instruction is immediately aborted, and the current instruction is executed starting from the position and velocity at the moment of interruption. • 1: Buffered mode: The current instruction continues without transition after the previous motion function block instruction is completed normally, with no dwell time between the two movements.
Done	OUT	BOOL	False	Once the axis reaches the target position, the Done output is set to True
Busy	OUT	BOOL	False	The instruction is running
Active	OUT	BOOL	False	The instruction is currently controlling the axis
CommandAborted	OUT	BOOL	False	The instruction is aborted by another instruction.
Error	OUT	BOOL	False	An error has occurred with the instruction
ErrorID	OUT	WORD	0	Error code (Page 1015)

[1] Motion instructions support T-curve, Triangular, and S-curve velocity profiles.

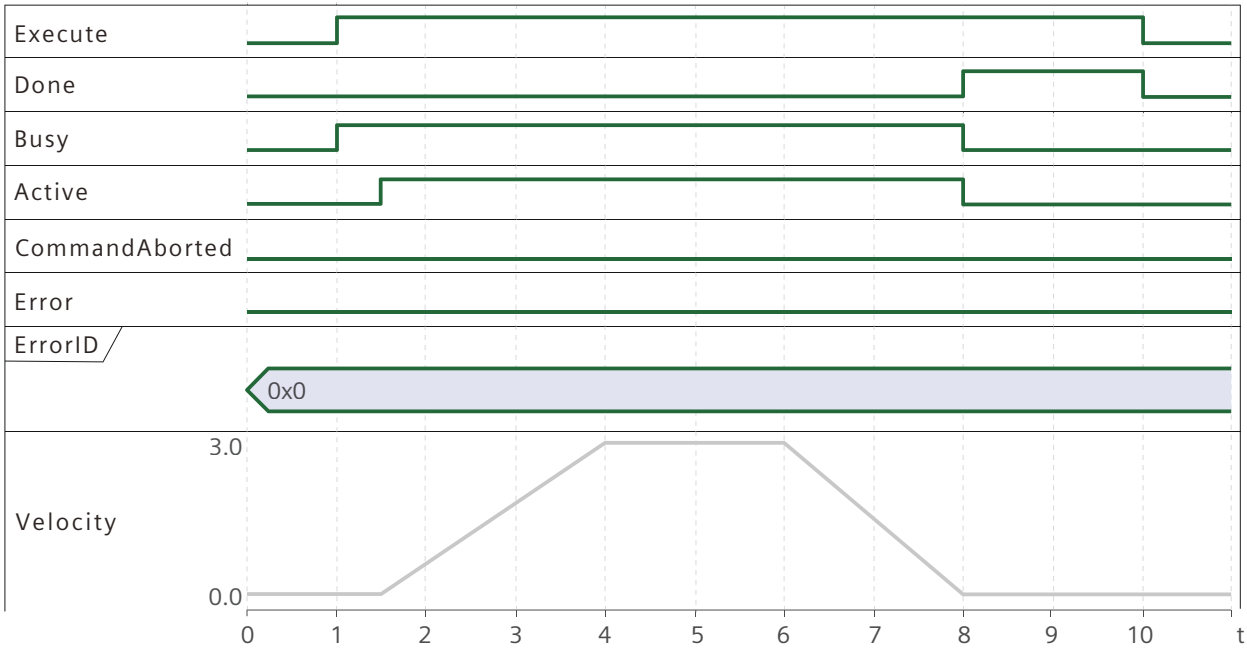
- When Jerk is set to 0:
  The system will execute a T-curve velocity profile. If the target velocity cannot be reached or is only just reached due to distance constraints, resulting in the absence of a constant velocity phase, the velocity profile will degenerate into a triangular curve.
- When Jerk is set to >0: The system will execute an S-curve velocity profile, which consists of up to 7 distinct segments. If the target velocity cannot be reached due to distance constraints, the S-curve will be replanned as a T-curve, because the system cannot find the optimal adjusted target velocity within a limited calculation time.

Preconditions

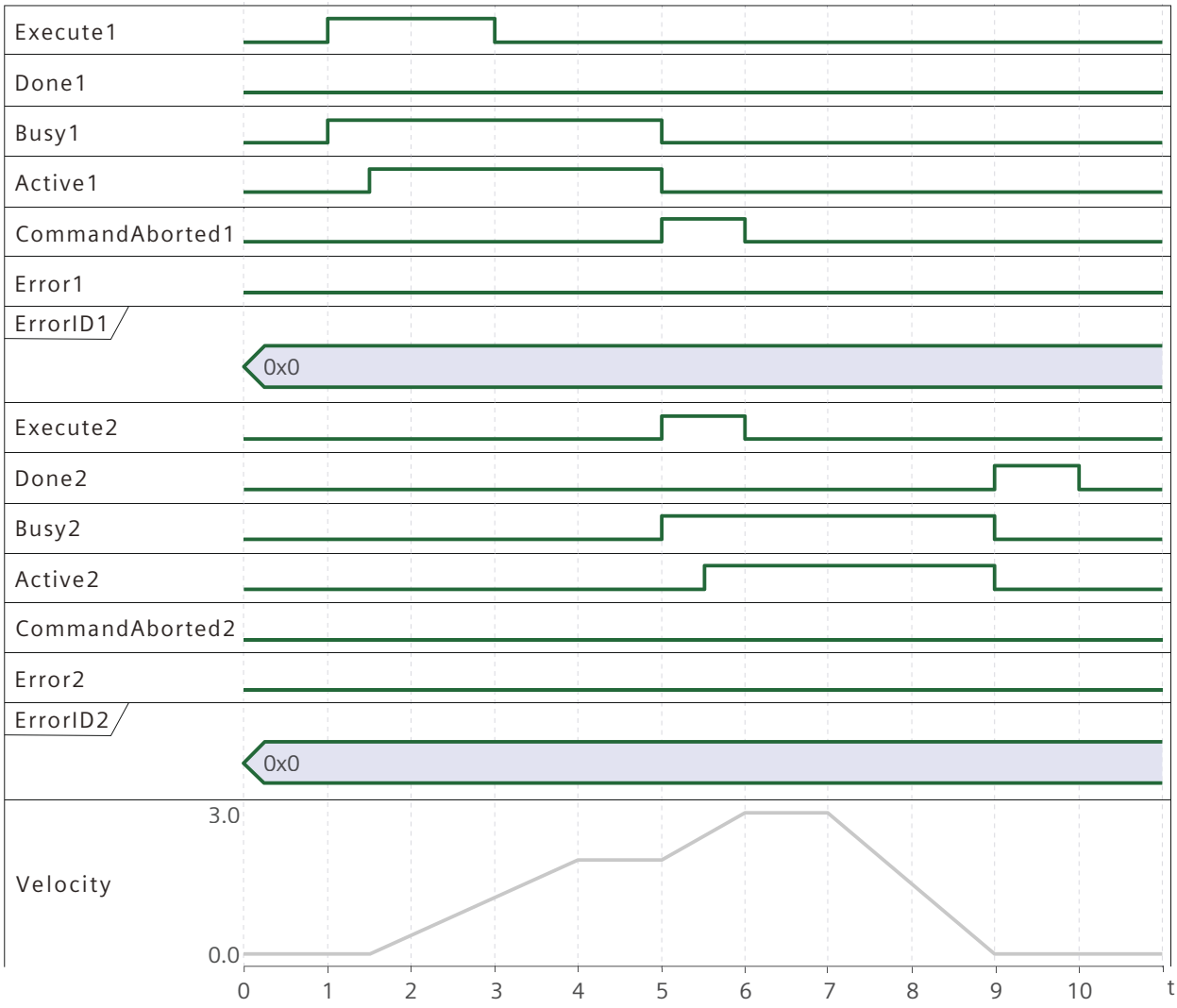
- The axis has been configured correctly.
- The axis is enabled.

Execution

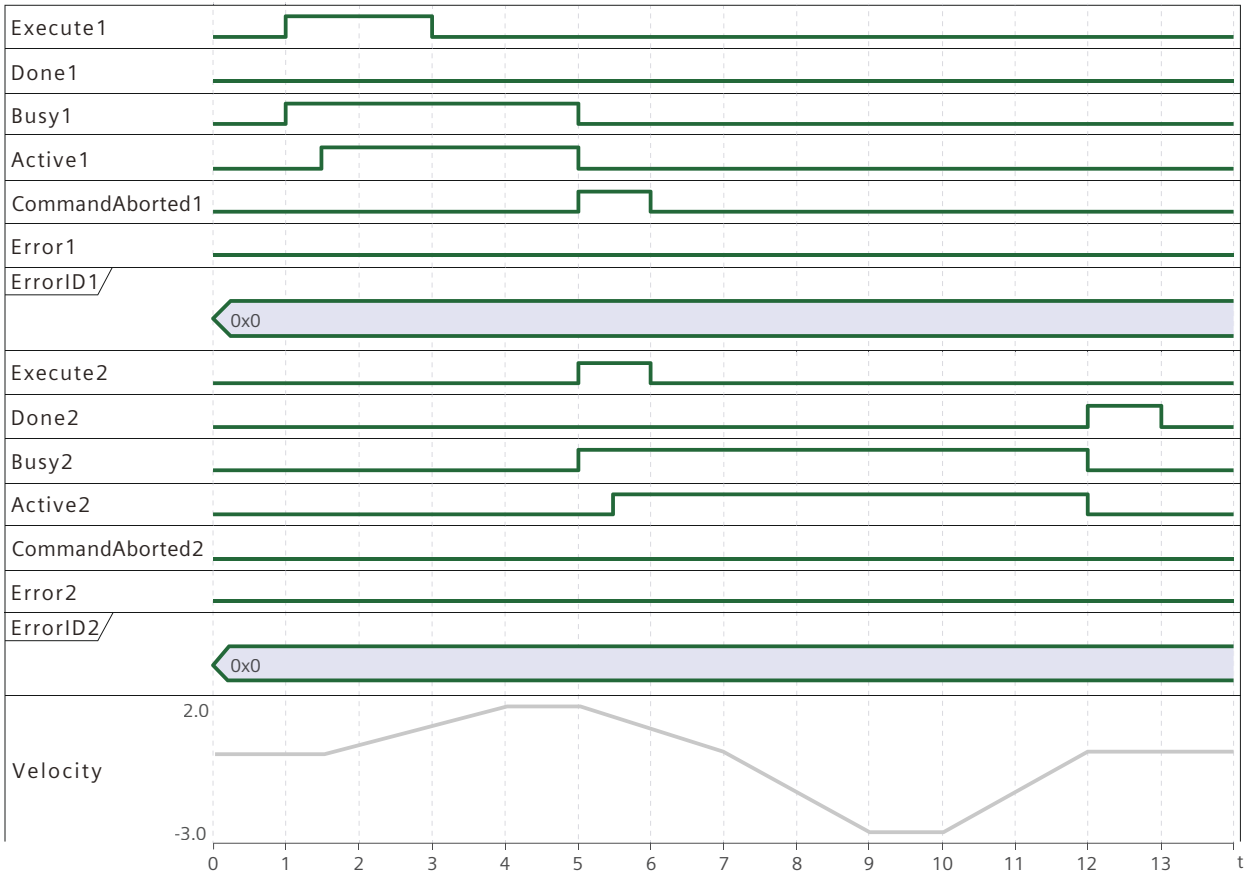
- When the axis is in the StandStill state, calling this instruction will execute a relative position movement using a T-curve velocity profile.



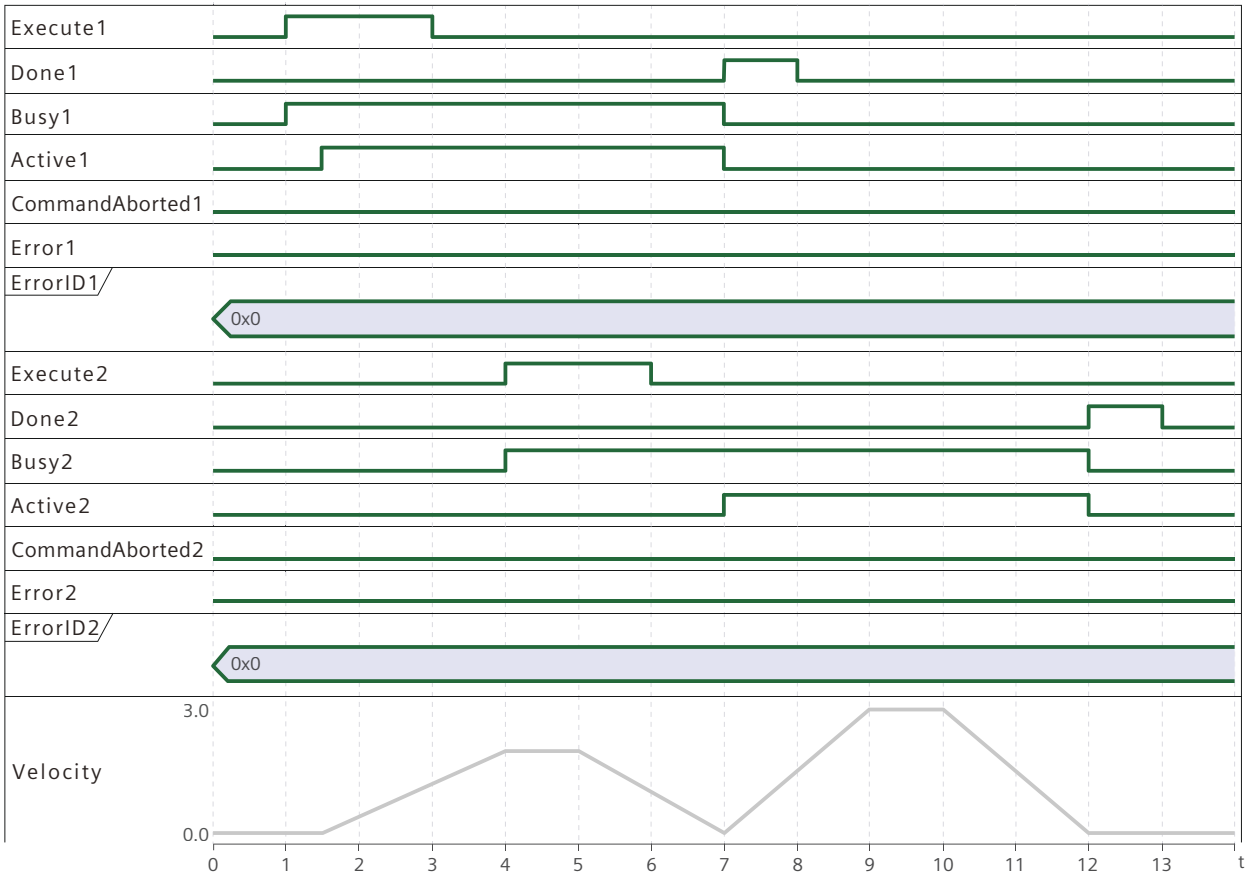
- While a T-curve relative motion is being executed, a new T-curve relative motion instruction in the same direction is triggered (BufferMode=0, forward aborting).


Execute1
Done1
Busy1
Active1
CommandAborted1
Error1
ErrorID1
0x0
Execute2
Done2
Busy2
Active2
CommandAborted2
Error2
ErrorID2
0x0
Velocity
3.0
0.0
0
1
2
3
4
5
6
7
8
9
10
t

- While a T-curve relative motion is being executed, a new T-curve relative motion instruction in the opposite direction is triggered (BufferMode=0, reverse aborting).



- While a T-curve relative motion is being executed, a new T-curve relative motion instruction is triggered (BufferMode=1).



Override response

- Motion control instructions that can abort or buffer the MC_MoveRelative instruction:

Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_Halt	MC_MoveAbsolute
MC_Stop	MC_MoveRelative
MC_ImmediateStop	MC_MoveVelocity
MC_MoveAbsolute	MC_CamIn
MC_MoveRelative	MC_GearIn
MC_MoveVelocity	
MC_MoveJog	
MC_CamIn	
MC_GearIn	

- A new MC_MoveRelative instruction can abort or buffer the following motion control instructions:

Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_Halt	MC_Halt
MC_MoveAbsolute	MC_MoveAbsolute
MC_MoveRelative	MC_MoveRelative
MC_MoveVelocity	MC_MoveVelocity
MC_MoveJog	MC_CamIn
MC_CamIn	MC_GearIn
MC_GearIn	
MC_CamOut	
MC_GearOut	
MC_PhasingRelative	

14.7.1.10 MC_MoveVelocity

The MC_MoveVelocity instruction is used to control a specified axis to perform continuous controlled motion at a set velocity. It supports PTO axes, virtual axes, and PN IRT axes, and is applicable to both linear and rotary axes.

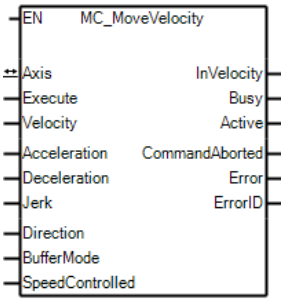
LAD/FBD	STL	Description
	**FBCALL MC_MoveVelocity, INSTANCE, Axis, Execute, Velocity, Acceleration, Deceleration, Jerk, Direction, BufferMode, SpeedControlled, Invelocity, Busy, Active, CommandAborted, Error, ErrorID**	This instruction commands the axis to perform an infinite controlled motion at the specified velocity.

Table 14-5 Parameters for MC_MoveVelocity

Parameter	Declaration	Data type	Initial value	Description
EN	IN	BOOL	-	Power flow
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Execute	IN	BOOL	-	Instruction triggered by rising edge
Velocity	IN	REAL	-	Target velocity The value range: Any real number
Acceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Acc	Acceleration value, optional The value range: > 0.0
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	Deceleration value, optional The value range: > 0.0
Jerk	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	Jerk value, optional The value range: >= 0.0

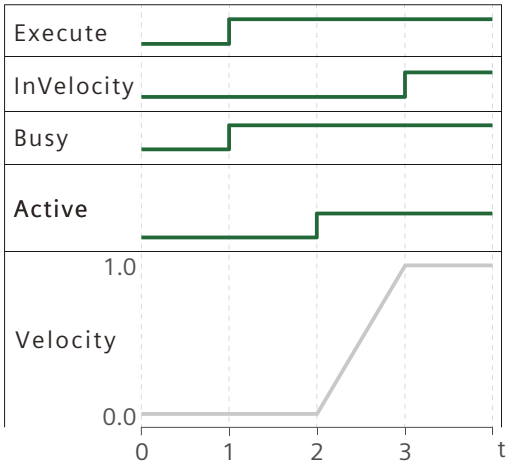
Parameter	Declaration	Data type	Initial value	Description
Direction	IN	BYTE	0	Moving direction 0 - Positive, 1 - Negative. This pin, together with the sign of the Velocity pin, determines whether the axis moves in a positive or negative direction. For example, if Direction = 1 and Velocity = 10.0, the axis moves in the negative direction. If Direction = 1 and Velocity = -10.0, the axis moves in the positive direction.
BufferMode	IN	BYTE	0	Buffered • 0: Aborting • 1: Buffered
SpeedControlled	IN	BOOL	0	Control mode • 0: Position-controlled mode • 1: Speed control mode
InVelocity	OUT	BOOL	False	True when the given velocity equals the target velocity
Busy	OUT	BOOL	False	The instruction is running
Active	OUT	BOOL	False	The instruction is currently controlling the axis
CommandAborted	OUT	BOOL	False	The instruction is aborted by another instruction.
Error	OUT	BOOL	False	An error has occurred with the instruction
ErrorID	OUT	WORD	0	Error code (Page 1015)

Preconditions

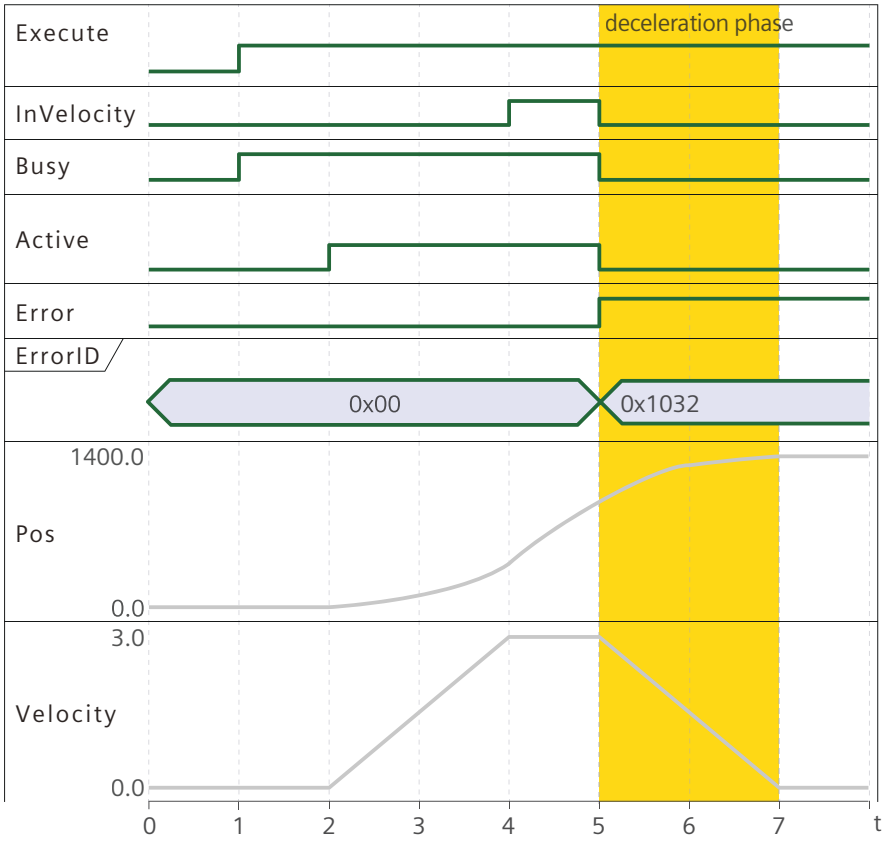
- The axis has been configured correctly.
- The axis is enabled.

Timing diagrams

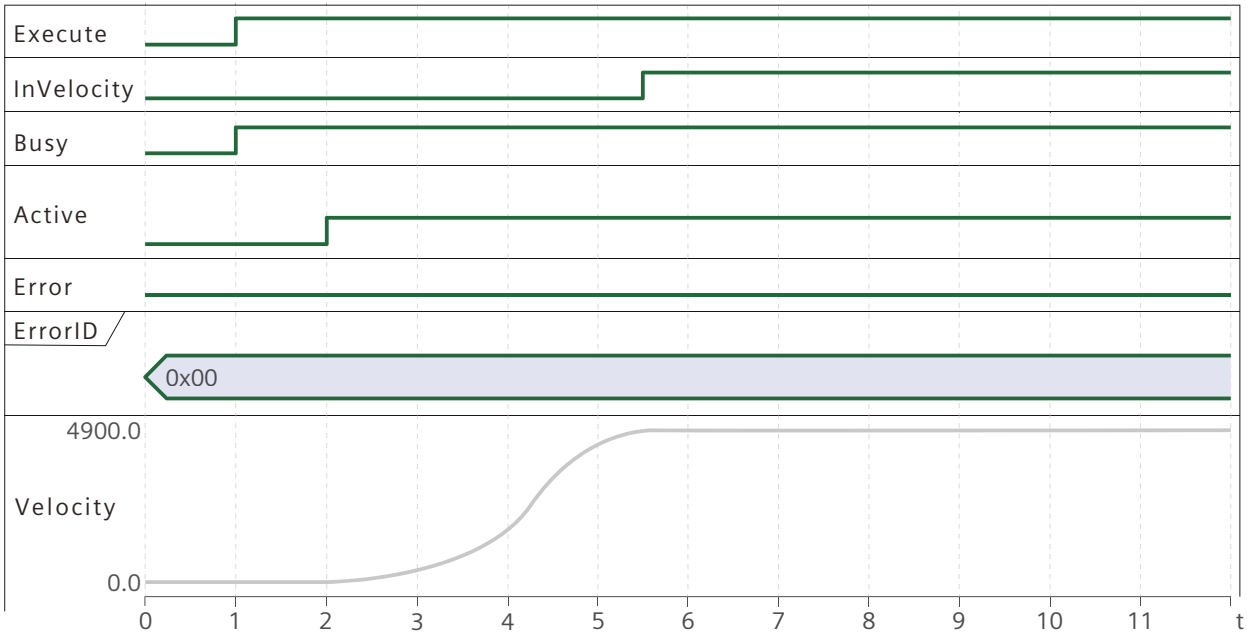
1. T-curve with constant acceleration and zero jerk



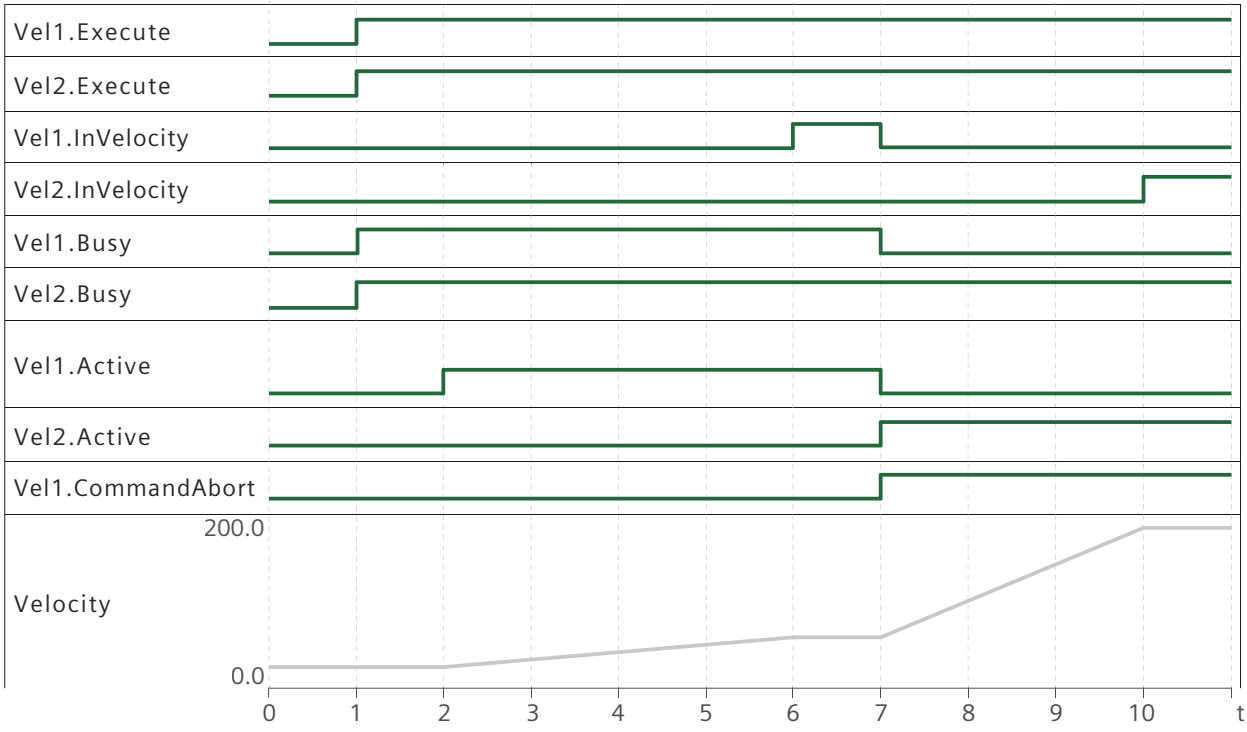
2. T-curve with constant acceleration and zero jerk (stops upon reaching a limit switch)
   Software limits are configured. When the position reaches the positive software limit setting of 999, deceleration begins, and the axis eventually stops at position 1400.



3. S-curve with variable acceleration and jerk greater than zero



4. Buffer an MoveVelocity instruction during the acceleration phase of another MoveVelocity instruction
   The buffered task only starts executing after the current task's InVelocity signal is active.



Override response

- The MC_MoveVelocity instruction can be aborted or buffered by the following motion control instructions:

Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_Halt MC_Stop MC_ImmediateStop MC_MoveVelocity MC_MoveAbsolute MC_MoveRelative MC_MoveJog MC_CamIn MC_GearIn	MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn

- A new MC_MoveVelocity instruction can abort or buffer the following motion control instructions:

Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveJog MC_CamOut MC_GearOut MC_MoveVelocity MC_CamIn MC_GearIn MC_PhasingRelative	MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn

Buffer other instructions after MC_MoveVelocity

- Buffer other instructions during the acceleration phase
  Busy is set when the buffer instruction's Execute is pulled high. It automatically ends and exits when the buffered task reaches InVelocity; CommandAborted of the buffered instruction is set, and InVelocity/Busy/Active are pulled low. At this point, the buffered instruction begins execution, and Active is set.
- Buffer other instructions during the constant velocity phase
  At this point, the buffered instruction has already reached InVelocity. Once the Execute of the buffering instruction is pulled high, the buffered instruction will immediately end and exit; CommandAborted of the buffered instruction is set, and InVelocity/Busy/Active are pulled low. The buffering instruction starts running immediately, and Active is set.

14.7.1.11 MC_ImmediateStop

MC_ImmediateStop instruction is used to stop the axis motion.

This instruction supports three axis types: PTO axes, virtual axes, and PN IRT axes, providing different stopping mechanisms based on the axis type:

- PTO Axis: The instruction performs an immediate stop.
- PN IRT axis, virtual axis: The instruction decelerates to a stop using the maximum deceleration configured for the axis.

Note

For torque control mode (MC_TorqueControl Mode 1), MC_ImmediateStop is not effective.

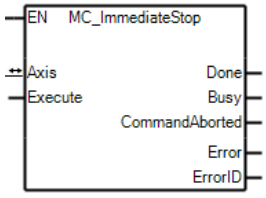
LAD/FBD	STL	Description
	**FBCALL MC_ImmediateStop, INSTANCE, Axis, Execute, Done, Busy, CommandAborted, Error, ErrorID**	This instruction commands axis immediate stop and transfers the axis to the state **Stopping**. It aborts any ongoing instruction execution. While the axis is in state **Stopping**, 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. As long as **Execute** remains TRUE or the zero value has not yet been reached, the axis remains in the Stopping state. As soon as **Done** is **TRUE** and **Execute** is FALSE the axis goes to state **Standstill**.

Table 14-6 Parameters for MC_ImmediateStop

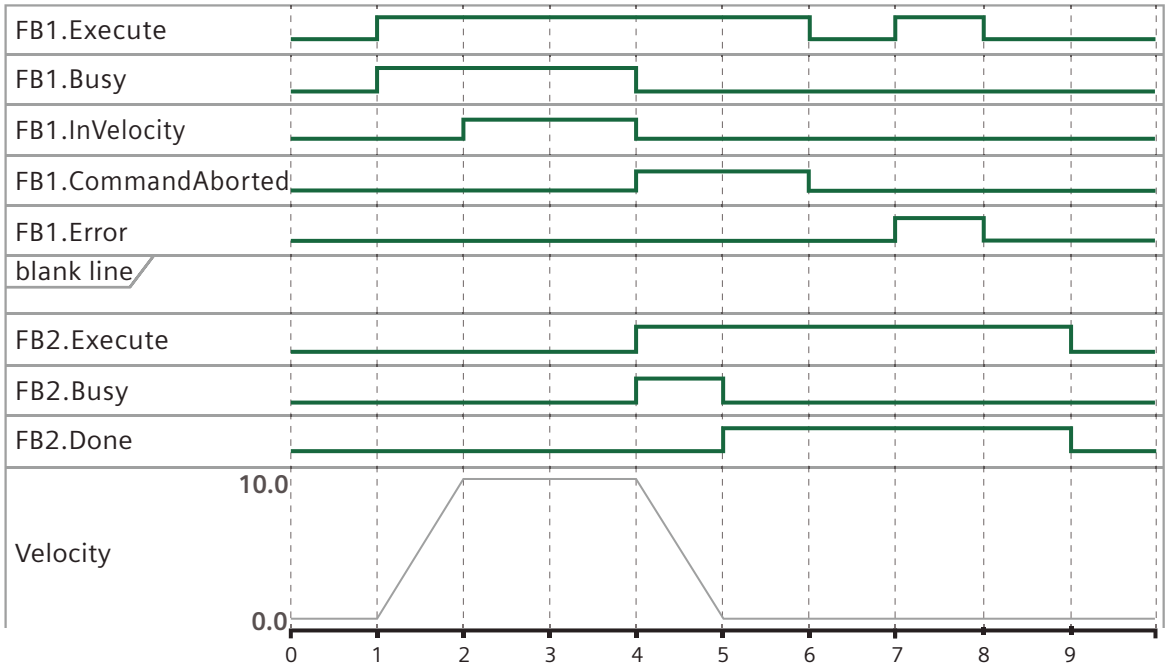
Parameter	Declaration	Data type	Initial value	Description
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Execute	IN	BOOL	-	The instruction is triggered on the rising edge.
Done	OUT	BOOL	False	Zero velocity reached
Busy	OUT	BOOL	False	The instruction is running
CommandAborted	OUT	BOOL	False	The current instruction is aborted by another instruction
Error	OUT	BOOL	False	An error has occurred with the instruction
ErrorID	OUT	WORD	0	Error code (Page 1015)

Preconditions

- The axis has been configured correctly.
- The axis is enabled.

Timing diagrams

- When MC_ImmediateStop interacting with MC_Velocity:
  FB1:MC_MoveVelocity --- FB2:MC_Immediate Stop



Override response

- The MC_ImmediateStop instruction cannot be aborted by any motion control instruction.
- A new MC_ImmediateStop instruction can abort the following motion control instructions:

Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_Home MC_Halt MC_Stop MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_MoveJog MC_CamOut MC_GearOut MC_CamIn MC_GearIn	

14.7.1.12 MC_MoveJog

Table 14-7 MC_MoveJog

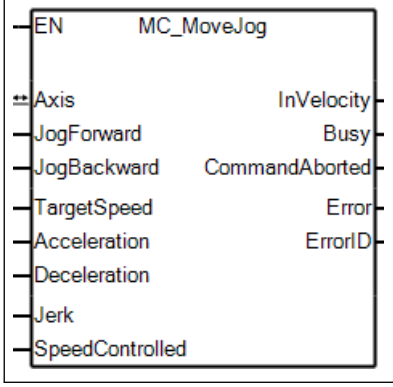
LAD/FBD	STL	Description
	**FBCALL MC_MoveJog, INSTANCE, Axis, JogForward, JogBackword, TargetSpeed, Acceleration, Deceleration, Jerk, SpeedControlled, InVelocity, Busy, CommandAborted, Error, ErrorID**	The MC_MoveJog instruction jogs an axis according to the specified target velocity.

Table 14-8 Parameters for MC_MoveJog

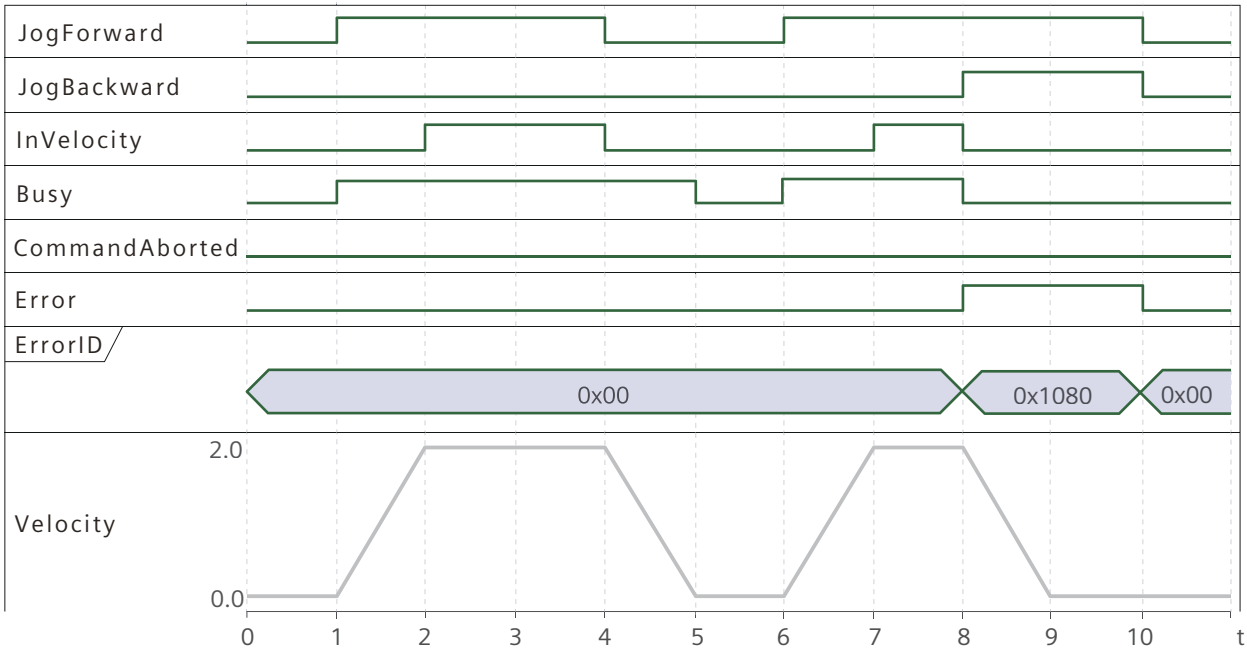
Parameter	Declaration	Data type	Initial value	Description
EN	IN	BOOL	-	Power flow
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
JogForward	IN	BOOL	-	Jogging in the forward direction, high-level triggered
JogBackward	IN	BOOL	-	Jogging in the reverse direction, high-level triggered
TargetSpeed	IN	REAL	-	Target velocity
Acceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Acc	Acceleration value
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	Value of deceleration
Jerk	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	Value of jerk
SpeedControlled	IN	BOOL	0	Control mode • 0: Position-controlled mode • 1: Speed control mode
InVelocity	OUT	BOOL	False	True when the given velocity equals the target velocity
Busy	OUT	BOOL	False	The instruction is running
CommandAborted	OUT	BOOL	False	The current instruction is aborted by another instruction
Error	OUT	BOOL	False	An error has occurred with the instruction
ErrorID	OUT	WORD	0	Error code (Page 1015)

Preconditions

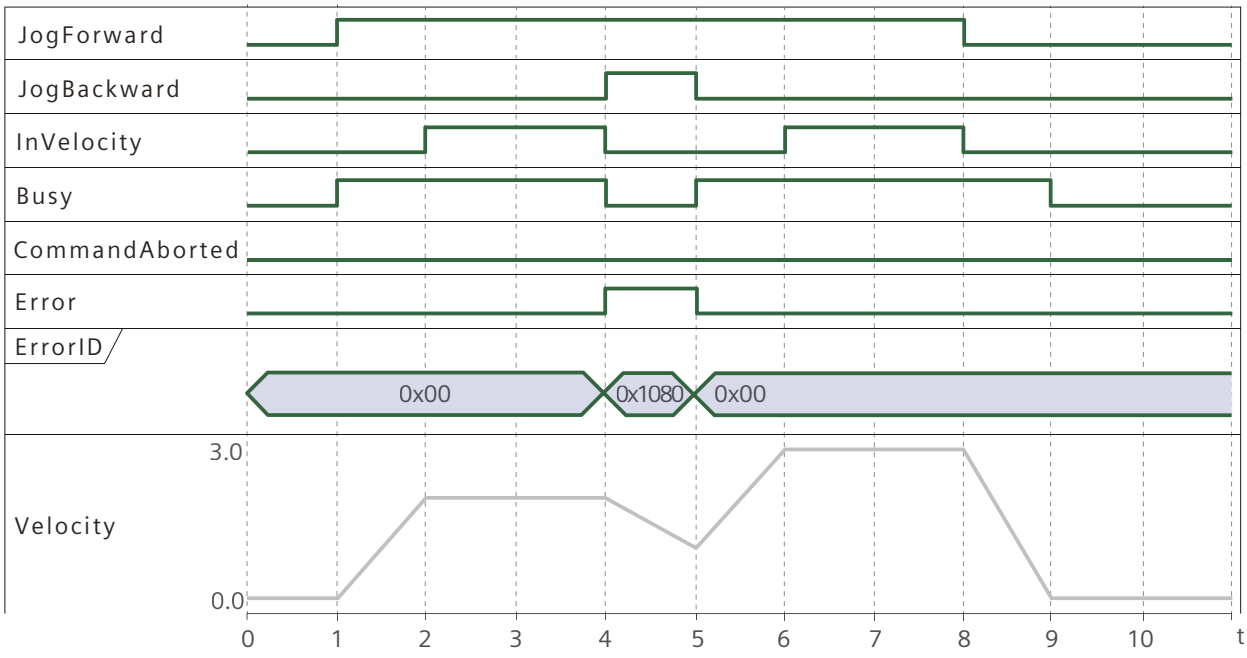
- The slave axis has been configured correctly.
- The axis is successfully enabled. The axis is successfully enabled via MC_Power in the user program, and the axis state machine enters the Standstill state.

Timing diagrams

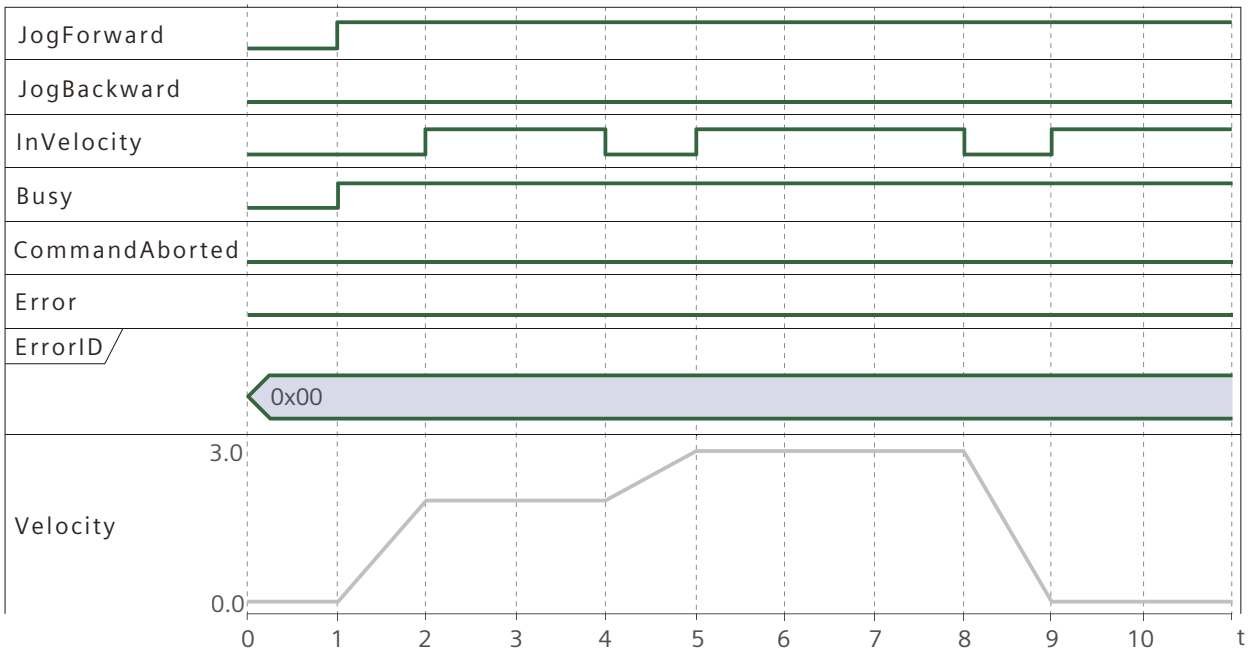
1. During forward jogging, triggering reverse jogging at the same time



2. When an 0x1080 error occurs during the jogging stop process, if a new jog motion is triggered, the stop motion will be interrupted immediately, and the new jog motion will be executed according to the new jogging parameters.



3. Adjusting velocity twice during motion


JogForward
JogBackward
InVelocity
Busy
CommandAborted
Error
ErrorID
0x00
Velocity
3.0
0.0
0
1
2
3
4
5
6
7
8
9
10
t

Override response

- The MC_MoveJog instruction can be aborted by the following Motion Control instructions:

Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_Halt MC_Stop MC_MoveJog MC_ImmediateStop MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn	

– A new MC_MoveJog instruction can abort the following motion control instructions:

Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamOut MC_GearOut MC_CamIn MC_GearIn MC_MoveJog MC_PhasingRelative	

Note

MC_MoveJog has no buffering behavior; it does not support buffering to other instructions, nor does it support being buffered by other instructions.

Modifying input pin parameters during motion will interrupt the MC_MoveJog process itself.

CAUTION

- Modifying the acceleration during the acceleration phase can easily cause mechanical shock if the difference between the old and new acceleration values is large.
- If an axis error occurs during jogging, the motion will not automatically resume after the error is cleared. You must pull JogForward or JogBackward low and then trigger the high level again.

14.7.1.13 MC_TorqueControl

Table 14-9 MC_TorqueControl

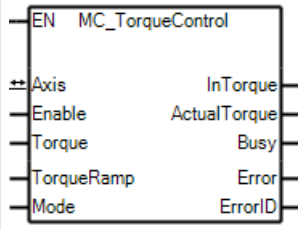
LAD/FBD	STL	Description
	`FBCALL MC_TorqueControl, Instance, Axis, Enable, Torque, TorqueRamp, Mode, InTorque, ActualTorque, Busy, Error, ErrorID`	Using the MC_TorqueControl instruction to perform torque control on a drive corresponding to an IRT axis. Torque data is transmitted via the Telegram 750.

Table 14-10 MC_TorqueControl parameter

Parameters	Declaration	Data type	Initial value	Description
EN	IN	BOOL	-	Power flow
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Enable	IN	BOOL	-	• Mode = 0 – High level: Enable instruction – Low level: Cancel instruction • Mode = 1 – High level: Enable the instruction and activate torque control mode simultaneously – Low level: Cancel the instruction and deactivate torque control mode simultaneously
Torque	IN	REAL	0.0	Can be omitted. • Mode = 0, representing additive torque • Mode = 1, representing the set torque value Valid Range: ±200% of the drive's reference torque Unit: Nm Once set to 1, Enable can continue to be updated.
TorqueRamp [1]	IN	REAL	0.0	Torque change per second. Valid Range: Greater than or equal to 0.0 Unit: Nm/S Once set to 1, Enable can continue to be updated.
Mode	IN	BYTE	0	Instruction Mode. • Mode = 0: Velocity mode, providing additive torque During the execution of the instruction, the torque control mode of the drive will not be activated (STW1.14=0). • Mode = 1: Torque control mode During the execution of the instruction, the torque control mode of the drive is activated (STW1.14=1). Updates are invalid after Enable being set to 1.
InTorque	OUT	BOOL	False	The set torque reaches the target torque setpoint.

Parameters	Declaration	Data type	Initial value	Description
ActualTorque	OUT	REAL	0.0	Actual torque when Enable set to 1; 0.0 when Enable set to 0.
Busy	OUT	BOOL	False	The instruction is running
Error	OUT	BOOL	False	An error has occurred with the instruction
ErrorID [2]	OUT	WORD	0	Error code (Page 1015)

[1] **Note**: When the initial value is 0.0, the Torque value is sent directly to the drive. In this case, the user has full control over how the Torque value changes. When the initial value is not 0.0, the torque gradually changes from the current Torque to the target Torque at the rate specified by the TorqueRamp.

[2] **Note**: If an error occurs during the execution of MC_TorqueControl , the drive's torque will be reset to 0.0. If in torque control mode (Mode = 1), the torque control mode will be deactivated.

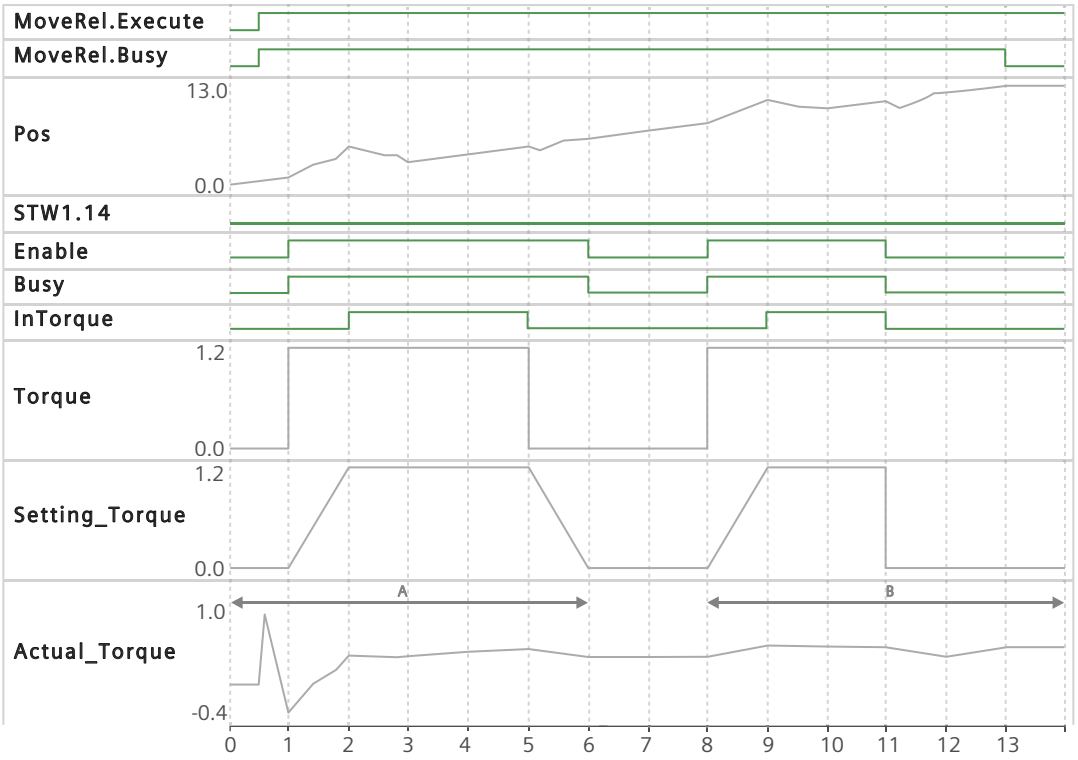
Preconditions

(Mode = 0 or Mode = 1):

- Correct configuration of the IRT axis
- Telegram 750 has been configured.
- In the axis object:
  - 1 on Drive Ready
  - 1 on Enable
  - No Error
- Axis Power Successful

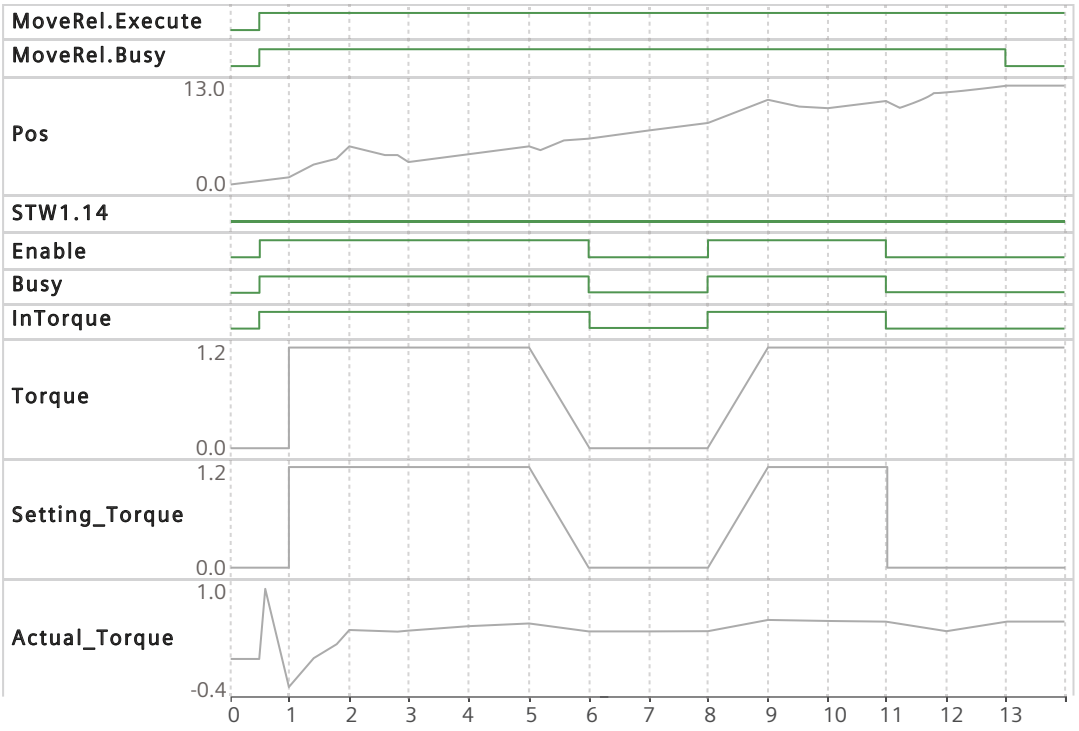
Timing diagrams

(1) Activate/Deactivate additive torque (Mode=0, TorqueRamp > 0.0)



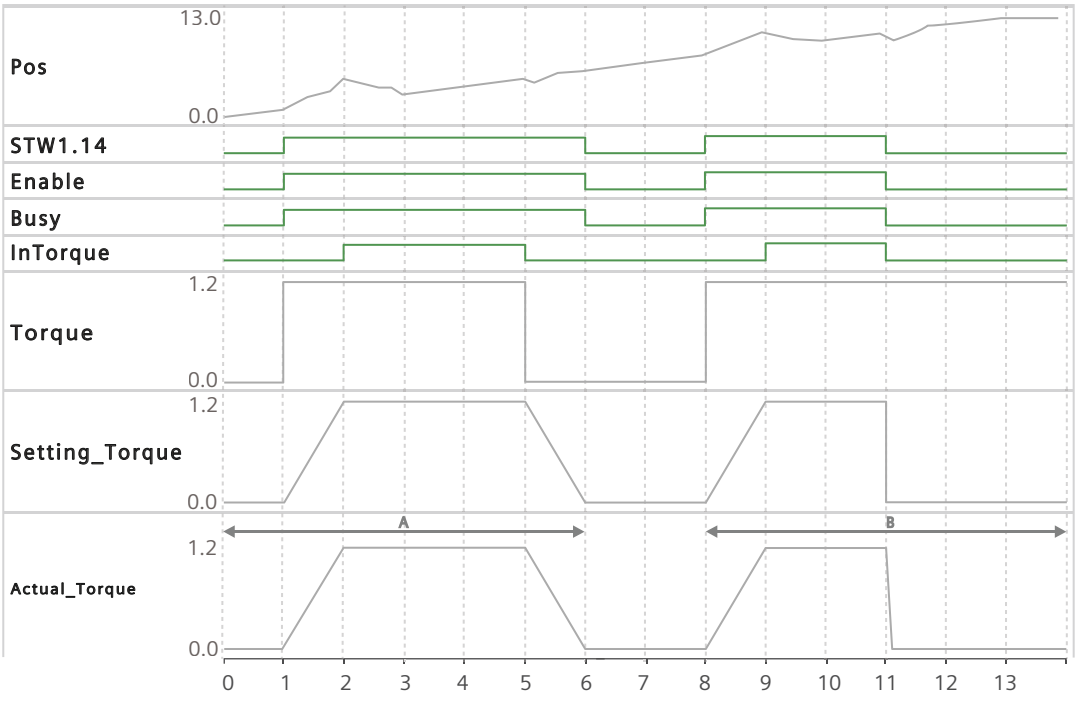
- In Process A, the user sets Enable = TRUE at time 1, while simultaneously changing the Torque value to 1.2. At this point, the set torque will gradually increase from 0.0 to 1.2 according to the TorqueRamp.
  Since the torque control mode is not activated, positioning instructions can also be executed. Since the axis is currently in motion, the actual torque is not zero; however, it will be influenced by the additive torque, causing the actual torque to change briefly. At the same time, the axis position will jump slightly, but since position closed-loop control remains primary, it will return to the planned position.
  At time 5, the user changes the Torque value to 0.0. At this point, the set torque will gradually decrease to 0.0 according to the TorqueRamp. Due to the change in torque, the position will also be affected, but it will subsequently return to the planned position.
- In Process B, the user sets Enable = TRUE at time 8 while simultaneously changing the Torque value to 1.2; this process is identical to Process A. Then, at time 11, the Enable signal is set to FALSE, and the additive torque value of 0.0 is sent directly to the drive. The pattern of position change is consistent with Process A.

(2) Activate/Deactivate additive torque (Mode=0, TorqueRamp = 0.0)



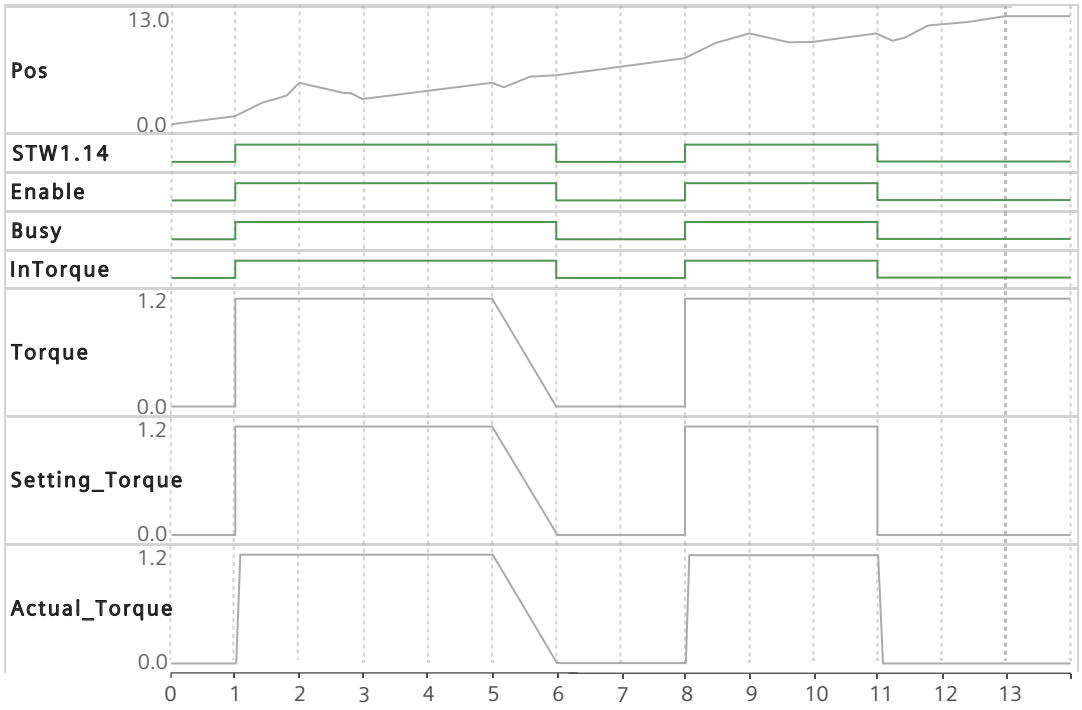
In this scenario, the motion behavior is consistent with (1); the difference lies in the "InTorque" pin. Since TorqueRamp = 0.0, the instruction sends the value from the Torque pin directly to the drive without a ramp process. This means the set torque value matches the target value immediately, so the InTorque pin becomes 1 (TRUE). Between times 5 to 6 and 8 to 9, the changes in the Torque pin value are caused by the user's program logic rather than the TorqueRamp.

(3) Activate/Deactivate torque control (Mode=1, TorqueRamp > 0.0)



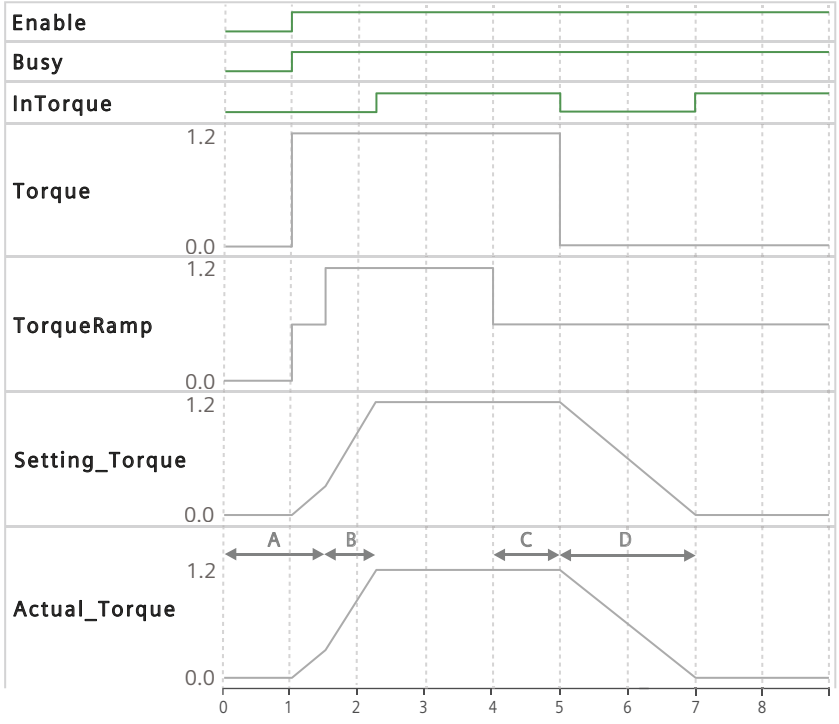
- In Process A, the user sets Enable = TRUE at time 1, while simultaneously changing the Torque value to 1.2. At this point, the set torque value will gradually increase from 0.0 to 1.2 according to the TorqueRamp. Once the set torque reaches the target value, InTorque is set to 1 (TRUE). As the torque control mode is activated, the actual torque will follow the changes in the set torque. It should be noted that the change in position at this time is caused by the torque and is unrelated to any motion instructions.
  At time 5, the user changes the Torque value to 0.0. At this point, the set torque will gradually decrease to 0.0 according to the TorqueRamp, and the actual torque will follow accordingly.
- In Process B, the user sets Enable = TRUE at time 8 while simultaneously changing the Torque value to 1.2; this process is identical to Process A. Then, at time 11, the Enable signal is set to FALSE, and the additive torque value of 0.0 is sent directly to the drive. The pattern of position change is consistent with Process A.

(4) Activate/Deactivate torque control (Mode=1, TorqueRamp = 0.0)



The phenomena defined in this scenario are similar to (2), except that the actual torque value follows the setpoint closely, as the system is in torque control mode.

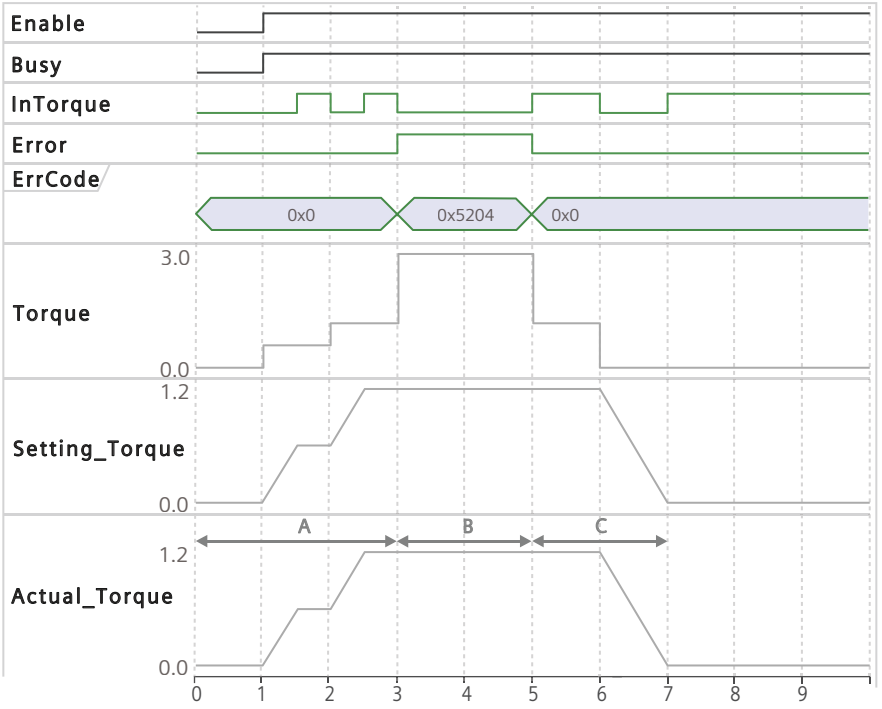
(5) Continuous updating of TorqueRamp



Taking Mode=1 and Torque=1.2 as an example.

- In Process A, the TorqueControl instruction is triggered. At time 1, the Ramp is set to 0.6, and the torque setpoint increases with a slope of 0.6.
- In Process B, at time 1.5, the user sets the Ramp to 1.2. From this point onward, the torque setpoint increases with a slope of 1.2, and the change in slope can be seen in the diagram.
- In Process C, since the user did not modify the target Torque, changing the Ramp to 0.6 will have no effect.
- In Process D, as the user modifies the target Torque to 0.0, the torque setpoint begins to decrease with a slope of 0.6 until it reaches 0.0.

(6) Error handling behavior for Torque and other parameter update errors



Taking Mode=1 and TorqueRamp=1.2 as an example, the drive's reference torque is 1.2, and the torque range is 200%.

- In Process A, the Torque pin value set by the user is within the permissible range, so the operation proceeds normally.
- In Process B, at time 3, the Torque pin value set by the user exceeds the range. At this point, an error is triggered, and the previously set Torque value is maintained.
- In Process C, at time 5, the user resets the Torque pin value to a valid range. The pin error disappears, and the operation returns to normal.

Note on use

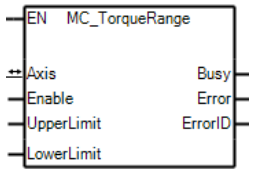
- Instruction Abort Behavior
  - Mode = 0
    The MC_TorqueControl instruction does not abort any motion instructions currently in execution. No motion instruction will cause the execution of MC_TorqueControl to be aborted. In this mode, MC_TorqueControl and motion instructions can coexist.
    The Power Off operation of the MC_Power management instruction will abort the running MC_TorqueControl. At this point, the Torque setpoint will be immediately set to 0.0.
    Axis errors (such as limit switch triggers, drive faults, etc.) will abort the running MC_TorqueControl. At this point, the Torque setpoint will be immediately set to 0.0.
  - Mode = 1
    The MC_TorqueControl instruction does not abort any motion instructions currently in execution. If a motion instruction is already running when MC_TorqueControl is executed, then MC_TorqueControl will report an error.
    While in torque control mode (STW1.14=1), no motion instructions can be executed. An error will be reported upon execution.
    The Power Off operation of the MC_Power management instruction will abort the running MC_TorqueControl. At this point, the Torque setpoint is immediately set to 0.0, and the system exits torque control mode (STW1.14=0).
    Axis errors (such as limit switch triggers, drive faults, etc.) will abort the running MC_TorqueControl. At this point, the Torque setpoint is immediately set to 0.0, and the system exits torque control mode (STW1.14=0).
- Instruction Buffering Behavior
  - Mode = 0
    Once the instruction is triggered, it is superimposed onto the current motion instruction without any buffer or abort behavior toward it. Furthermore, any buffer or abort behaviors of subsequent motion instructions remain unaffected.
  - Mode = 1
    MC_TorqueControl cannot be buffered after other instructions, nor can any other instructions be buffered after MC_TorqueControl.
- PLCOpen State Machine
  - Mode = 0
    While MC_TorqueControl is executing, it does not cause any changes to the axis state machine.
  - Mode = 1
    While MC_TorqueControl is executing, the axis remains in the Continuous state.
    When the user stops the MC_TorqueControl instruction normally, the axis enters the Standstill state.
    When an axis error occurs, the axis enters the "Error Stop" state.

Note

When MC_TorqueControl is already active, executing a new MC_TorqueControl instruction will not abort the previous one; instead, the new MC_TorqueControl will report an error.

14.7.1.14 MC_TorqueRange

Table 14-11 MC_TorqueRange

LAD/FBD	STL	Description
	**FBCALL MC_TorqueRange,INSTANCE , Axis, Enable, UpperLimit, LowerLimit, Busy, Error, ErrorID**	This instruction is used to set the real-time upper and lower torque limits of the drive. This function utilizes the SIEMENS additional telegram 750 to transmit real-time upper and lower torque limits to the drive via PROFIdrive.

Parameter	Declaration	Data type	Initial value	Description
EN	IN	BOOL	-	Power flow
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Enable	IN	BOOL	False	Level-triggered. Once Drive_Ready set to 1: • True： The upper and lower torque limit values in the instruction will be written to the drive in real-time, achieving custom torque limiting. • False： Restore the default upper and lower torque limits, and discard the limit settings of this instruction.
UpperLimit	IN	REAL	1.0 E6	Upper Torque Limit: This can be omitted. The valid range is [0, 200], representing a percentage of the Base Torque. When the limit is exceeded, the actual effective value is 2 × the Reference Torque. The upper limit cannot be negative, and it represents the motor's forward rotation. This parameter can be updated in real-time while the Enable bit is set.
LowerLimit	IN	REAL	1.0 E6	Lower Torque Limit: This can be omitted. The valid range is [-200, 0], representing a percentage of the Base Torque. When the limit is exceeded, the actual effective value is -2 × the Reference Torque. The lower limit cannot be positive, and it represents the motor's reverse rotation. This parameter can be updated in real-time while the Enable bit is set.
Busy	OUT	BOOL	False	The instruction is running
Error	OUT	BOOL	False	Signals that an error has occurred within the instruction
ErrorID	OUT	WORD	16#0000	Error code (Page 1015)

Note

Description of the Torque Reference Value

The Torque Reference Value is prioritized to be fetched from the drive. If the Torque Reference Value cannot be fetched from the drive, and you have checked "Use offline values" in STEP 7-Micro/WIN SMART V3, the Torque Reference Value configured in the software will be adopted instead.

If you have checked "Use offline values" but the configured offline Torque Reference Value does not match the online value fetched from the drive, the system will still prioritize the drive's online value and trigger Axis Warning Axis_Status.Config_Warning.

If the online Torque Reference Value cannot be fetched and you have not checked "Use offline values," a configuration error will occur, preventing the drive from being set to the "Ready" state.

Timing diagrams

Note

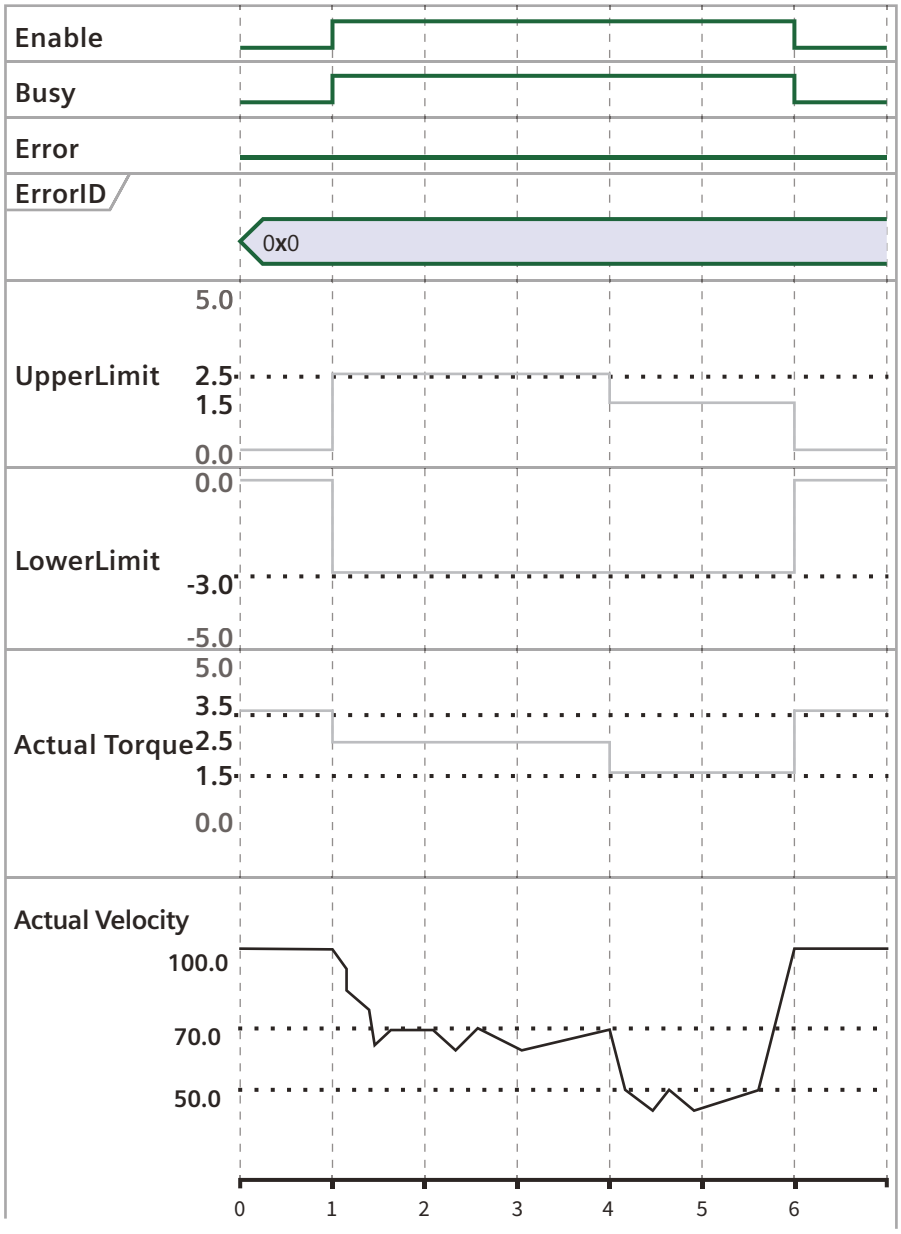
If a parameter setting error occurs during operation, the Error pin will be set to 1. Upon entering correct parameters again, the Error pin will be cleared, and the instruction will take effect with the new valid values.

Note

For clarity and comparison, all UpperLimit and LowerLimit values in the timing diagrams have been converted to torque units in Nm, although the actual FB (Function Block) pins use percentages of the reference torque.

- While the motor is running and generating torque (e.g., in InVelocity state), the torque limits can be set by enabling MC_TorqueRange via level-triggering and configuring the correct parameters. During operation, the torque limits can still be updated in real-time, and the new limits will take effect immediately.

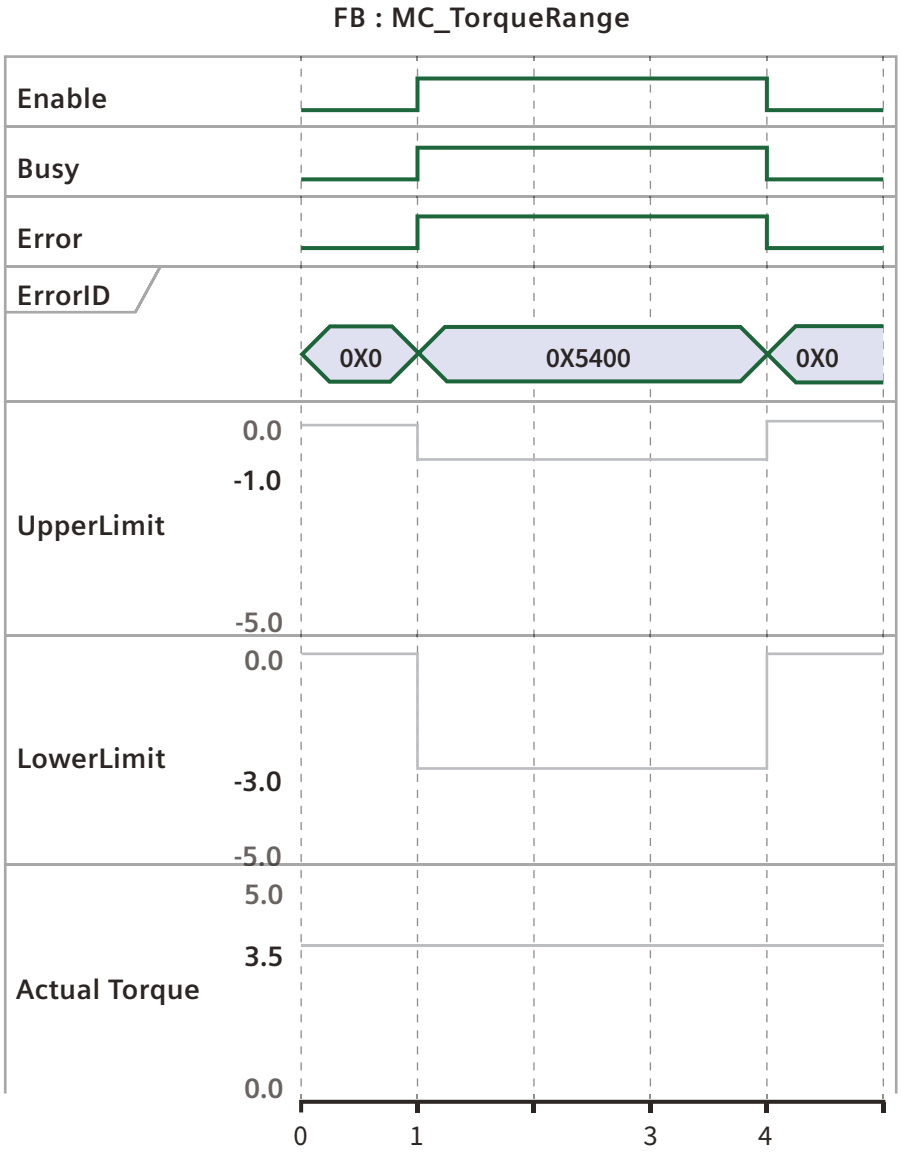
FB : MC_TorqueRange



Note

Note: The Actual Torque mentioned here represents only a peak value influenced by the limit settings. The actual torque is the fluctuation value generated during the drive's continuous attempt to reduce torque; it is not a fixed constant.

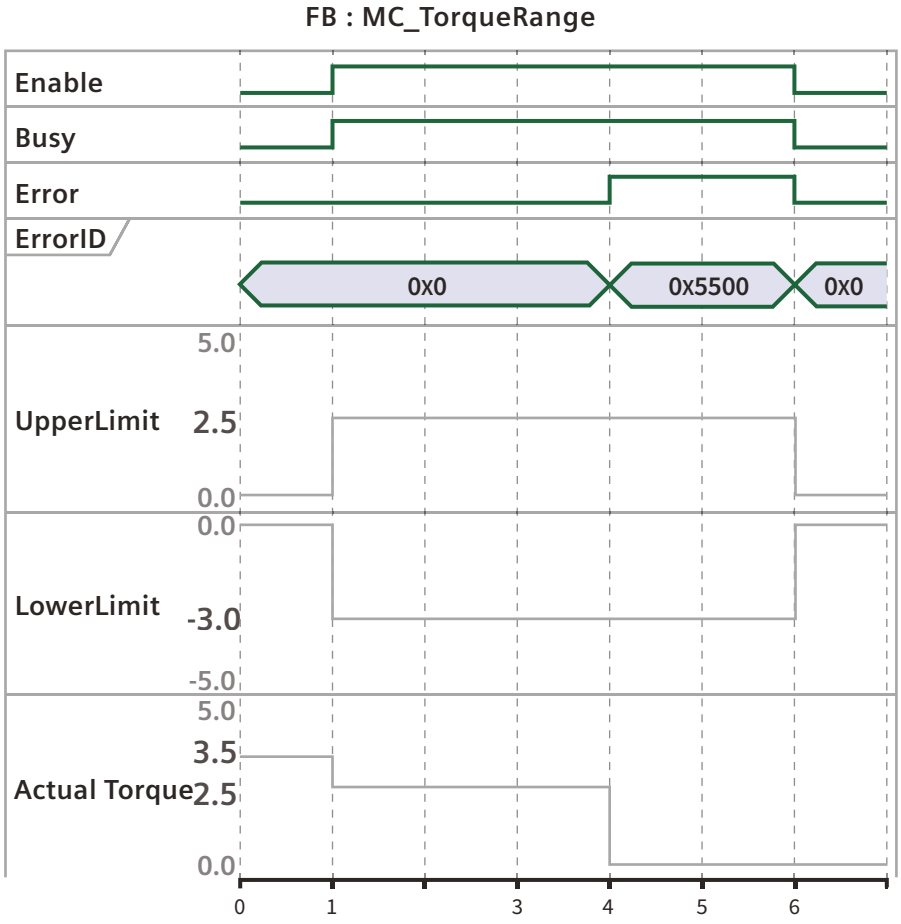
- While the motor is running and generating torque (e.g., in InVelocity state), enable MC_TorqueRange using level-triggering and set the torque limit. If an illegal parameter is set, the Error pin will be set to 1, and the current torque limit setting will not take effect.



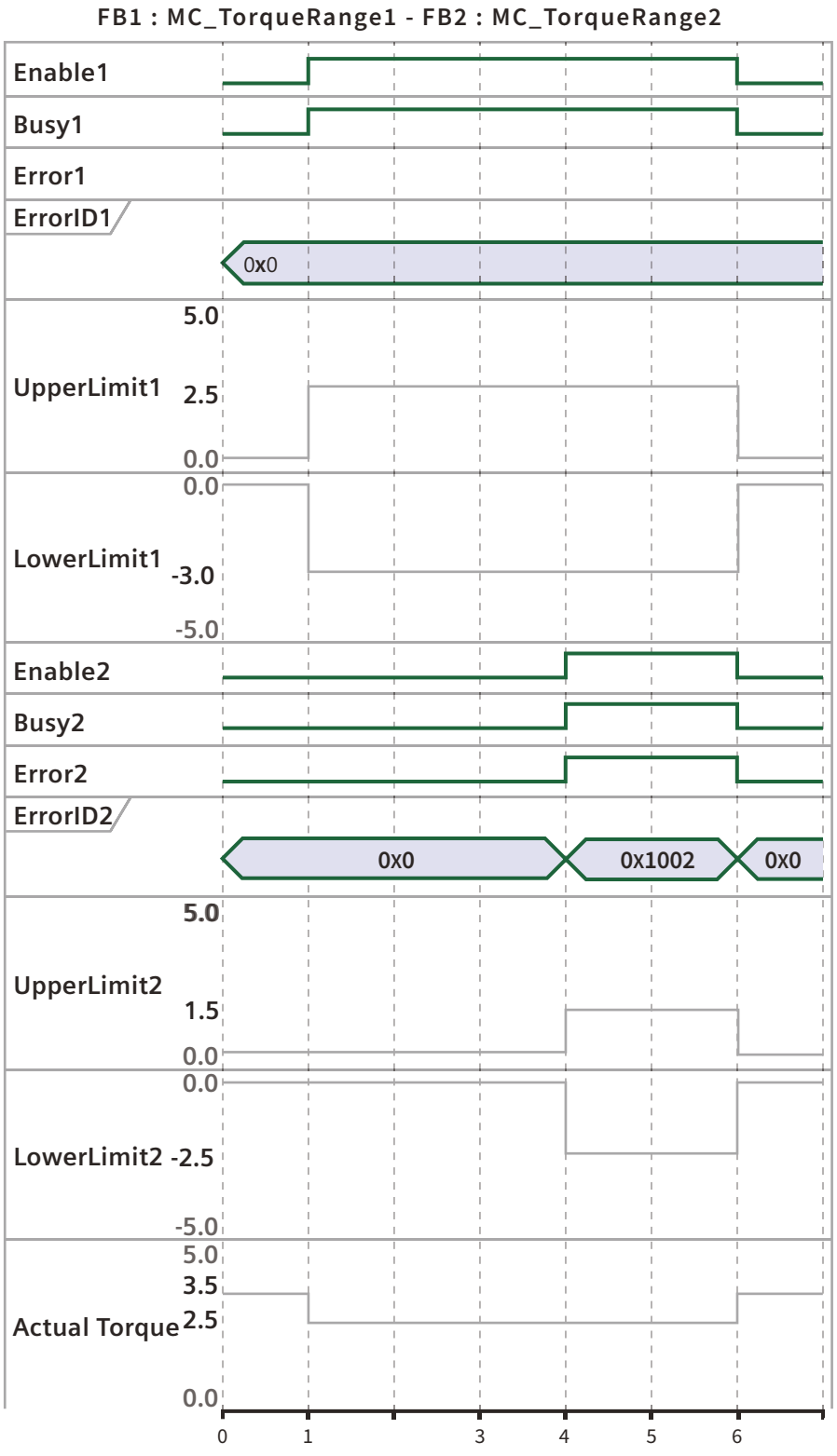
- While the motor is running and generating torque (e.g., in InVelocity state), enable MC_TorqueRange via level-triggering and ensure the parameters are set correctly. If a communication error occurs during operation, the Error pin will be set, and the torque limit control will be interrupted.

Note

In the event of a communication error, the system will no longer monitor the actual torque value of the drive. Under normal circumstances, the drive will trigger a fault stop, and the actual motor torque will return to 0.0.



- While the motor is running and generating torque (e.g., in InVelocity state), enable the first MC_TorqueRange via level-triggering and ensure the parameters are configured correctly. If a second MC_TorqueRange instance is triggered during operation, the second instance will report an error.


FB1 : MC_TorqueRange1 - FB2 : MC_TorqueRange2
Enable1
Busy1
Error1
ErrorID1
0x0
5.0
UpperLimit1
2.5
0.0
0.0
LowerLimit1
-3.0
-5.0
Enable2
Busy2
Error2
ErrorID2
0x0
0x1002
0x0
5.0
UpperLimit2
1.5
0.0
0.0
LowerLimit2
-2.5
-5.0
5.0
3.5
Actual Torque
2.5
0.0
0
1
3
4
5
6

NOTE

- Instruction Interrupt and Buffering Behavior (Override Response):
  - No other motion control instructions will abort the execution of "MC_TorqueRange".
  - The new "MC_TorqueRange" instruction will not abort any currently active motion control instructions.
  - Only one MC_TorqueRange instruction instance is permitted at any given time; any attempt to activate a subsequent instance will result in an error.
- Following error monitoring and parameter limits:
  Using MC_TorqueRange will affect following error monitoring; you must uncheck "Enable following error monitoring" in STEP 7-Micro/WIN SMART V3. For tolerance detection and Done bit set timeouts, you must manually adjust the relevant parameters in STEP 7-Micro/WIN SMART V3.
- Axis and drive-related notes:
  - MC_TorqueRange is an axis-based instruction; different axes can independently use their respective Range instruction.
  - You must pre-configure the drive's stall detection parameters (e.g., setting parameter P2175 of the S200 drive to a reasonable value); otherwise, an error will be triggered.
- Parameter errors and recovery behavior: If a parameter error occurs during operation, the Error pin will be set, and the values will be restored to their defaults. When the parameters are corrected, the Error pin will automatically clear, and the new parameters will take effect immediately.

14.7.1.15 MC_TorqueLimiting

Using the motion control instruction MC_TorqueLimiting, users can activate and parameterize the following functions:

- Torque/Force Limiting
- Fixed Stop Detection

You can set the default percentage limit value in the axis configuration.

The MC_TorqueLimiting instruction supports activation and deactivation either before a motion task starts or while a task is in progress.

Note

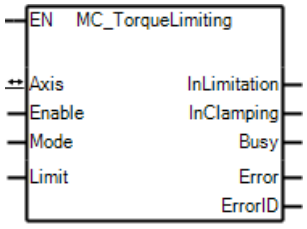
If a torque control instruction, such as MC_TorqueControl or MC_TorqueRange, is currently being executed, calling this instruction is not permitted.

Note

This instruction requires the drive to fully support Siemens Telegram 105.

Parameters

The following table lists the parameters for the motion control instruction MC_TorqueLimiting:

LAD/FBD	STL	Description
	`FBCALL MC_TorqueLimiting,INSTANCE, Axis, Enable, Mode, Limit, InLimitation, InClamping, Busy, Error, ErrorID`	The MC_TorqueLimiting instruction provides the following functions: • Activate and configure torque/force limiting • Activate and monitor fixed stop detection

Parameters	Declaration	Data type	Initial value	Description
EN	IN	BOOL	-	Power flow
Axis	IN_OUT	SYS_AXIS	-	Reference to an axis
Enable	IN	BOOL	-	Enable/disable the function corresponding to the input parameter "Mode" • TRUE: Start the instruction • FALSE: Attempt to deactivate the instruction [1]
Mode[2]	IN	BYTE		• 0: Torque/Force Limiting • 1: Fixed Stop Detection
Limit [3]	IN	REAL	0.0	Torque/force limit value, configured as a percentage (%). • When the parameter value is within the valid range of [0, 100]: The system will use the value specified by this parameter as the limit. • When the parameter is not explicitly specified: The system will automatically use the default value preset in the "Torque limit or fixed stop detection" configuration window. Related data tags: • Torque/force limitation: <TO>.Torque_Lmt.Torque_Force If the provided parameter value is outside the valid range [0, 100], the system will report an "Invalid Parameter" error.
InLimitation	OUT	BOOL	False	When Mode = 0 or 1: The actual torque output from the drive is greater than or equal to torque threshold value 2.
InClamping	OUT	BOOL	False	Mode = 1: The drive remains at the fixed stop (clamped). Axis position is within the positioning tolerance.
Busy	OUT	BOOL	False	The instruction is running
Error	OUT	BOOL	False	An error has occurred with the instruction
ErrorID	OUT	WORD	0	Error code (Page 1015)

[1] In the "InClamping" state, it is not permitted to deactivate the instruction

[2] Dynamic modification is not permitted while the instruction is in execution. Once the instruction has started successfully, switching modes will be ignored.

[3] Dynamic modification is permitted while the instruction is in execution.

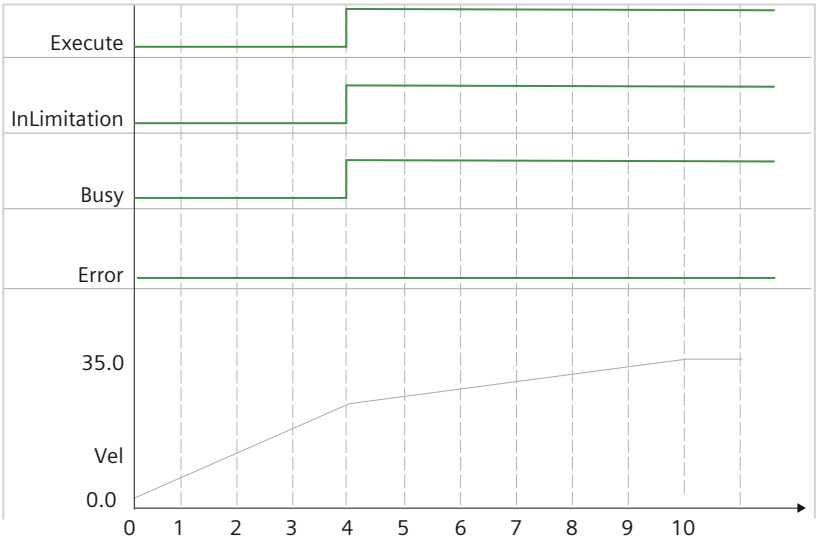
Precondition

- The axis has been configured correctly.
- The control loop has been correctly configured.

Torque Limiting Mode

In this mode, the instruction is used to control the upper torque limit of the drive in real-time and monitor whether the real-time torque has reached the set threshold.

- Enabling torque limiting during acceleration will limit the acceleration of the movement.

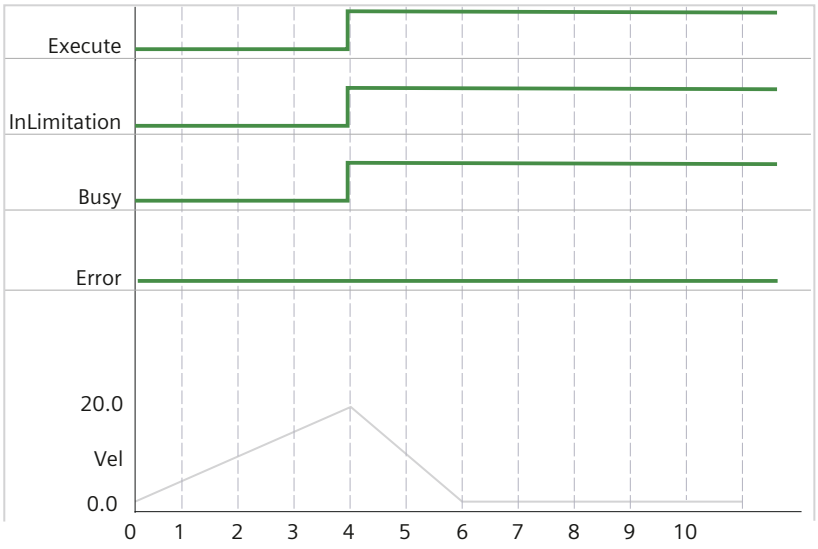


- If torque limiting is enabled during acceleration and the torque cannot overcome friction, the axis will be unable to continue accelerating; instead, it will begin to decelerate until it comes to a complete standstill.

WARNING

The "Runaway" phenomenon

If the torque limit is released immediately during this deceleration process, it may lead to uncontrolled acceleration of the axis, known as the "Runaway" phenomenon (in positioning mode).



Fixed stop mode

In this mode, the instruction is used to monitor for a fixed stop and clamp the axis against the fixed stop using the set torque value. Monitor the real-time torque to check if it has reached the set threshold and determine whether the axis is in a clamping state.

Note

This mode requires the motor's position loop to be active; it will not take effect if the motor is operating in a non-positioning loop, such as torque control mode.

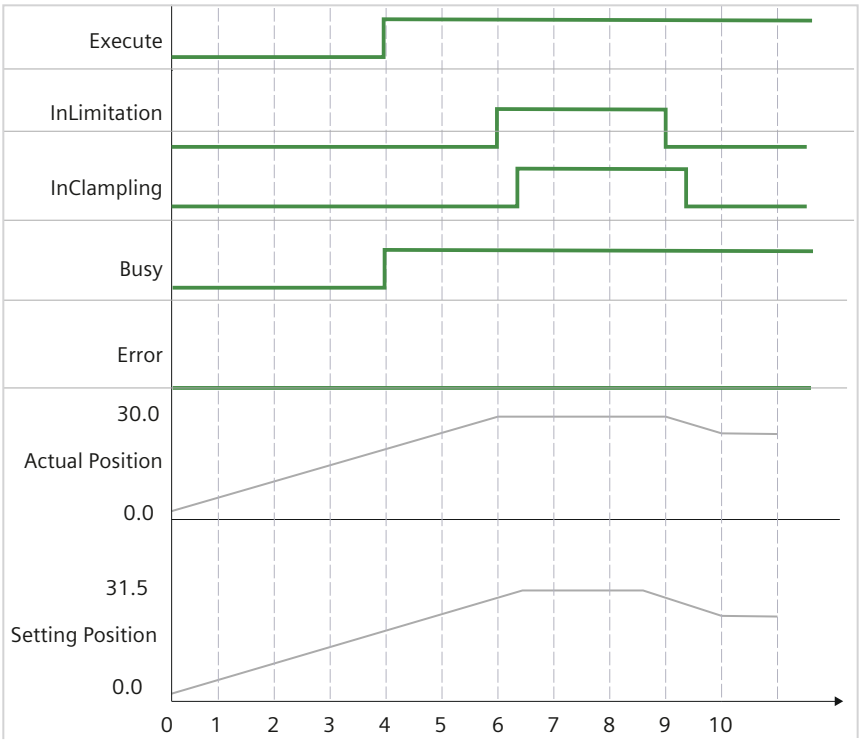
When the axis contacts a mechanical stop during normal or intended operation, the axis will stop and then enter the InClamping state; if a reverse instruction is initiated at this point, the InClamping state will be automatically cleared as the reverse movement begins.

Before enabling the instruction, you need to configure the Fixed Stop detection settings (Page 849) in the "Torque limit or fixed stop detection" window.

Explanation of unexpected breakaway (Page 850)

Note

If the set position is within the positioning tolerance, an unexpected breakaway cannot be detected.



Note

If the Enable signal is set to False while the axis is in the InClamping state, the FB instance will trigger an error.

State retention and error handling:

- During this process, status bits such as InLimitation, InClamping, and Busy will maintain their original states.
- The system will report the corresponding Error and ErrorID.
- It is worth noting that despite the error report, the axis will not enter the ErrorStop state; therefore, no reset operation is required.

14.7.2 Cam and Gear Instruction

14.7.2.1 MC_CamIn

This instruction enables synchronous camming, allowing the phase (master axis) and displacement (slave axis) to operate in sync according to a cam profile. The cam profile specified by this instruction must be created in advance using the Cam Editor and downloaded to the PLC. To modify the cam profile while the PLC is running, please refer to MC_GenerateCamTable (Page 1000).

The instruction is triggered at the rising edge of Execute.

The supported types of master and slave axes for this instruction are listed in the table below:

Slave axis	Master axis			
	IRT	PTO	Virtual	Encoder
IRT	✓	-	✓	✓
PTO	-	✓	✓	✓
Virtual	✓	✓	✓	✓

Note

PTO and IRT cannot be strictly synchronized. If a Virtual Axis is used to implement Cam cascading (e.g., IRT → Virtual → PTO), the synchronization relationship between IRT and PTO still cannot be guaranteed.

Note

The slave axis number must be greater than the master axis number.

Note

During synchronous motion, the Setting_Info.Jerk variable of the slave axis always displays as 0.

Note

When configuring the cam master axis phase, ensure that the value is sufficiently large to avoid poor synchronization results.

For example, in the following configuration, the displacement of the master axis within one interpolation cycle is exactly equal to the total master axis phase of the cam (1 mm). This means the slave axis must complete the entire motion defined by the cam profile within 1 ms. For most practical applications, achieving high-quality synchronization under these conditions is extremely difficult.

- Total master axis phase of the cam: 1 mm
- Motion interpolation cycle: 1 ms
- Master axis speed: 1000 mm/s

⚠ WARNING

Potential dangesr of sudden velocity changes in synchronized axes

If the velocity of the master 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, avoid executing MC_Home or MC_ImmediateStop instructions on the master axis whenever possible. Executing the MC_Home/MC_ImmediateStop instruction on the master axis can cause the slave axis to track rapidly, which is very dangerous.

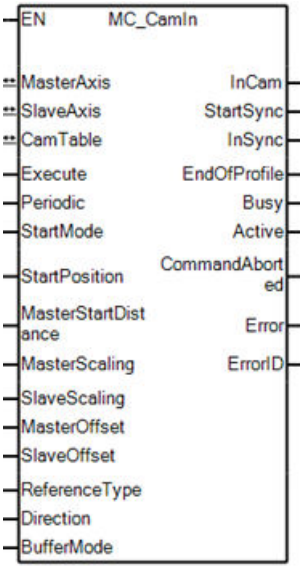
If you want to use the MC_Home/MC_ImmediateStop instruction on the master axis, release the relationship between the master and slave axes before executing this instruction.

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.
- PTO: EndOfProfile is advanced a maximum of 24ms relative to the end of the cam table.
- IRT: EndOfProfile will not lead relative to the cam profile end point, and the maximum lag will not exceed 2 IRT cycles.

Parameter

LAD/FBD	STL	Description
	**FBCALL MC_CamIn, INSTANCE, MasterAxis, SlaveAxis, CamTable, Execute, Periodic, StartMode, StartPosition, MasterStartDistance, MasterScaling, SlaveScaling, MasterOffset, SlaveOffset, ReferenceType, Direction, Buffermode, InCam, StartSync, InSync, EndOfProfile, Busy, Active, CommandAborted, Error, ErrorID**	The MC_CamIn instruction starts a cam operation by using a specified cam table.

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)

Parameter	Declaration	Data type	Initial value	Description
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
SlavingScaling	IN	REAL	1.0	Zoom in/out slave axis displacement by the specified ratio
MasterOffset [1]	IN	REAL	0.0	Master axis table offset; angular offset from master axis to cam.
SlaveOffset [1]	IN	REAL	0.0	Slave axis table offset; sharpened cam (i.e., higher peaks and deeper valleys).
Reference-Type [2]	IN	BYTE	0	Specify the position type of the master axis. • 0: Instruction location (calculated value within the same task cycle) • 1: Feedback location (master axis feedback position acquired within the same task cycle) When using a non-cyclic cam with Reference Type = 1, a Hysteresis value (Page 861) must be configured. This is to prevent the fluctuation of the actual position from triggering an EOP during CamIn, which would result in the termination of cam synchronization. • 2: Instruction location (calculated value from the previous task cycle)
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
StartSync	OUT	BOOL	False	Catch-up behavior of the slave axis relative to the master axis • 0: The slave axis does not catch up with the master axis • 1: The slave axis begins catching up to the master axis but has not yet reached synchronization (not yet in synchronous state)
InSync	OUT	BOOL	False	TRUE when the slave axis is synchronized with the master axis.
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

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 1020)

[1] PTO: The input values for these four parameters will not take effect; the default parameters will be used instead. The default value for StartMode is 1 (Relative Motion Mode), while the default values for the other parameters are 0.0.

[2] When the slave axis is PTO, setting this parameter is not supported and it defaults to 0. When the slave axis is a virtual axis, setting it to 1 is equivalent to setting it to 0.

Note

For unidirectional cams using actual values as the reference type (ReferenceType=1), it is recommended to set an appropriate hysteresis value on the "Actual Value Inference" page of the master axis.

Preconditions

- The master and slave axes have been correctly configured.
- Successfully enable the slave axis via MC_Power in the user program.

Start Cam movement

After the start command is issued, wait for the master axis to reach StartPosition (the starting position of the cam table).
When the master axis passes through StartPosition, the starting point of the cam table is executed, and the output variable InCam (Cam motion in progress) changes to 1.

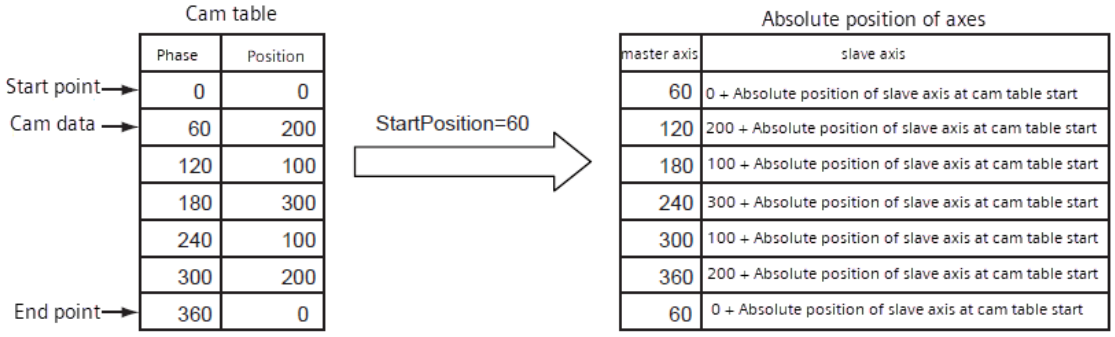
The phase and displacement of the cam table are specified as relative values from the zero point. Therefore, the absolute positions of each axis in each phase are relative values based on the absolute positions of the cam table's starting point for each axis. For example, when the counting mode of the 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.

For the cam table, both phase and displacement are defined as relative amounts starting from the starting point "0.0". Regarding the instruction positions of the master and slave axes. Interpolation is performed linearly between two sets of cam data to determine the displacement (slave axis) corresponding to the phase (master axis). When the number of cam data points is small, the phase interval increases, resulting in lower-precision cam motion. Conversely, when the number of cam data points is large, the phase interval decreases, resulting in higher-precision cam motion.

Note

The criteria for determining if StartPosition has been reached

- The absolute position of the master axis crosses the StartPosition (including both forward and backward crossings)
- The absolute position of the master axis is exactly at the StartPosition



Furthermore, when the master axis passes through the MasterStartDistance (master axis tracking distance), the slave axis begins its cam motion, and the output variable StartSync (start catching up) changes to 1. MasterStartDistance is specified as either an absolute position or a relative distance from the StartPosition (the starting position of the cam table). The absolute position or relative position is configured via StartMode (starting position specification method).

Example: Slave axis movement varies depending on StartPosition

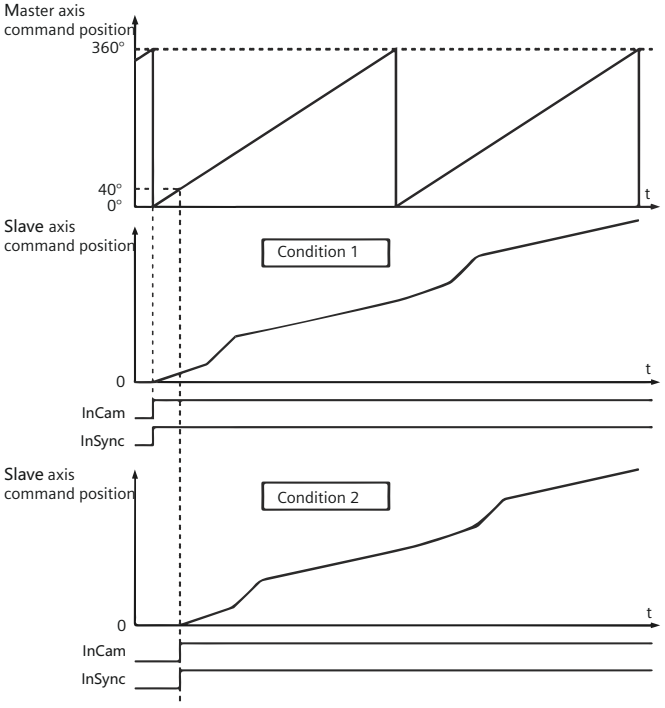
The cam table is configured as follows:

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 cam motion are as follows:

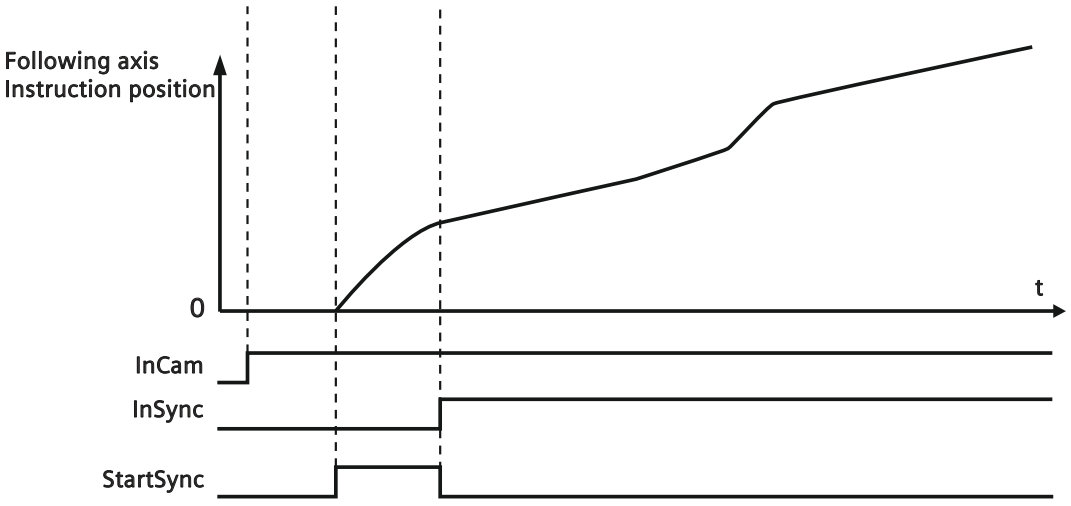
Input variable	Condition 1	Condition 2
Periodic (Repeat mode)	TRUE(Repeat)	TRUE(Repeat)
StartMode(Relative mode)	1	1
StartPosition	0	40
MasterStartDistance	0	0

- 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.



Note

When MasterStartDistance and StartPosition are not at the same point, the slave axis begins catching up when the master axis is at the MasterStartDistance position. Please refer to the following timing diagram:



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.

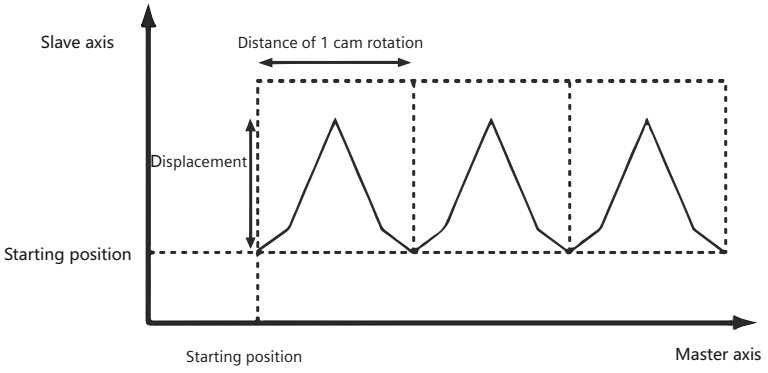
When FALSE [No repeat] is set, the cam motion is ended if execution continues to the end of the cam table.

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).

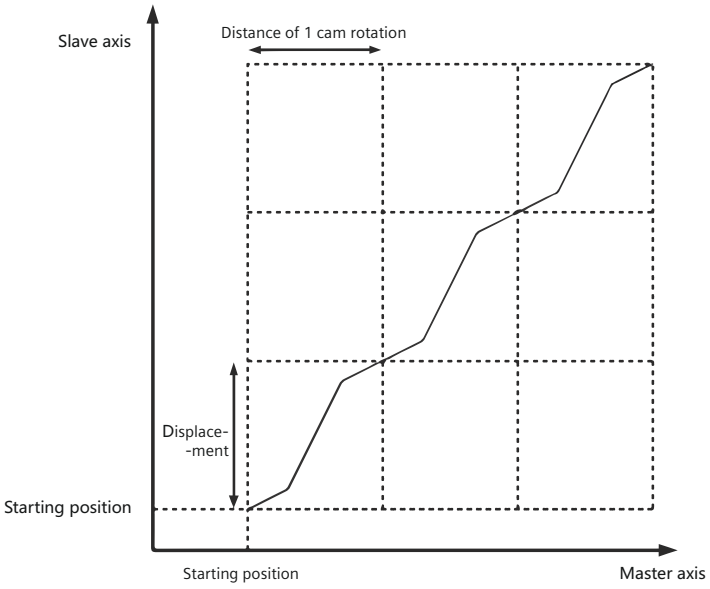
If the travel positions of the slave axis at the start and end are different, the movement is performed as a transfer cam (see Transfer Cam movement).

In the figure below, the horizontal axis shows the position of the master axis and the vertical axis shows the position of the slave axis.

Reciprocating cam movement



Transfer cam movement



EndOfProfile (Cam cycle complete)

Within one cycle of outputting the instruction value for the cam motion, EndOfProfile (Cam cycle complete) is set to 1. This cam motion utilizes the phase and displacement defined by the endpoint of the cam table.

By setting an absolute position of the master axis for the StartPosition (the starting position of the cam table), the cam table serves as a relative table starting from that specific position.

Note

When configuring the EndOfProfile pin, please note:

- For non-periodic cams, EndOfProfile is set to 1 and held upon completion of the cam cycle.
- For periodic cams, a scan cycle is set at the end of each cam cycle.
- PTO: EndOfProfile is set to 1 at a maximum of 24 ms before the endpoint of the cam table.
- IRT: EndOfProfile will not occur ahead of the cam table endpoint. The lag will not exceed a maximum of 2 IRT cycles.

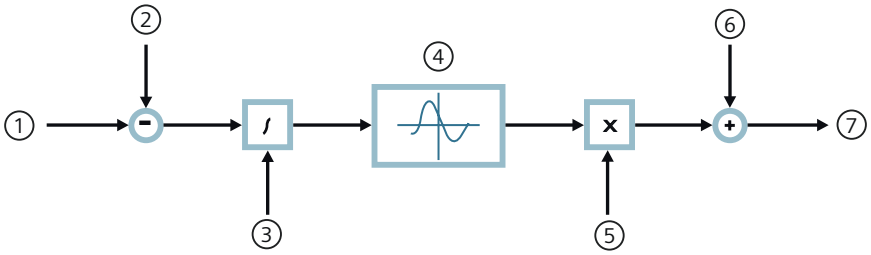
EndOfProfile (Cam cycle complete) is the output that represents the endpoint of the cam table.

End Cam movement

To stop the cam motion mid-operation, use MC_CamOut (Cam Decouple), MC_ImmediateStop, or MC_Stop (Forced Stop).

Scaling and Offsetting Cams

The following figure illustrates the general sequence for scaling and shifting (offsetting) a cam:



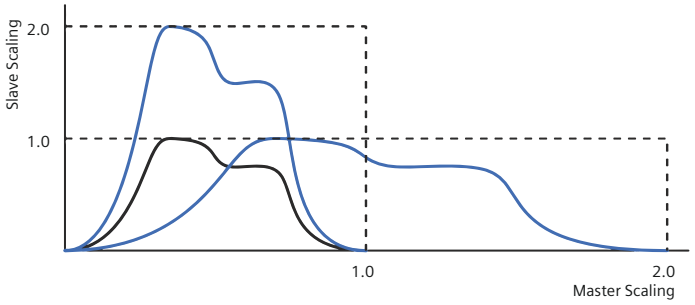
1. Leading axis position
2. Master offset
3. Value scaling
4. Cam
5. Slave scaling
6. Slave offset
7. Following axis position

Note

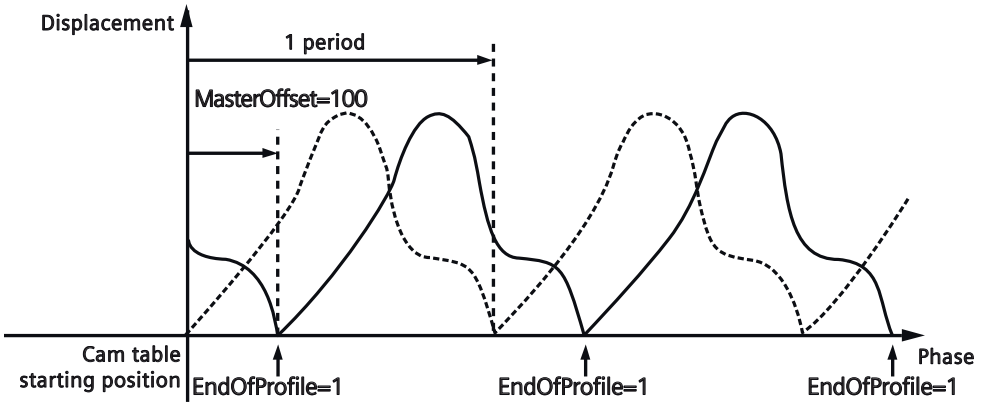
restrictions

PTO functions do not support offsets

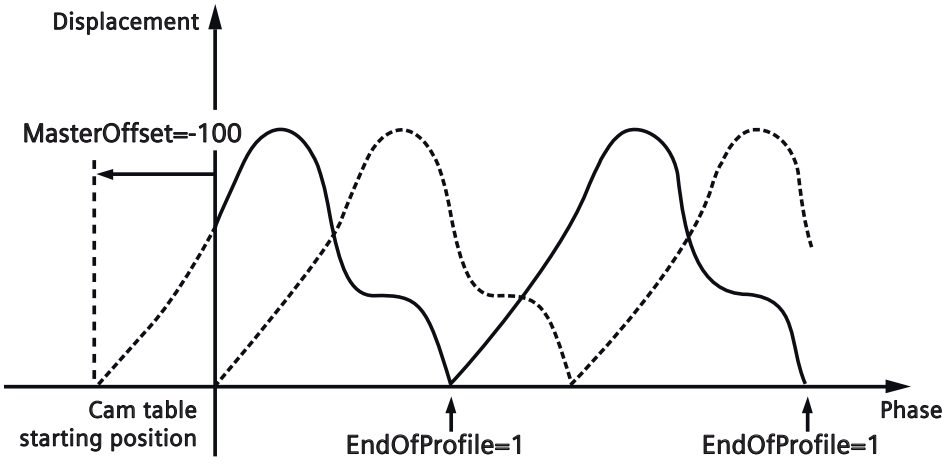
- Coefficient (Scaling)
  The master axis phase and slave axis displacement for a specified cam table can be scaled according to a designated ratio.
  Different scaling factors can be set for both the master axis and the slave axis.



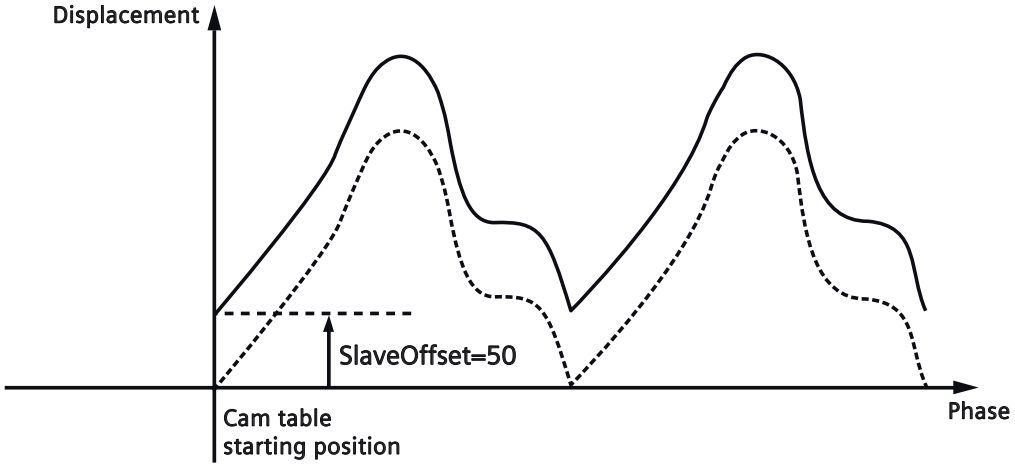
- Biasing
  For a specified cam table, the phase and displacement can be shifted according to a defined offset. Different offsets can be specified for the master axis phase and the slave axis displacement.
  - When MasterOffset (master axis offset) > 0:



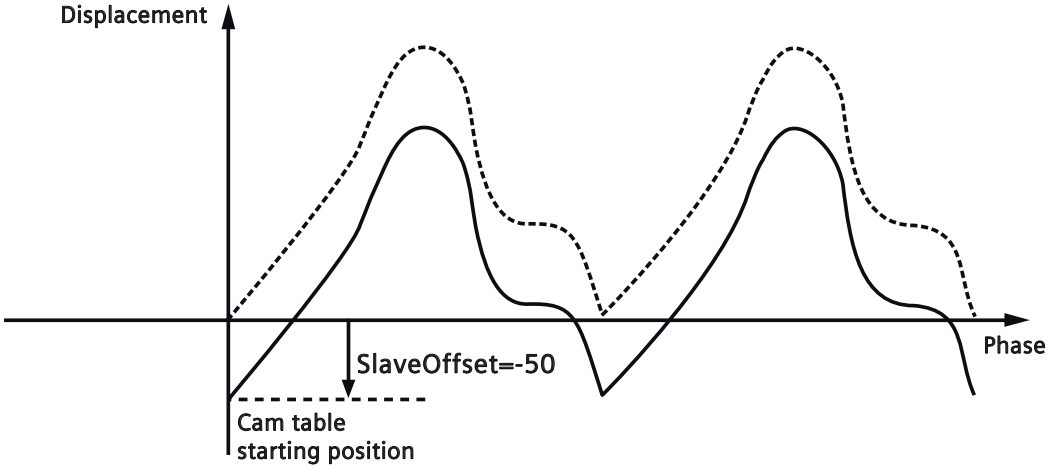
  - When MasterOffset (master axis offset) < 0:



– When SlaveOffset (slave axis offset) > 0:



– When SlaveOffset (slave axis offset) < 0:



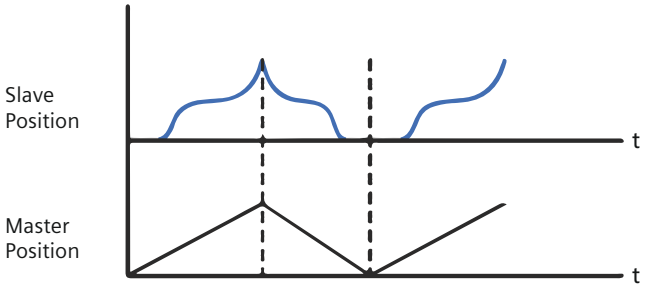
Direction

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.

Direction is valid only when InSync is 1.

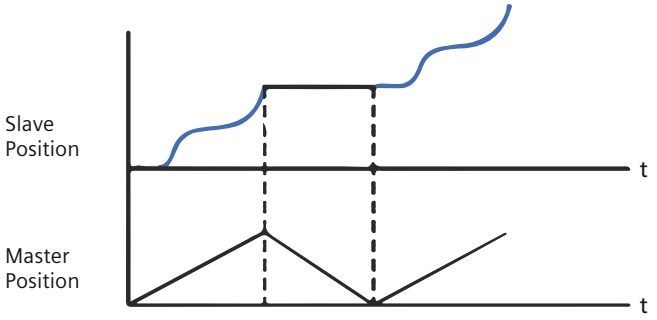
No direction specified

Cam movement starts regardless of master axis movement in positive or negative direction.



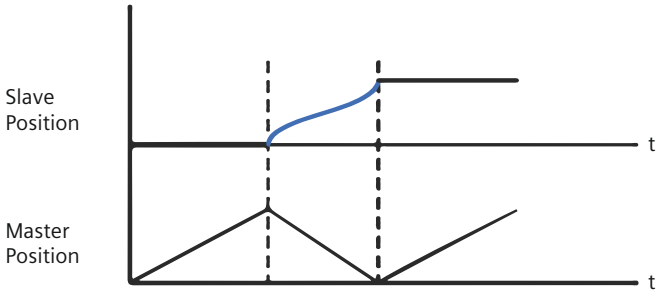
Positive direction

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.



Slave axis overspeed behavior

Axis type	Prior to synchronization state (overspeed behavior)	After reaching synchronization state
PN IRT axis Virtual axis	When the slave axis overspeed occurs, the system will immediately enter catch-up status. The slave axis will accelerate to catch up with the master axis until the synchronization state is restored.	Cyclic cam • After the slave axis overspeed occurs, it will continue to operate at the maximum velocity specified in the axis configuration. • At this time, the StartSync flag is set to 1, the InSync flag is set to 0, and the EOP (End of Path) flag is set to 0. Other parameters remain unchanged. The VelLimit flag in the slave axis TO (Technology Object) will be set to 1, indicating that the velocity limit has been activated. • Once the master and slave axes enter the synchronization state again, the StartSync flag will be set to 0, and the InSync flag will be set to 1. At the same time, the VelLimit flag in the slave axis TO (Technology Object) will be set to 0, and the system will return to the normal synchronized operation state. Non-cyclic cam • If the slave axis is in an overspeed state when the master axis reaches the EOP point, the slave axis will continue to catch up until it reaches that EOP point. • Once the slave axis reaches the EOP point, it will exit CamIn mode, and the EOP flag will be set to 1 (this behavior is similar to the Done bit in other instructions).
PTO axis	When the slave axis overspeed occurs, the system will immediately report an error and exit the current operation.	

Override response

The MC_CamIn instruction can be aborted or buffered by the following motion control instructions:

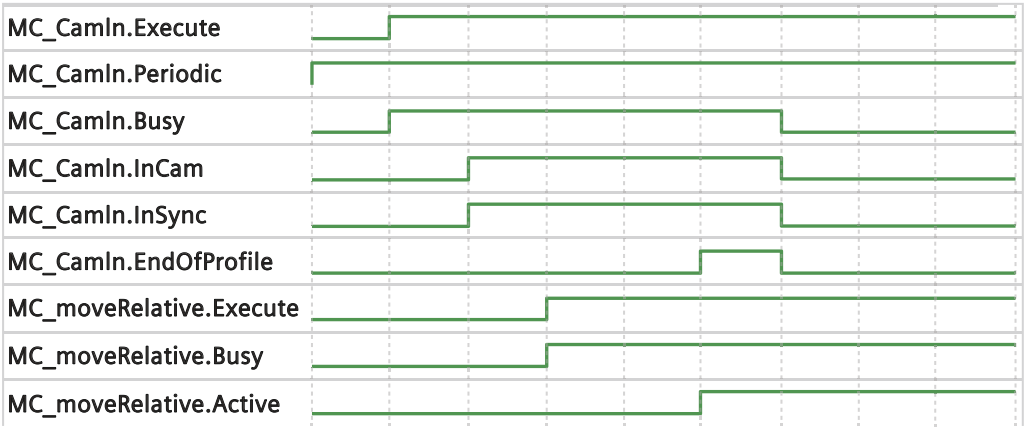
Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_Halt MC_Stop MC_ImmediateStop MC_MoveVelocity MC_MoveAbsolute MC_MoveRelative MC_MoveJog MC_CamIn MC_GearIn MC_CamOut	MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn

- A new MC_CamIn instruction can abort or buffer the following motion control instructions:

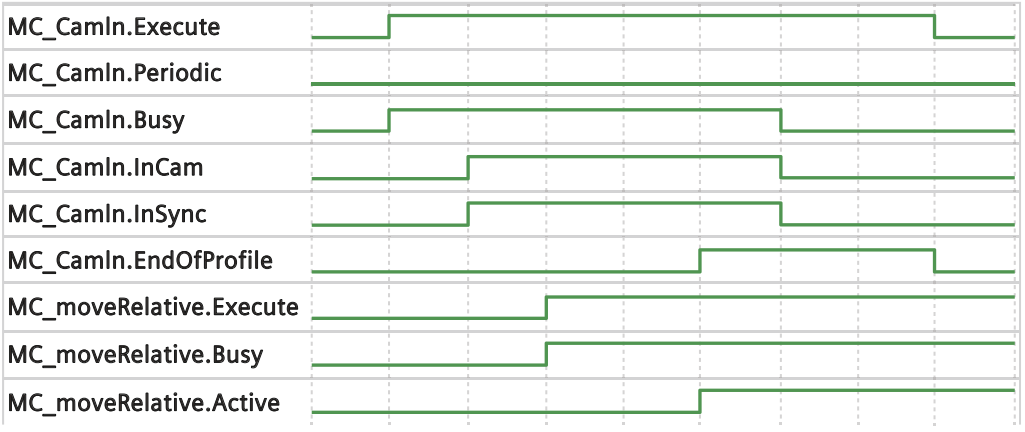
Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveJog MC_CamOut MC_GearOut MC_MoveVelocity MC_CamIn MC_GearIn MC_PhasingRelative	MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn

When MC_CamIn is buffered by another instruction, the rising edge of EOP and the rising edge of the subsequent instruction's Active occur in the same cycle; meanwhile, the output pins remain False, and CommandAborted will not be set to high. Please refer to the following timing diagram:

- Timing diagram for buffered cyclic cams:



- Timing diagram for a non-cyclic cam being buffered:



14.7.2.2 MC_CamOut

Terminates the cam synchronization motion of the axis specified by the Axis input pin.

Table 14-12 MC_CamOut

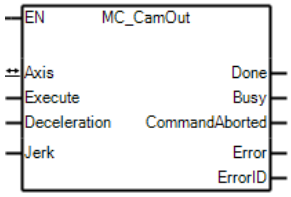
LAD/FBD	STL	Description
	**FBCALL MC_CamOut, INSTANCE, SlaveAxis, Execute, Deceleration, Jerk, Done, Busy, CommandAborted, Error, ErrorID**	Disable MC_CamIn cam behavior and decelerate the axis to stop

Table 14-13 Parameters for MC_CamOut

Parameters	Declaration	Data type	Initial value	Description
EN	IN	BOOL	-	Power flow
Axis	IN_OUT	SYS_AXIS	-	Name of the axis
Execute	IN	BOOL	-	Triggered by a rising edge
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	Value of deceleration
Jerk	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	Value of jerk
Done	OUT	BOOL	False	IRT axis: In Standstill state PTO axis and virtual axis: When the axis velocity is 0
Busy	OUT	BOOL	False	The instruction is running
CommandAborted	OUT	BOOL	False	The instruction is aborted by another instruction.
Error	OUT	BOOL	False	An error has occurred with the instruction
ErrorID	OUT	WORD	0	Error code (Page 1020)

Note

Jerk parameter effectiveness rules

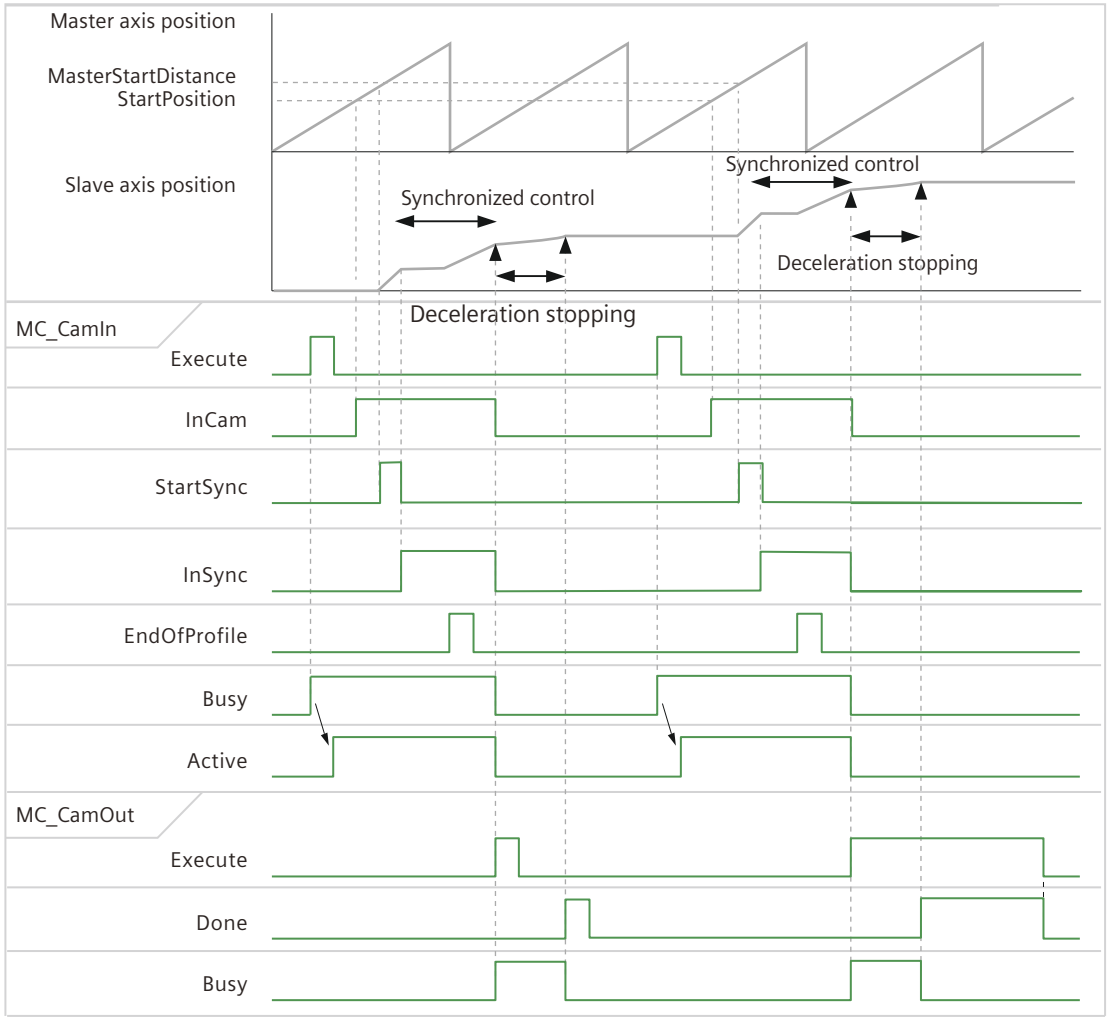
Starting from STEP 7 Micro/WIN SMART V3 (Version 3.2), the Jerk parameter for the CamOut function takes effect and is applicable to all axis types.

Preconditions

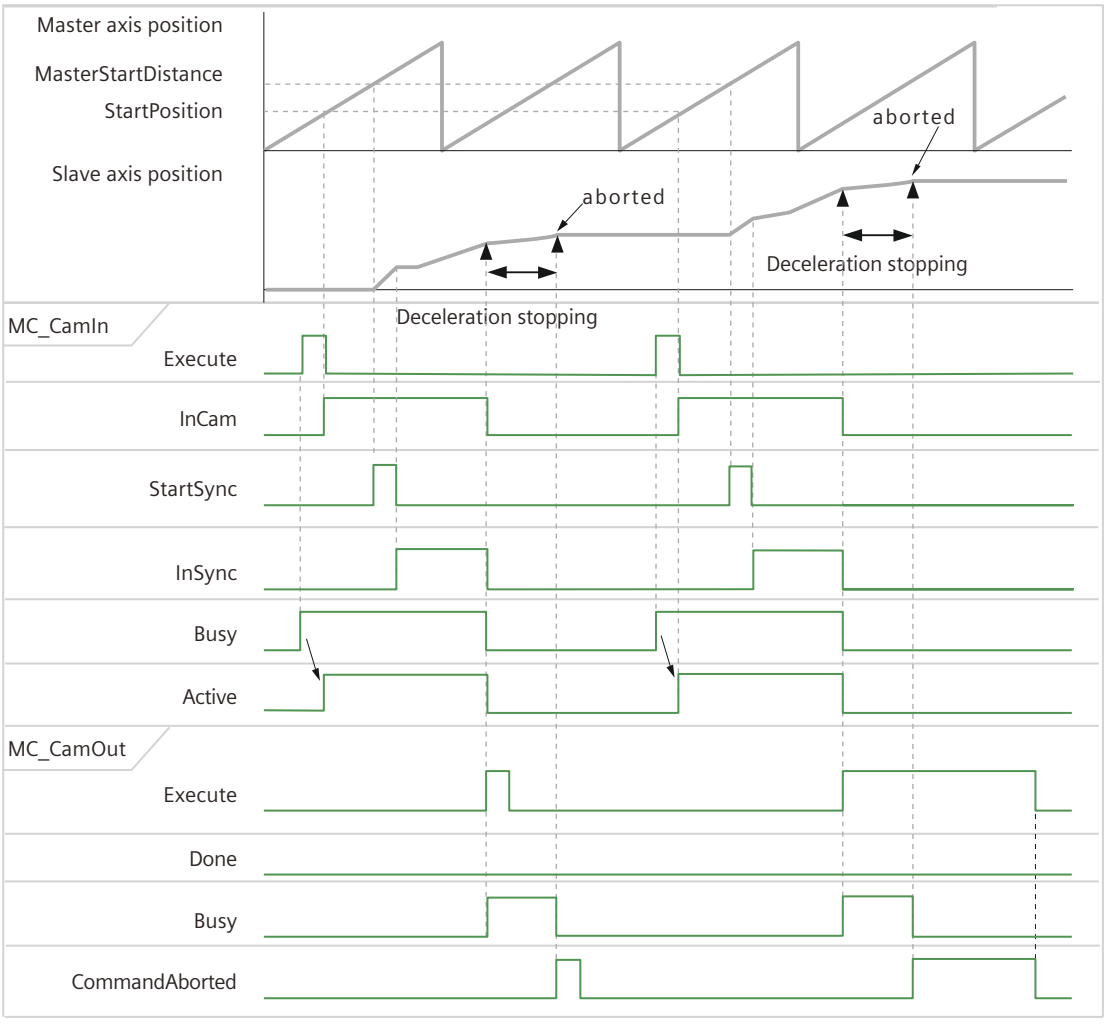
- The slave axis has been configured correctly.
- The slave axis is running the MC_CamIn instruction.

Timing diagrams

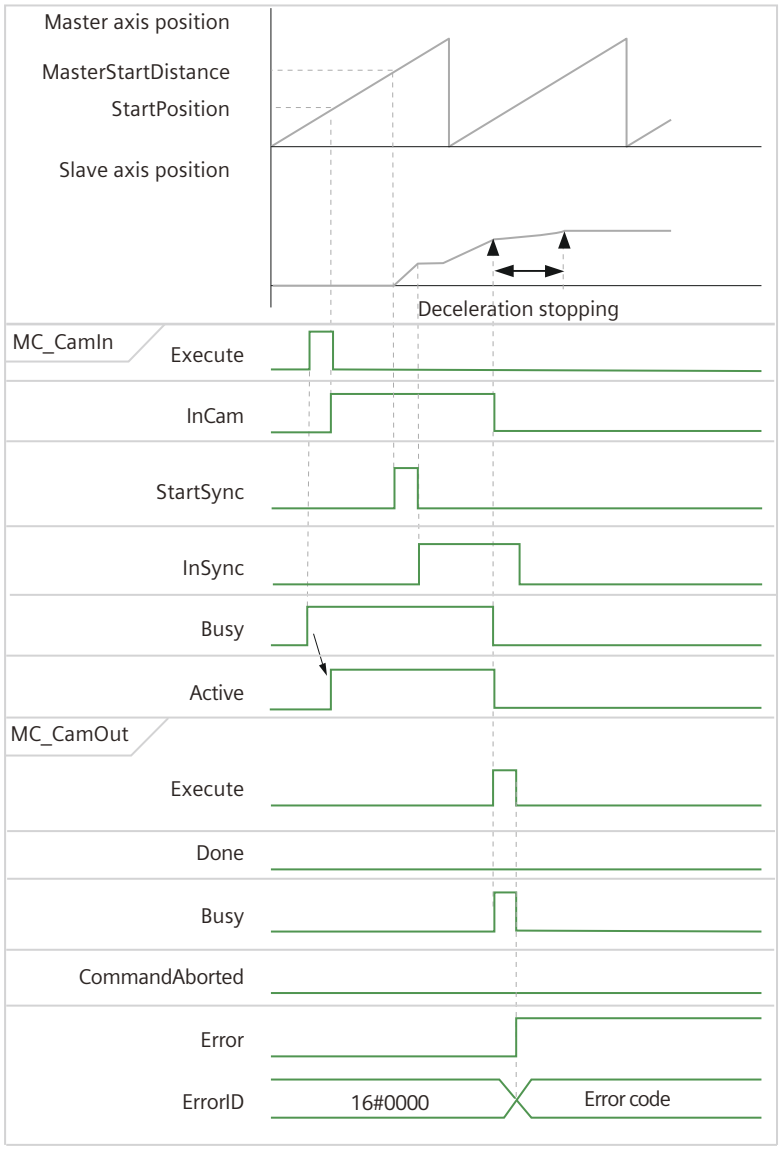
1. Rising edge trigger and Level trigger



2. Interrupted by another instruction during execution



3. An axis error occurs during motion


Master axis position
MasterStartDistance
StartPosition
Slave axis position
Deceleration stopping
MC_CamIn
Execute
InCam
StartSync
InSync
Busy
Active
MC_CamOut
Execute
Done
Busy
CommandAborted
Error
ErrorID
16#0000
Error code

Usage Notes

- Instruction Abort Behavior

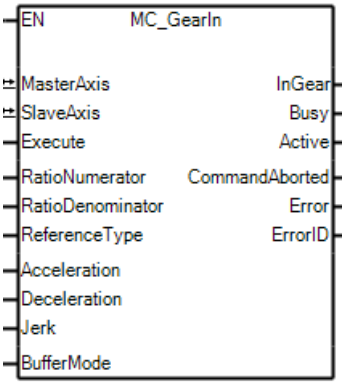
Current instruction	Subsequent instructions
MC_CamOut	MC_Halt MC_Stop MC_MoveJog MC_ImmediateStop MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn
MC_CamIn	MC_CamOut

- Instruction Buffering Behavior
  CamOut cannot be buffered behind any other instructions. No other instructions can be buffered after CamOut.

14.7.2.3 MC_GearIn

MC_GearIn function description

MC_GearIn used to activate electronic gear transmission; by setting a gear ratio between the master and slave axes, the slave axis can synchronously follow the master's motion based on the specified ratio.

LAD/FBD	STL
	`FBCALL MC_GearIn, INSTANCE, MasterAxis,SlaveAxis, Execute, RatioNumerator, RatioDenominator, ReferenceType, Acceleration, Deceleration, Jerk, BufferMode, InGear, Busy, Active, CommandAborted, Error, ErrorID`

The following table lists the axis types supported by the master and slave axes in the MC_GearIn instruction:

Slave axis	Master axis			
	IRT	**PTO**	**Virtual**	**Encoder**
IRT	✓[1]	-	✓	✓
PTO	-[2]	✓	✓	✓
Virtual axis	✓	✓	✓	✓

[1] "✓" indicates that the axis type is supported.

[2] "-" indicates that the axis type is not supported.

Note

Since strict synchronization cannot be achieved between PTO-controlled axes and axes communicating via IRT, precise synchronization between IRT and PTO cannot be guaranteed even through gear cascading via a virtual axis (e.g., IRT axis → Virtual → PTO axis).

Note

During synchronous motion, the Setting_Info.Jerk variable of the slave axis always displays as 0.

Parameter

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	-	The instruction is triggered on the 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

Parameter	Declaration	Data type	Initial value	Description
Reference-Type	IN	BYTE	0	When performing slave axis synchronization, you can select from the following master axis speed types: • 0: Instruction speed (the master axis set-point speed within the same task cycle) • 1: Feedback speed (the master axis feedback speed acquired within the same task cycle) • 2: Instruction speed (set-point speed of the previous task cycle) The supported speed types for different types of master axes are as follows: • IRT axis: Supports 0, 1, 2 • Virtual Axis: Supports 0, 1, 2 When the master axis is a virtual axis, setting it to 1 is equivalent to setting it to 0. • Encoder Axis: 1 • PTO Axis: 0
Acceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Acc	Acceleration value
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	Deceleration value
Jerk	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	Value of 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 current instruction is aborted by another instruction
Error	OUT	BOOL	False	An error has occurred with the instruction
ErrorID	OUT	WORD	0	Error code (Page 1020)

Note

The slave axis number must be greater than the master axis number.

⚠ WARNING

Notes on using MC_GearIn

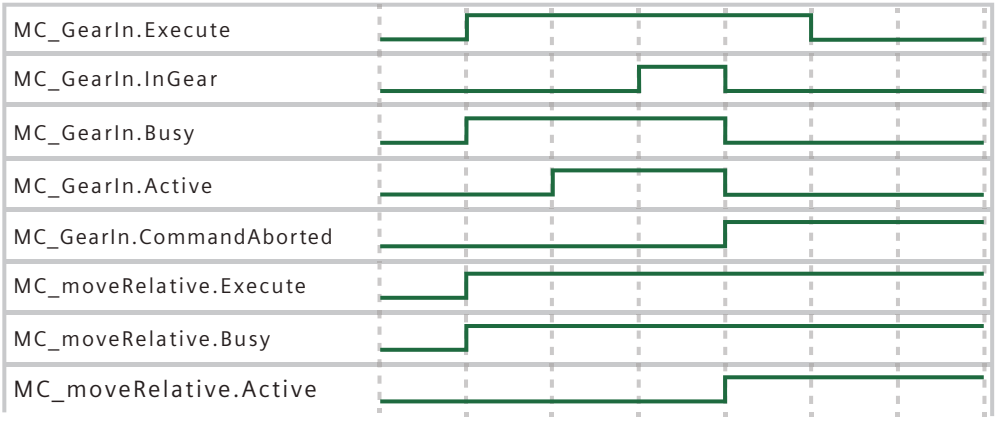
When using MC_GearIn, avoid executing MC_Home or MC_ImmediateStop instructions on the master axis. Executing these instructions may cause the slave axis to move at high speeds (or trigger axis errors), which can be extremely hazardous.

Preconditions

- The master and slave axis have been configured correctly.
- The slave axis is enabled.

When the MC_GearIn instruction is executed in buffered mode

When the MC_GearIn instruction is executed in buffered mode, its InGear and CommandAborted signals will set to TRUE simultaneously with the Active signal of the buffered instruction within the same task cycle.



NOTICE: If MC_GearIn is not called within OB2, the timing sequence described above cannot be strictly guaranteed.

Override response

- The MC_GearIn instruction can be aborted or buffered by the following motion control instructions:

Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_Halt MC_Stop MC_ImmediateStop MC_MoveVelocity MC_MoveAbsolute MC_MoveRelative MC_MoveJog MC_CamIn MC_GearIn MC_GearOut	MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn

- A new MC_GearIn instruction can abort or buffer the following motion control instructions:

Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveJog MC_CamOut MC_GearOut MC_MoveVelocity MC_CamIn MC_GearIn MC_PhasingRelative	MC_Halt MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn

14.7.2.4 MC_GearOut

Table 14-14 MC_GearOut

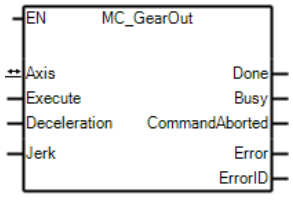
LAD/FBD	STL	Description
	`FBCALL MC_GearOut, INSTANCE, SlaveAxis, Exec, Dec, Jerk, Done, Busy, CommandAborted, Error, ErrorID`	Disable MC_GearIn gear behavior and decelerate the axis to stop

Table 14-15 Parameters for MC_GearOut

Parameter	Declaration	Data type	Initial value	Description
EN	IN	BOOL	-	Power flow
Aixs	IN_OUT	SYS_AXIS	-	Name of the axis
Execute	IN	BOOL	-	The instruction is triggered on the rising edge.
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	Deceleration value

Parameter	Declaration	Data type	Initial value	Description
Jerk	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	Value of jerk
Done	OUT	BOOL	False	IRT axis: While in the Standstill state: PTO axis, virtual axis: When the axis velocity is 0
Busy	OUT	BOOL	False	The instruction is running
CommandAborted	OUT	BOOL	False	The instruction is aborted by another instruction.
Error	OUT	BOOL	False	An error has occurred with the instruction
ErrorID	OUT	WORD	0	Error code (Page 1020)

Note

Jerk parameter effectiveness rules

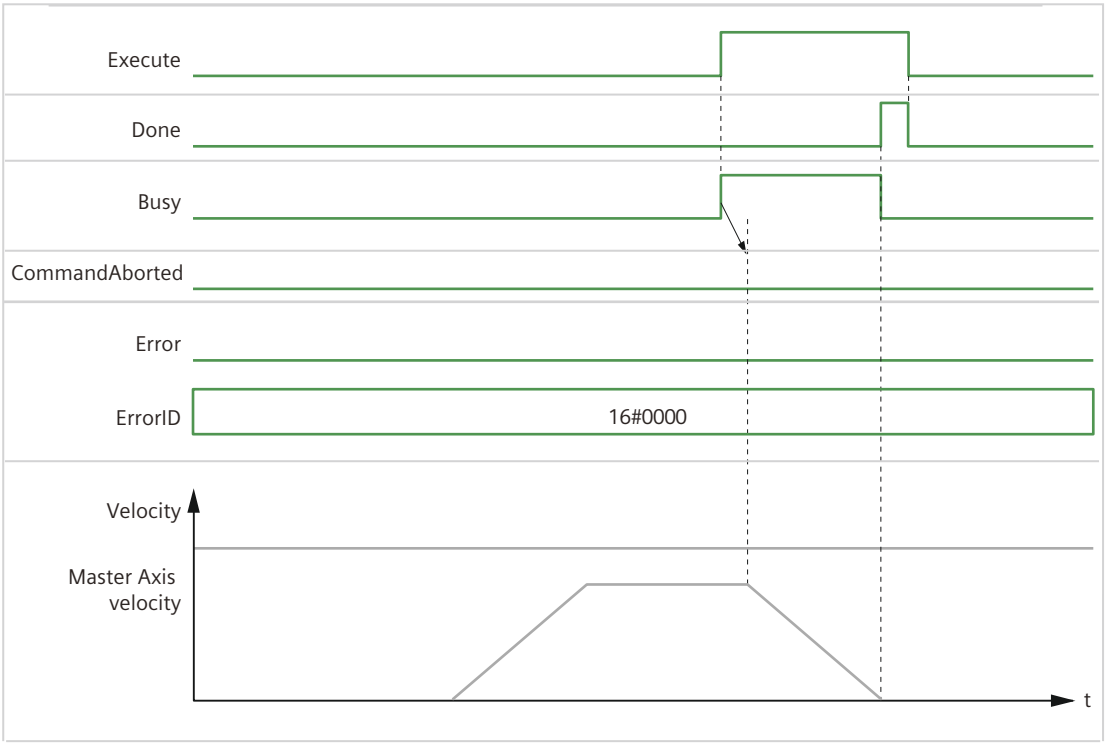
Starting from STEP 7 Micro/WIN SMART V3 (Version 3.2), the Jerk parameter for the GearOut function will take effect and is applicable to all axis types.

Preconditions

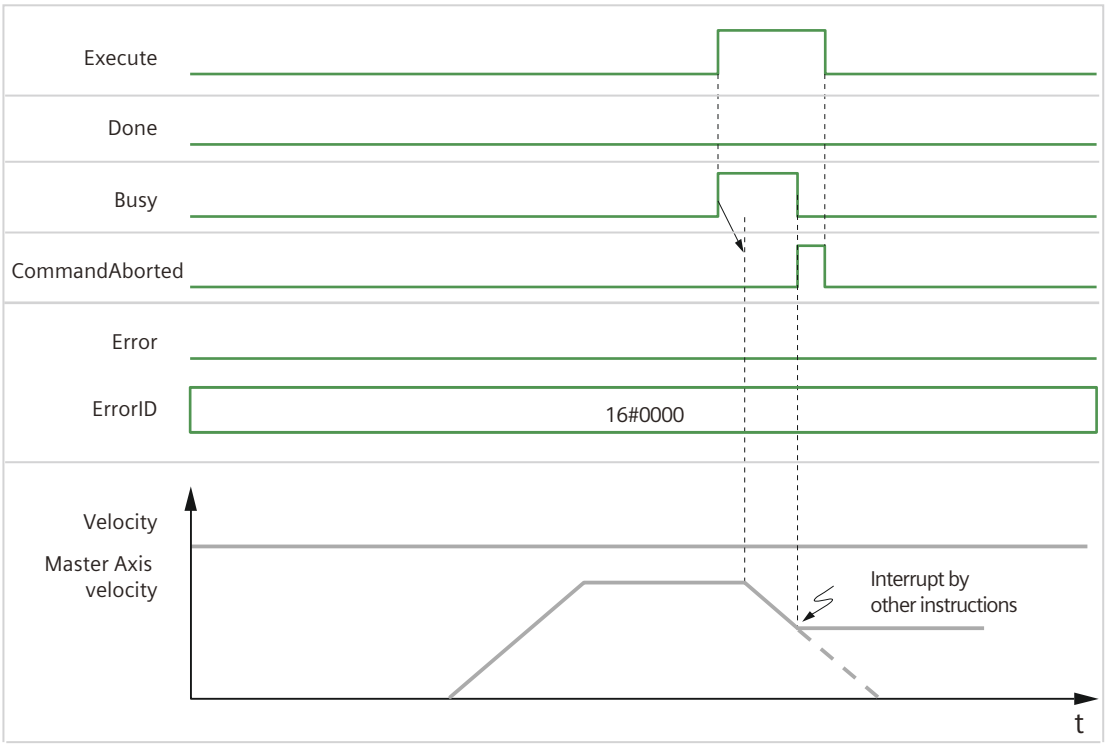
- The slave axis has been configured correctly.
- The axis is currently executing the MC_GearIn instruction.

Timing diagrams

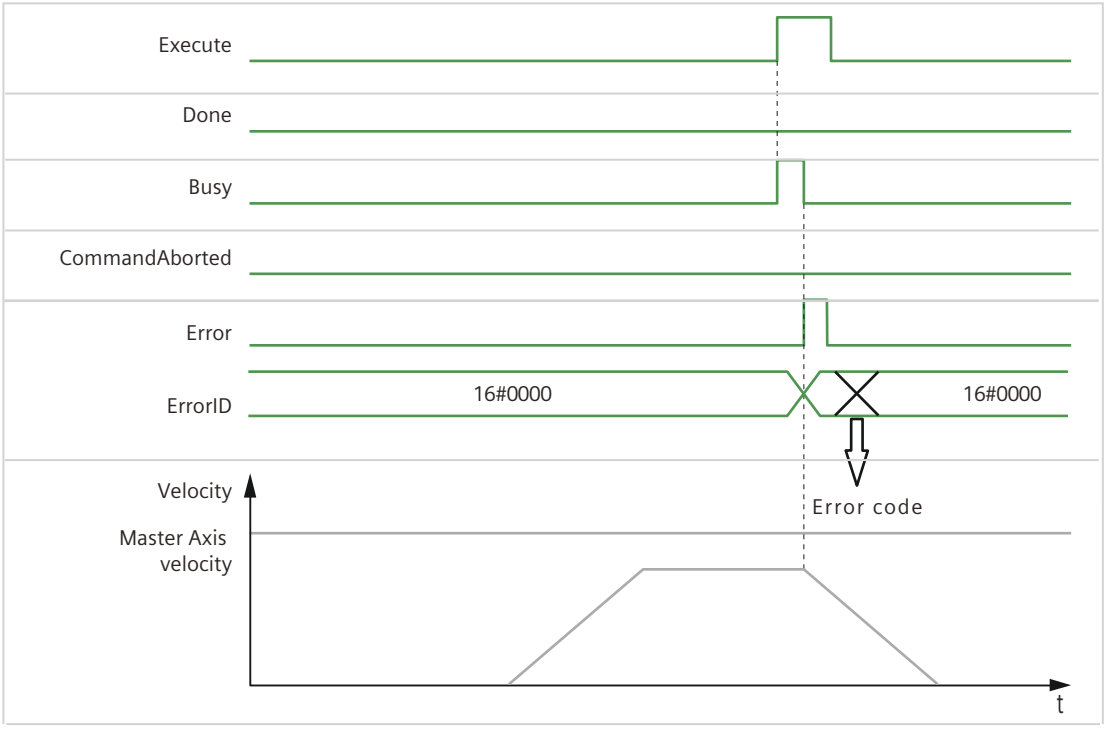
1. Triggered by a rising edge



2. Interrupted by another instruction during execution



3. An axis error occurs during motion



Override response

- Instruction Abort Behavior

Current instruction	Subsequent instructions
MC_GearOut	MC_Halt MC_Stop MC_MoveJog MC_ImmediateStop MC_MoveAbsolute MC_MoveRelative MC_MoveVelocity MC_CamIn MC_GearIn
MC_GearIn	MC_GearOut

- Instruction Buffering Behavior
  GearOut cannot be buffered behind any other instructions. No other instructions can be buffered after GearOut.

Note

When an MC_GearOut instruction aborts an MC_GearIn instruction, any MC_GearIn instructions currently in the buffer queue cannot be aborted.

Note

The MC_GearOut instruction does not support restart behavior.

14.7.2.5 MC_PhasingRelative

The MC_PhasingRelative instruction is used during synchronous motion (such as MC_CamIn or MC_GearIn) to compensate for the current phase of the slave axis relative to the master axis. With this instruction, you can specify a relative phase offset to adjust the synchronous position of the slave axis without interrupting the current synchronous operation.

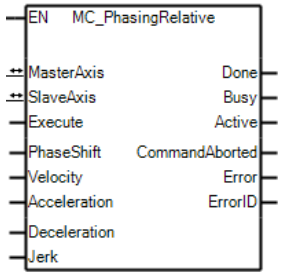
Note

The master and slave axis must be running the MC_GearIn or MC_CamIn.

Note

During synchronous motion, the Setting_Info.Jerk variable of the slave axis always displays as 0.

Parameter

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.

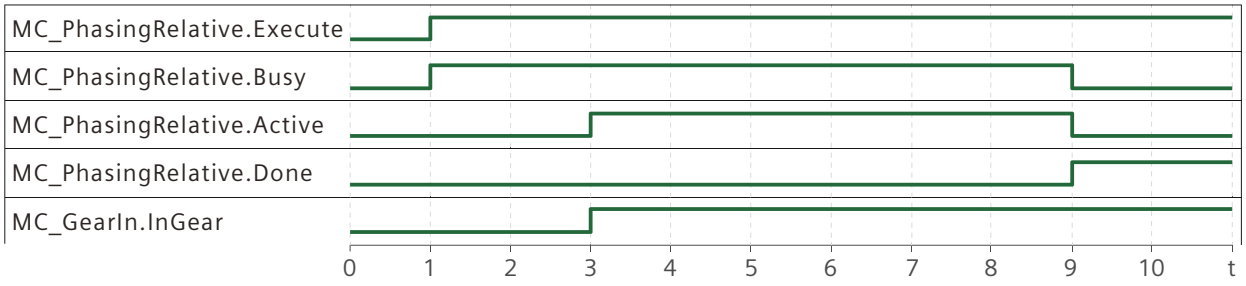
Parameter	Declaration	Data type	Initial value	Description
EN	IN	BOOL	-	Power flow
MasterAxis	IN_OUT	SYS_AXIS	-	Reference to a master axis
SlaveAxis	IN_OUT	SYS_AXIS	-	Reference to a slave axis
Execute	IN	BOOL	-	Start the instruction at rising edge
PhaseShift	IN	REAL	-	Phase compensation Positive compensation when > 0.0 Negative compensation when < 0.0
Velocity	IN	REAL	-	Specify the phase shift velocity The value range: > 0.0
Acceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Acc	Specify the acceleration rate The value range: > 0.0
Deceleration	IN	REAL	Axis_Config. Dynamic_Default. Default_Decel	Specify the deceleration rate The value range: > 0.0
Jerk	IN	REAL	Axis_Config. Dynamic_Default. Default_Jerk	Specify the jerk The value range: >= 0.0
Done	OUT	BOOL	False	TRUE when the instruction is completed Set when the compensated phase value of the master axis is equal to the PhasingShift configured in the FB (Function Block)
Busy	OUT	BOOL	False	The instruction is not finished
Active	OUT	BOOL	False	The instruction is currently controlling the axis
CommandAborted	OUT	BOOL	False	Instruction is aborted
Error	OUT	BOOL	False	An error has occurred with the instruction
ErrorID	OUT	WORD	0	Error code (Page 1020)

Preconditions

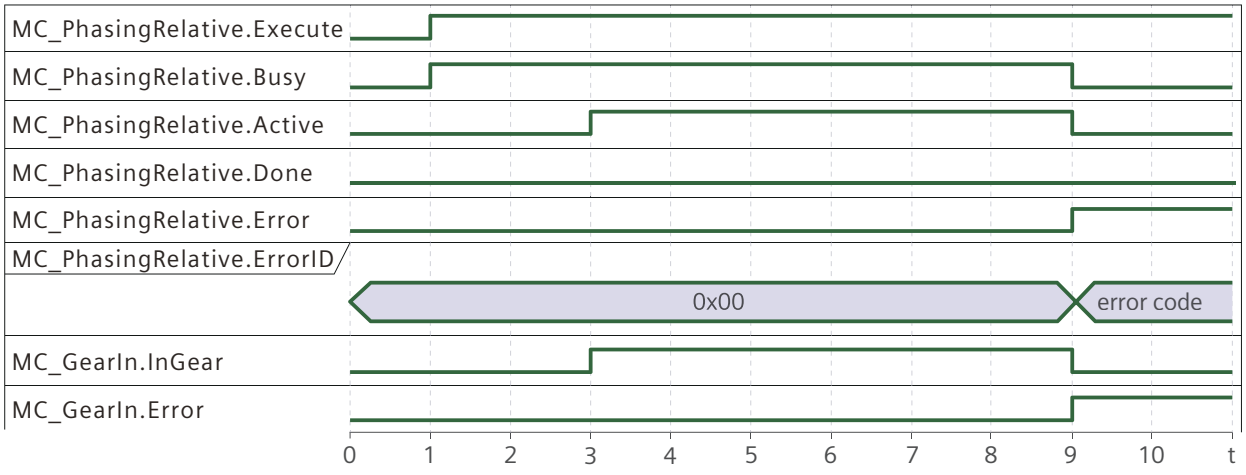
- Master and slave axes are correctly configured.
- Master and slave axes are enabled.
- The master and slave axes must be running the MC_GearIn or MC_CamIn instruction, and there must be a one-to-one correspondence between them.

Timing diagrams

- Normal Operating Scenarios



- Scenarios of Operation Failure Due to Errors



Override response

MC_PhasingRelative can only be executed by superimposing it onto the corresponding motion while CamIn or GearIn is active.

14.7.2.6 MC_GenerateCamTable

Table 14-16 MC_GenerateCamTable

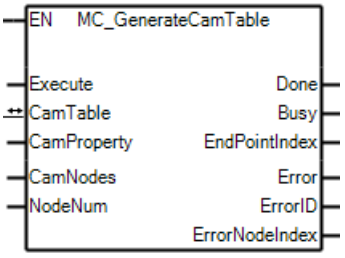
LAD/FBD	STL	Description
	**FBCALL MC_GenerateCamTable, INSTANCE, Exec, CamTable, CamProperty, CamNodes, NodeNum, Done, Busy, EndPointIndex, Error, ErrorID, ErrorNodeIndex**	The MC_GenerateCamTable instruction can be used to modifiy the cam table.

Table 14-17 Parameters for MC_GenerateCamTable

Parameters	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 cases
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 data in the `Cam_Data` 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 (Page 1020) 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 greater than or equal to NodeNum.

Data structure

Table 14-18 SYS_CAM_PROPERTY

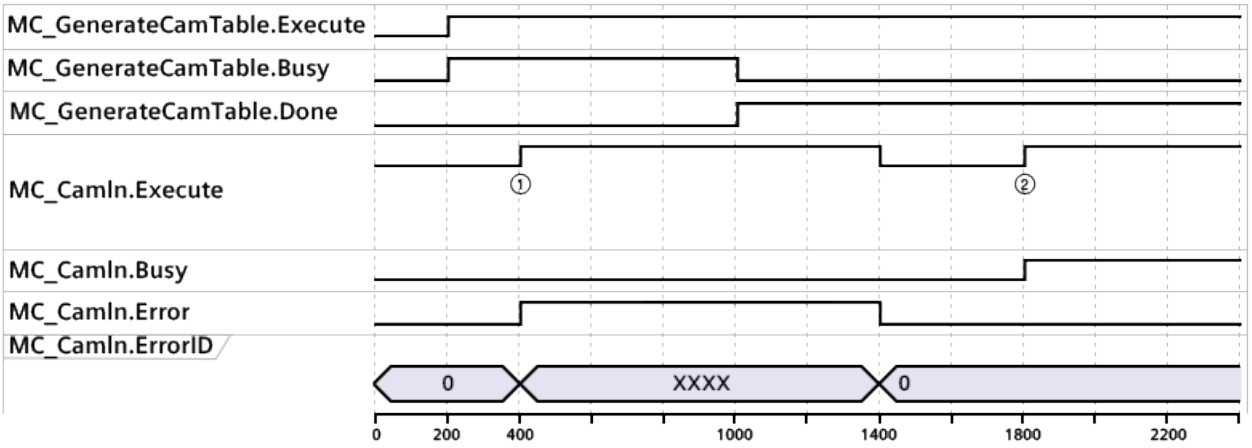
	Type	Whether to participate in calculation?
m_unit	BYTE	N
s_unit	BYTE	N
period	REAL	Y
init_vel	REAL	Y
init_acc	REAL	Y
limit_vel	REAL	N
limit_acc	REAL	N
limit_jerk	REAL	N
empty_datanum	WORD	N

Table 14-19 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

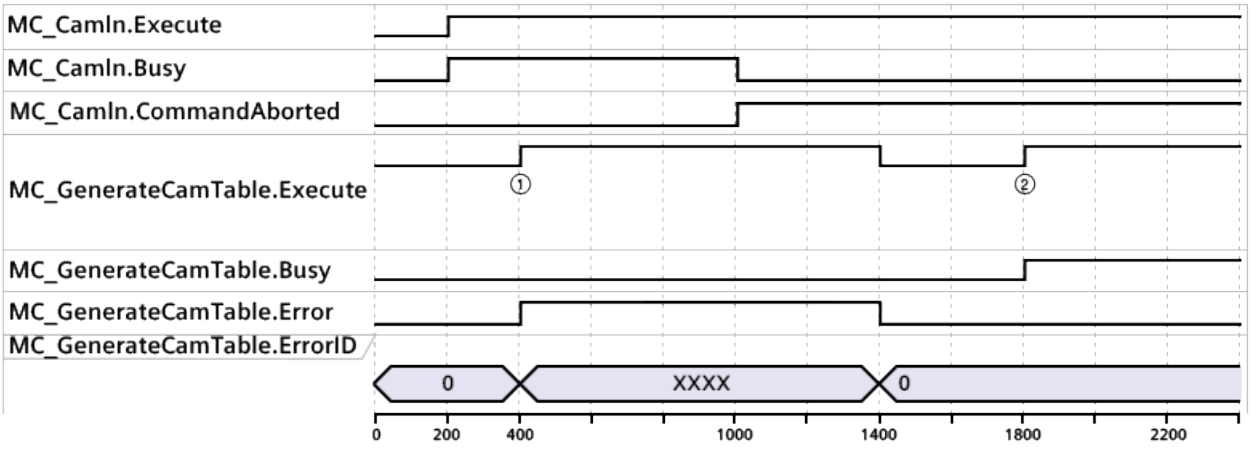
MC_GenerateCamTable and MC_CamIn

- You cannot execute the MC_CamIn instruction while executing MC_GenerateCamTable.



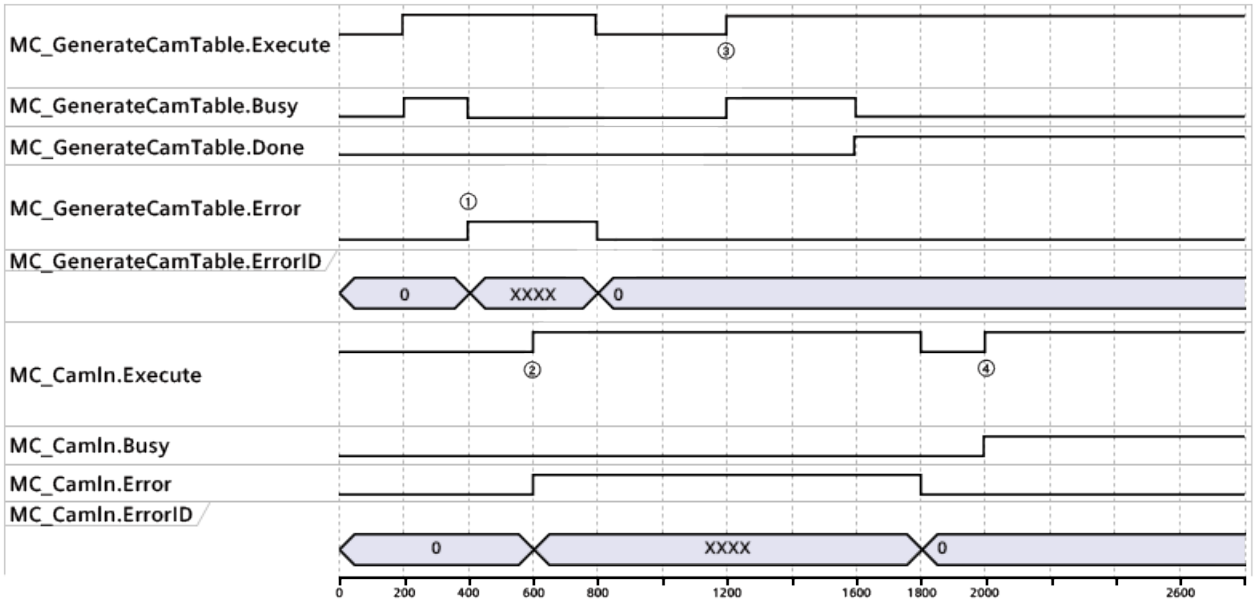
① Error occurs when executing MC_CamIn during busy state.
② MC_CamIn can be executed after MC_GenerateCamTable execution is complete.

- 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.

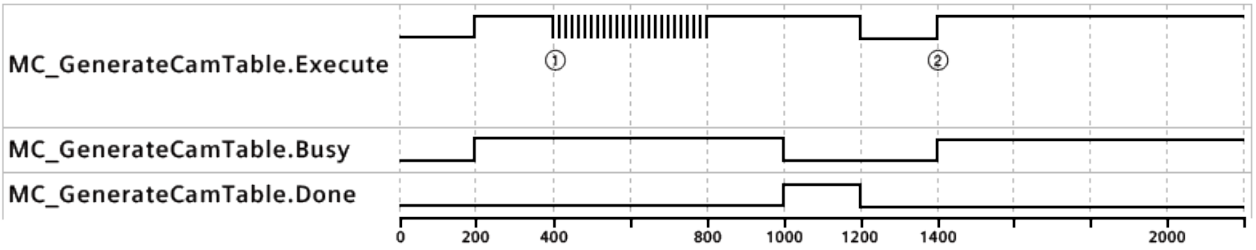


① Error occurs when executing MC_GenerateCamTable.
② Error occurs when executing MC_CamIn during error state.
③ Handle the error and re-execute MC_GenerateCamTable.
④ MC_CamIn can be executed after MC_GenerateCamTable execution is complete.

Using MC_GenerateCamTable

Multiple MC_GenerateCamTable instructions cannot be executed simultaneously for the same cam table specified by the cam table pin.

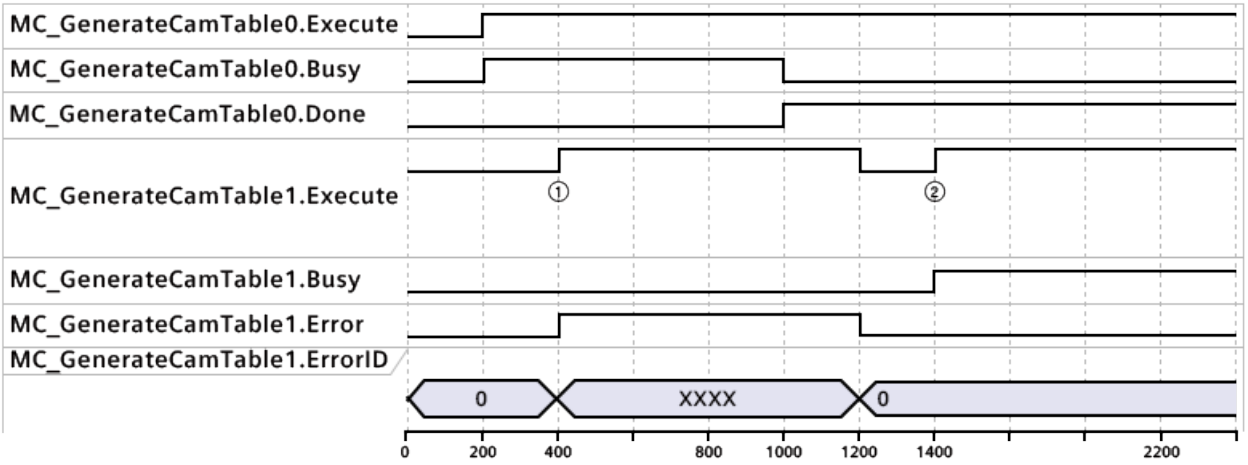
- The same FB instance



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.

14.7.3 Axes Group instructions

14.7.3.1 MC_GroupEnable

Table 14-20 MC_GroupEnable

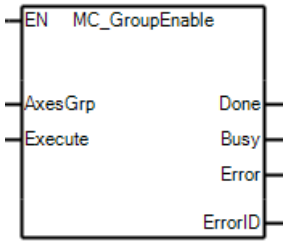
LAD / FBD	STL	Description
	**FBCALL MC_GroupEnable, INSTANCE, AxesGrp, Execute, Done, Busy, Error, ErrorID**	The MC_GroupEnable instruction changes the axes group's state from GroupDisabled to GroupStandby. It is an administrative instruction that does not generate any movement.

Table 14-21 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 (Page 1021)

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.

14.7.3.2 MC_GroupDisable

Table 14-22 MC_GroupDisable

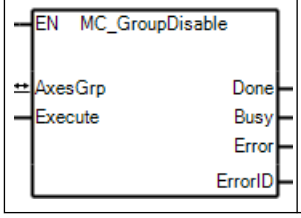
LAD/FBD	STL	Description
	**FBCALL MC_GroupDisable, INSTANCE, AxesGrp, Execute, Done, Busy, Error, ErrorID**	The MC_GroupDisable instruction changes the axes group state to GroupDisabled. It's an administrative instruction as no movement is generated.

Table 14-23 Parameters for MC_GroupDisable

Parameters	Declaration	Data type	Initial value	Description
AxesGrp	IN_OUT	SYS_AXES_GROUP	-	Reference to an axis
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 occurred during the instruction execution
ErrorID	OUT	WORD	0	Error code (Page 1021)

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.

14.7.3.3 MC_GroupReset

Table 14-24 MC_GroupReset

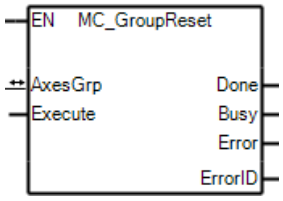
LAD/FBD	STL	Description
	**FBCALL MC_GroupReset, INSTANCE, AxesGrp, Execute, Done, Busy, Error, ErrorID**	The MC_GroupReset instruction transitions the status from GroupErrorStop to GroupStandby by resetting all internal axis group-related errors. This instruction does not affect the outputs of any other instruction instances.

Table 14-25 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	-	The instruction is triggered on the rising edge.
Done	OUT	BOOL	False	Resetting of all axes in the axes group completed
Busy	OUT	BOOL	False	The instruction is running
Error	OUT	BOOL	False	An error has occurred with the instruction
ErrorID	OUT	WORD	0	Error code (Page 1021)

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.

14.7.3.4 MC_GroupStop

Table 14-26 MC_GroupStop

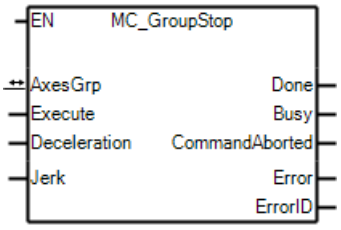
LAD/FBD	STL	Description
	**FBCALL MC_GroupStop, INSTANCE, AxesGrp, Execute, Deceleration, Jerk, Done, Busy, Active, CommandAborted, Error, ErrorID**	The MC_GroupStop instruction commands the axis group to decelerate and stop, and transfers the axes group to the state GroupStopping. The instruction aborts 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 14-27 Parameters for MC_GroupStop

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
Deceleration	IN	REAL	-	Value of the Deceleration parameter
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 occurred during the instruction execution
ErrorID	OUT	WORD	0	Error code (Page 1021)

Preconditions

- The positioning axes group has been configured correctly.
- The axes group is enabled.

Override response

- Motion control instructions that can abort the MC_GroupStop instruction:

Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_GroupDisable MC_Power	

- A new MC_GroupStop instruction can abort the following motion control instructions:

Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_MoveLinearAbsolute MC_MoveLinearRelative	

14.7.3.5 MC_MoveLinearAbsolute

Table 14-28 MC_MoveLinearAbsolute

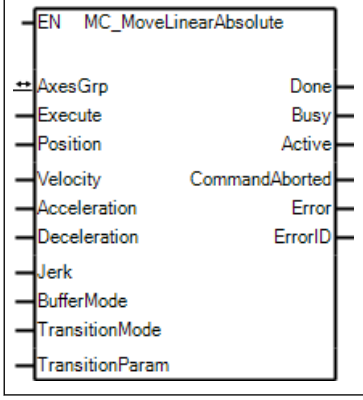
LAD/FBD	STL	Description
	**FBCALL MC_MoveLinearAbsolute , INSTANCE, AxesGrp, Execute, Position, Velocity, Acceleration, Deceleration, Jerk, BufferMode, TransitionMode, TransitionParameter, Done, Busy, Active, CommandAborted, Error, ErrorID**	The MC_MoveLinearAbsolute instruction command a linear interpolated 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 14-29 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	-	The instruction is triggered on 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
Velocity	IN	REAL	-	Target velocity (Not necessarily reached) The value range: Any real number
Acceleration	IN	REAL	-	Acceleration value, optional The value range: > 0.0
Deceleration	IN	REAL	-	Deceleration value, optional The value range: > 0.0

Parameter	Declaration	Data type	Initial value	Description
Jerk	IN	REAL	0.0	Jerk value, optional The value range: >= 0.0
BufferMode	IN	BYTE	0	• 0: Aborting mode. According to the Deceleration value of the original instruction, decelerate to zero with its original path. And then start this command. • 1: Buffered mode. Start instruction after current motion has finished
TransitionMode	IN	BYTE	0	Reserved
TransitionParam	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	The instruction is controlling the axes group
CommandAborted	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 (Page 1021)

Preconditions

- The axes group is configured correctly.
- The axes group is enabled.
- All single axis in the axes group completed the MC_Home instruction.

Override response

- Motion control instructions that can handle the MC_MoveLinearAbsolute instruction by aborting or keeping them buffered:

Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_Power MC_MoveLinearAbsolute MC_MoveLinearRelative MC_GroupDisable MC_GroupStop	MC_MoveLinearAbsolute MC_MoveLinearRelative

- A new MC_MoveLinearAbsolute instruction can handle the following motion control instructions by aborting or keeping them buffered:

Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_MoveLinearAbsolute MC_MoveLinearRelative	MC_MoveLinearAbsolute MC_MoveLinearRelative

14.7.3.6 MC_MoveLinearRelative

Table 14-30 MC_MoveLinearRelative

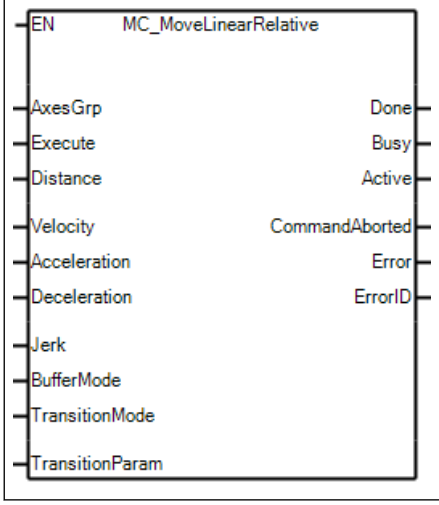
LAD/FBD	STL	Description
	**FBCALL MC_MoveLinearRelative , INSTANCE, AxesGrp, Execute, Distance, Velocity, Acceleration, Deceleration, Jerk, BufferMode, TransitionMode, TransitionParameter, Done, Busy, Active, CommandAborted, Error, ErrorID**	The MC_MoveLinearRelative instruction commands a linear interpolated 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 14-31 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	-	The instruction is triggered on 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	-	Target velocity (Not necessarily reached) The value range: Any real number
Acceleration	IN	REAL	-	Acceleration value, optional The value range: > 0.0
Deceleration	IN	REAL	-	Deceleration value, optional The value range: > 0.0
Jerk	IN	REAL	0.0	Jerk value, optional The value range: >= 0.0
BufferMode	IN	BYTE	0	• 0: Aborting mode. According to the Deceleration value of the original instruction, decelerate to zero with its original path. And then start this instruction. • 1: Buffered mode. Start instruction after current motion has finished
Transition-Mode	IN	BYTE	0	0
TransitionPara	IN	DWORD	-	Reserved

Parameter	Declaration	Data type	Initial value	Description
Done	OUT	BOOL	False	All axes has reached the commanded end position
Busy	OUT	BOOL	False	The instruction is running
Active	OUT	BOOL	False	The instruction is controlling the axes group
CommandA-borted	OUT	BOOL	False	The current instruction is aborted by another instruction
Error	OUT	BOOL	False	An error has occurred with the instruction
ErrorID	OUT	WORD	0	Error code (Page 1021)

Preconditions

- The axes group is configured correctly.
- The axes group is enabled.

Override response

- Motion control instructions that can handle the MC_MoveLinearRelative instruction by aborting or keeping them buffered:

Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_Power MC_MoveLinearAbsolute MC_MoveLinearRelative MC_GroupDisable MC_GroupStop	MC_MoveLinearAbsolute MC_MoveLinearRelative

- A new MC_MoveLinearRelative instruction can handle the following motion control instructions by aborting or keeping them buffered:

Aborting (BufferMode = 0)	Buffered (BufferMode = 1)
MC_MoveLinearAbsolute MC_MoveLinearRelative	MC_MoveLinearAbsolute MC_MoveLinearRelative

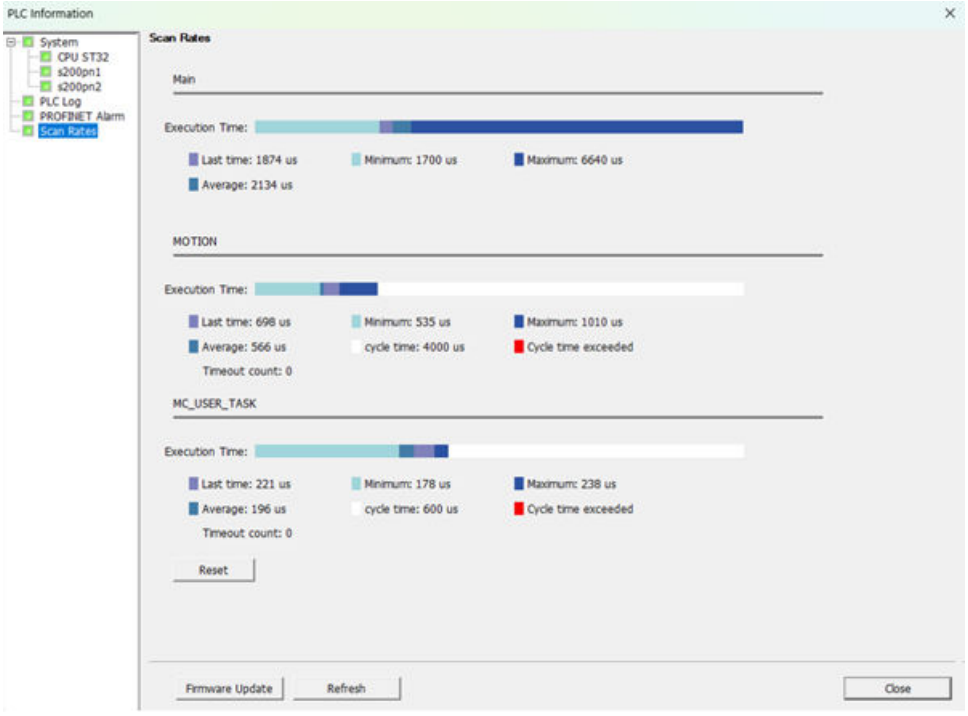
14.8 Troubleshooting and optimization for insufficient motion control resources

In complex automation systems, the stability and reliability of motion control performance are crucial. When the system runs out of motion control resources, it causes the CPU to STOP and the axis to stop. This chapter provides guidance on how to identify and resolve these problems to ensure optimum operation of your motion control system.

Identifying insufficient motion control resources

Before proceeding with troubleshooting, it is first necessary to confirm whether the current issue is actually caused by insufficient motion control resources. Here are some common diagnostic methods:

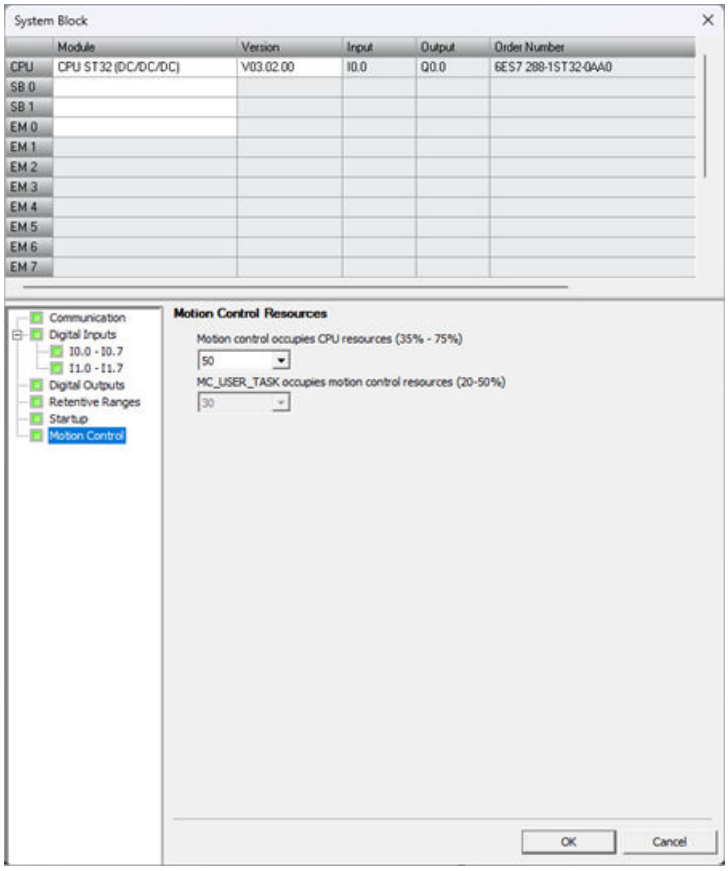
- In PLC -> System -> CPU (Page 174), check if any of the following non-fatal errors are present:
  - Insufficient Motion resources
  - MC_USER_TASK timeout
- Scan rates (Page 179): Using the scan rate, you can check the actual execution time of motion control tasks (Main, Motion, MC_USER_TASK). If the actual period is significantly longer than the configured period, or the period fluctuates significantly and the number of timeouts exceeds the limit, the task may not be completed on time, resulting in insufficient resources.



- PLC log: In the PLC log, look closely for events relating to program exit or stop. If keywords such as "Timeout", "Insufficient resources" appear in the log, this is often an indication of insufficient motion control resources. This information can help you identify the cause of the insufficient resources.
- Check the motion control resource ratio in PLC information: In the System Block -> Motion Control (Page 218), you can view the proportion of CPU resources occupied by the motion control tasks and the proportion of MC_USER_TASK resources occupied within motion control.

Solutions for insufficient motion control resources

- Increase the resource allocation for motion control tasks: In the System Block -> Motion Control (Page 218), you can adjust the system configuration to allocate more CPU time to motion control tasks. This is one of the most direct and typically most effective solutions.



- Optimize motion control programs
  - Review program code in motion control to remove unnecessary calculations, logic decisions, or data processing.
  - Avoid time-consuming operations in isochronous tasks, such as complex floating-point operations, string processing, or reading or writing large data blocks.
  - Move the non-time-critical code out of MC_USER_TASK and put it in the main scan for execution.
  - Use optimized data types and data access methods whenever possible. For example, directly access data in a data block (DB) rather than using complex pointers or indirect addressing.
  - Avoid the frequent calls of function blocks (FBs) that require large amounts of system resources in isochronous tasks.

- **Re-evaluate system requirements**
  - Re-evaluate whether all axes in the current project must perform high-speed, high-precision motion simultaneously.
  - Is it possible to serialize the motion of certain axes instead of executing them in parallel?
  - Is it possible to simplify the motion trace or control strategies for certain axes?
- **Implement in phases**
  If the project allows, consider splitting a large multi-axis system into multiple independent motion control units with fewer axes, each controlled by a separate PLC or motion controller. This will increase hardware costs and system complexity but can effectively distribute CPU load.

14.9 Error codes

14.9.1 Error code for single axis motion

Table 14-32 Error code for motion system configuration

Error code	Description
0x0001	Invalid axis type STEP 7 Micro/WIN SMART V3 supports 0-Virtual Axis, 1-PTO Axis, 2-Encoder Axis, and 3-PN IRT 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 setting for the encoder axis
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	When two input points are used, the encoder axes are not configured in the same group (I0.X or I1.X refers to the group for each axis)
0x0029	Invalid axis TO pointer element number
0x0030	Unreasonable start/stop speed configuration for the PTO axis Pulse range: [200 hz, 200k hz]
0x0031	The velocity filtering time constant of the encoder axis is unreasonable 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
0x0037	Incorrect configuration for PTO output point 0
0x0038	Incorrect configuration for PTO output point 1
0x003A	PN device ID exceeds the limit
0x003B	Incorrect configuration for the encoder type
0x003C	Incorrect configuration for the active homing mode

Table 14-33 Error code for motion axis configuration

Error code	**Description**
0x0200	TO version error
0x0201	The unit of measurement is incorrect STEP 7 Micro/WIN SMART V3 supports 0-pulse, 1-cm, 2-mm, 3-in, 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
0x0209	Encoder axis velocity filtering time constant error. The filtering time constant range is [0 ms, 2000 ms]
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]
0x0212	The Axis ID in the TO (Technology Object) issued by STEP 7 Micro/WIN SMART V3 is invalid
0x0213	The Axis Type in the TO (Technology Object) issued by STEP 7 Micro/WIN SMART V3 is invalid
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 the execution of the motion instruction; please refer to the error code in the corresponding instruction
0x0300	The drive has reported an error
0x0301	SOL signal loss between the drive and CPU
0x0302	Communication loss between the drive and CPU
0x0303	Drive parameter read failure
0x0304	Drive enable response failure
0x0305	Axis following error exceeds tolerance
0x0310	Axis position monitoring error
0x0311	Axis clamping error

Table 14-34 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 axes 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 buffered instructions for TO_AXIS
0x1011	Invalid Direction parameter
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)
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.)
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)
0x1028	Cannot execute other motion instructions during deceleration to stop
0x1029	Aborting Invalid
0x1030	Triggered the activated software positive limit switch
0x1031	Triggered the activated software negative limit switch
0x1032	Error detected on TO axis
0x1033	PN IRT axis instruction execution timeout: The axis failed to successfully complete the operation within the preset tolerance time (including failure to reach the target position or zero speed) and set the "Done" bit.
0x1035	A motion instruction was executed while in Torque Control Mode
0x1036	The motion direction is opposite to the clamping direction
0x1037	Instruction does not support velocity mode
0x1038	Axis does not support velocity mode
0x1039	Invalid buffered mode
0x1040	The axis ID has been changed during the execution of the function block Axis ID restart error
0x1080	Both Jogforward and JogBackward are in rising edge
0x1FFF	Inner unexpected error code

Table 14-35 Error codes for single-axis motion control

Error code	Description
0x2000	Multi-instance calls to MC_Power are not allowed
0x2001	Axis enable failure: Drive not ready
0x2002	The axis has been pre-occupied by the control panel
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
0x2505	PN encoder abnormality
0x2506	Active homing submode error
0x2507	Drive not ready when homing instruction is issued
0x2600	MC_ImmediateStop cannot be executed while MC_Stop is in progress
0x2C00	When the Busy status of an MC_AbortTrigger or MC_TouchProbe function block instance is TRUE, if the Execute pin of that instance is called again, the system will trigger this error.
0x2C01	MC_abort trigger executed while MC_touchprobe is not running
0x2C02	The axis is performing a homing operation when MC_abort trigger or MC_touchprobe is executed
0x2C03	Encoder error (for PN IRT axis only)
0x2C04	Drive does not support probe function (for PN IRT axis only)
0x1090	When using the Control Panel, if the connection between the PLC and STEP 7-Micro/Win SMART V3 is interrupted.
0x1091	Execute function block when the control panel takes over control

Table 14-36 Torque control error codes

Error code	Description
0x5200	Axis configuration does not include Telegram 750
0x5202	Repeated execution of torque control instruction
0x5203	Torque control instruction executed on non-PN IRT axis
0x5204	Invalid Torque parameter
0x5205	Invalid Ramp parameter
0x5206	Torque control instruction with mode=1 executed while the axis is performing motion
0x5207	Invalid mode parameter
0x5300	Using MC_TorqueLimiting instruction requires configuration of Telegram 105
0x5301	Axis status is Disabled in fixed stop mode
0x5302	Invalid parameters
0x5303	Abnormal exit from InClamping state, for example, fixed stop removed or pushed back
0x5304	Torque Limiting function block conflicts with other torque control function blocks (such as MC_TorqueControl or MC_TorqueRange)

Error code	Description
0x5305	InClamping state: [Enable Pin] must not be disabled
0x5400	Invalid UpperLimit/LowerLimit parameter
0x5500	Drive not ready

Table 14-37 Control panel-related error codes

Error code	Description
0x8104	Compatibility error codes for legacy PTO control panel, for example: sps7_COMMAND_NOT_SUPPORTED_ERROR
0x9000	CPU not running, control panel cannot be used
0x9001	Current function block is in control, control panel cannot be used
0x9002	TO axis is in error state, control panel cannot be used
0x9003	Control panel instruction invalid
0x9004	Current axis is not in Standstill state
0x9005	This axis does not support the command

14.9.2 Error codes for electronic cam and gear

Table 14-38 Error codes for electronic cam and gear

Error code	Description
0x0400	Invalid Cam Table ID
0x0401	Task type error
0x0402	Task deleted due to position limit exceeded
0x4000	The master axis ID of the electronic gear is the same as the slave axis ID
0x4001	The master axis ID of the electronic gear must be less than the slave axis ID
0x4002	Setpoint becomes invalid after Power Off on PN IRT master axis
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
0x4008	Invalid selection of master axis speed type
0x4100	The instruction currently being executed is not GearIn
0x4200	The master axis ID of the electronic cam is the same as the slave axis ID
0x4201	The master axis ID of the electronic cam must be less 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
0x4213	Cam table data error
0x4300	CamOut executed without prior CamIn instruction
0x4301	Invalid cam master axis reference value
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 out of range, phase value cannot be INF or NAN
0x4403	Unexpected error occurred
0x4500	Cam table interpolation calculation resource occupied
0x4501	Cam table is being used by MC_CamIn
0x4502	Cam table key point data is empty
0x4503	Cam table key point data address error
0x4504	Cam table data object address error
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
0x4513	Cam table interpolation data address is empty
0x4514	Cam table interpolation data address error
0x4515	Cam table interpolation interrupted by CPU Stop
0x4517	An internal error occurred during GenerateCamTable instruction execution

14.9.3 Error code for axes group

Error code	Description
0x0100	Invalid axis group configuration byte
0x0101	Invalid axis group dimension
0x0102	Invalid X-axis ID

0x0103	Invalid Y-axis ID
0x0104	Invalid Z-axis ID
0x0105	Invalid auxiliary axis ID
0x0106	Invalid axis group TO pointer area
0x0107	Invalid axis group TO pointer type
0x0108	Invalid number of axis group TO pointer elements
0x0109	Invalid axis group ID
0x010A	Duplicate axis group SDB record ID
0x010B	Invalid axis group SDB record length
0x010C	Unknown version
0x010D	X-axis ID already occupied
0x010E	Y-axis ID already occupied
0x010F	Z-axis ID already occupied
0x0110	Auxiliary axis ID already occupied
0x0111	Axis group TO address pointer check error
0x0112	Number of axis groups exceeds limit (maximum of two axis groups supported)
0x0381	Axes group is stopped due to an axis error
0x1026	Axis group does not exist
0x1027	Axis group has an error and is not in ErrorStop state
0x5100	Axis group instruction execution failed due to an axis error
0x5101	Single axis enabled simultaneously in multiple axis groups
0x5200	Axis configuration does not include Telegram 750

Error code for single axis motion (Page 1015)

14.10 Example programs

14.10.1 Motion control example

Use of the device

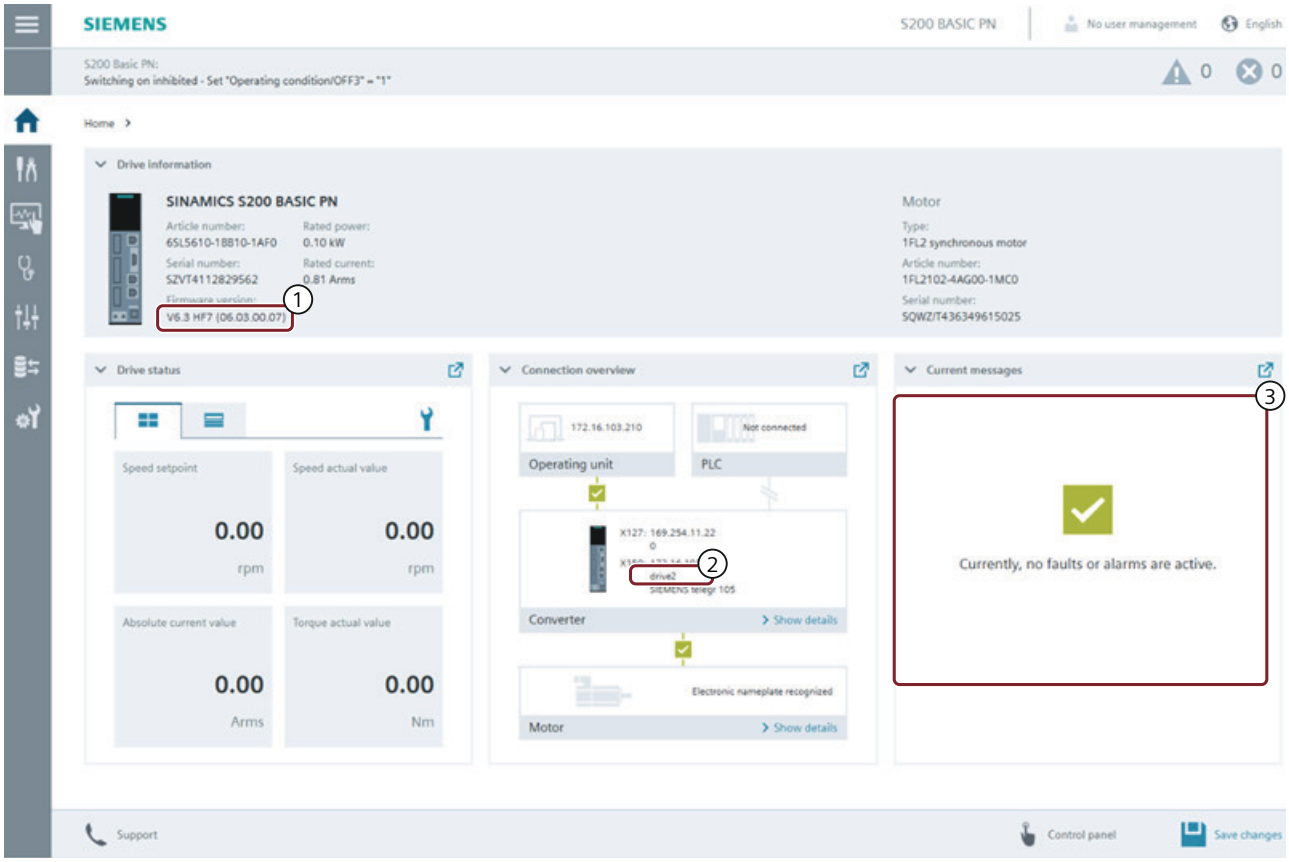
- CPU ST32
- Servo drive S200

Device connections

Connect an Ethernet cable from port P2 of the CPU ST32 to port P1 of the S200 drive.

Checking the S200 drive

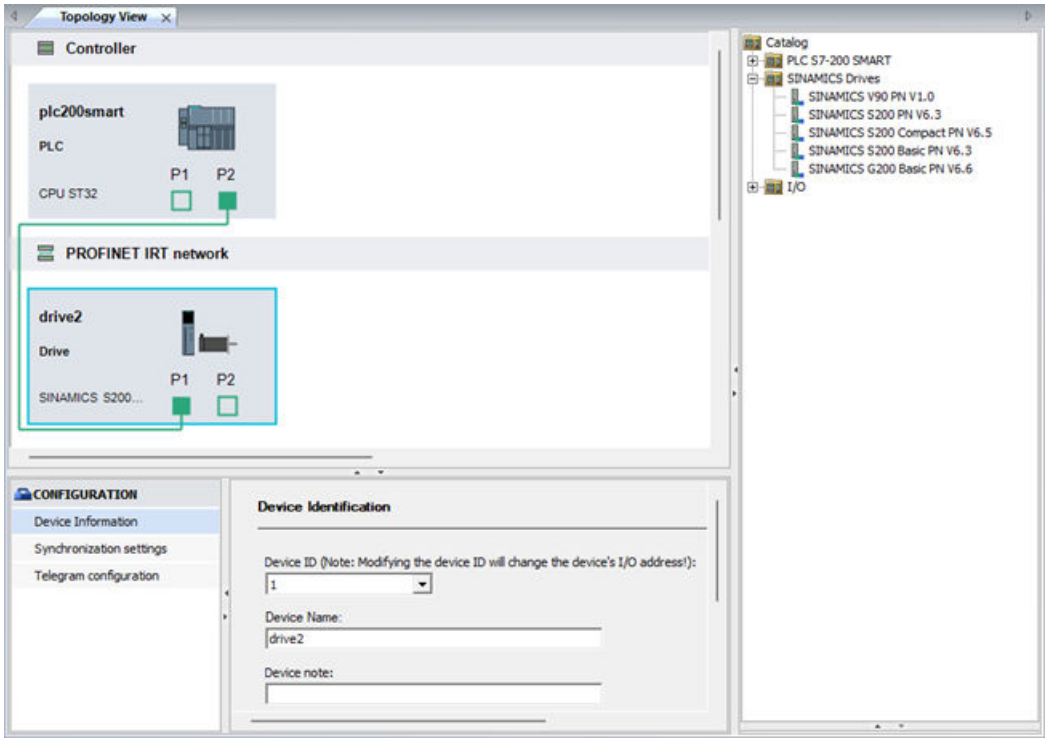
1. Connect your computer to the S200 drive using an Ethernet cable.
2. Open your browser and enter the IP address of your S200 drive.
   Consult the S200 drive manual for S200 drive default information:
   - S200 IP: 192.168.2.1
   - S200 user name: Administrator
   - Password: sinamicsS200
3. On the Web server of your S200 drive, verify the following:
   - Version of the firmware: In this example, it is V6.3
   - Asset name: In this example, it is driver2
   - Check the alarms: Ensure that the S200 drive does not display any alarm or error information.



Configuring a PROFINET network

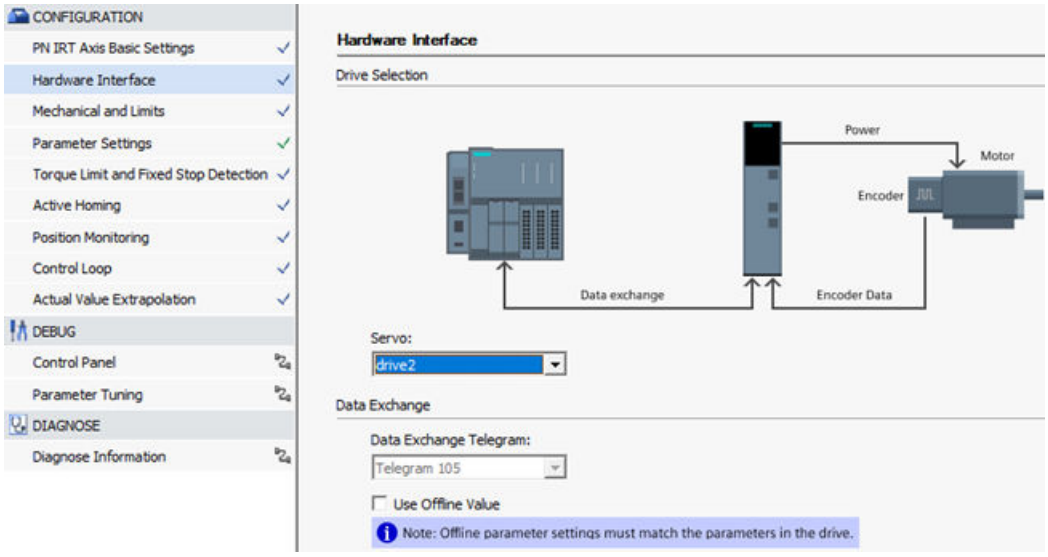
1. Open STEP 7-Micro/WIN SMART V3.
2. Double-click "Hardware network topology (Page 791)" in the project tree to open the hardware network topology.

3. In the right-hand device catalog "SINAMICS Drives", select the S200 PN drive with firmware version V6.3, "SINAMICS S200 PN V6.3", and drag and drop it into the PROFINET IRT network.
4. Select the controller and set its station name and IP address in the configuration window.
5. Select the S200 drive you just added and set its device name, IP address, and telegram in the configuration window.
   - The device name must match the name in the S200 Webserver, driver2 in this example
   - Make sure that the IP address of the controller and the S200 drive are in the same IP subnet
   - This example configuration uses telegram 105



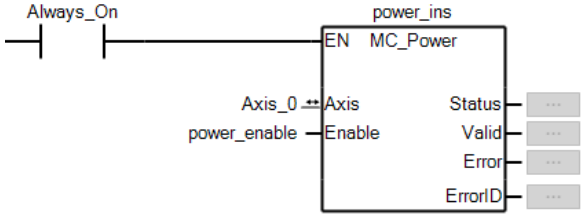
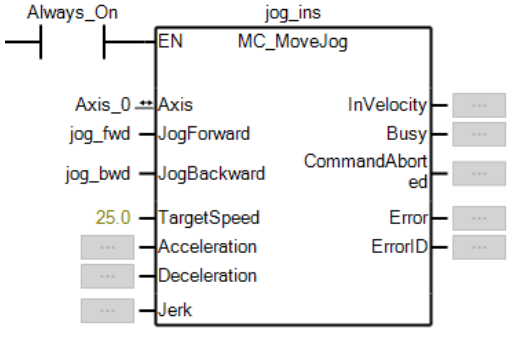
Configure the axis

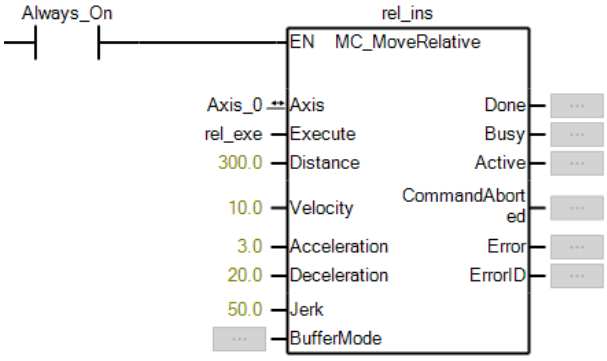
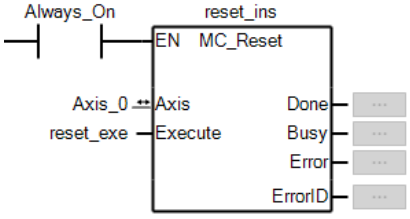
1. In the project tree, right-click on "Motion control" and select "Add" > "Axis" > "PN IRT axis".
2. In the PN IRT axis configuration interface (Page 842), click on "Hardware interface".
3. Assign the previously configured S200 drive (via its device name) to this axis.



Writing a motion control program

1. In the project tree, open the "Program block" folder. You can right-click on a POU to open it, edit its properties, protect it with a password, or rename it.
2. Add the following motion control instructions

LAD/FBD	Description
Network 1: 	For enabling the axis
Network 2: 	Axis_0, jog function

LAD/FBD	Description
Network 3: 	Axis_0, for relative displacement control
Network 4: 	For resetting Axis_0

Compiling programs

Click the "Compile" button on the toolbar to ensure there are no compilation errors.

If compilation errors occur, troubleshoot and correct them based on the error information.

Downloading the Program

1. Search for and log in to the PLC.
2. Click the "Download" button on the toolbar to download the compiled program to the PLC.

Executing motion instructions

WARNING

Safety operation

Before performing any motion control programs, it is important to verify that all safety protective measures are in place, functioning properly, and that there are no personnel or obstacles in the area of operation. Improper operation may cause equipment damage or personal injury.

1. In online mode, trigger MC_Power to enable the axis.
2. Trigger instructions such as MC_MoveJog and MC_MoveRelative, and observe the motion of the axis.

Checking the POU program status

Click the "Program status (Page 733)" button on the toolbar and monitor the status of the program blocks and axes.

Drive parameter optimization

In the configuration interface of the PN IRT axis, you can further optimize the parameters of the S200 drive on the "Parameter optimization (Page 875)" page.

Tracking axis motion traces

Open the "Trace (Page 184)" function, configure the trace to record the actual position, velocity, and other parameters of the axis for motion trace analysis.

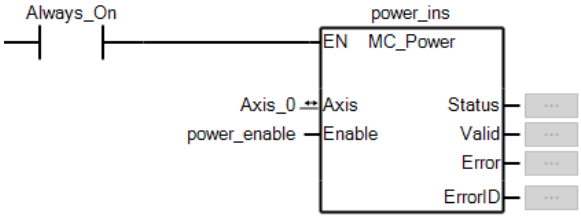
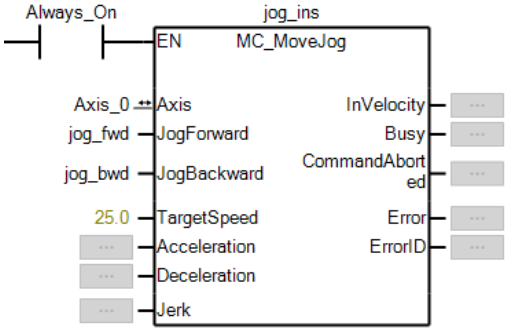
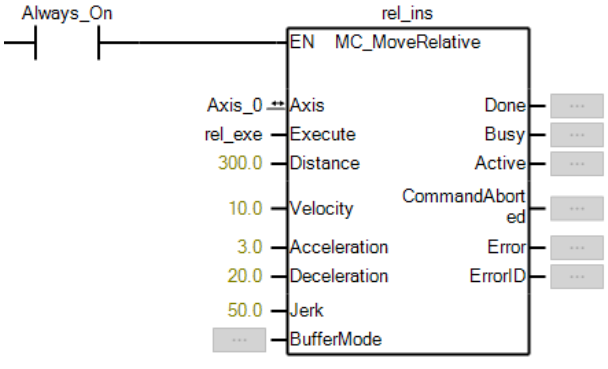
See also

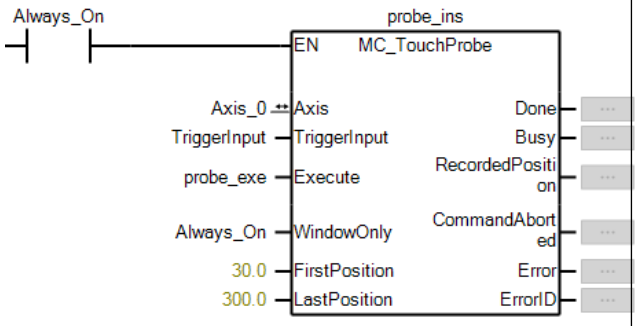
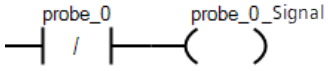
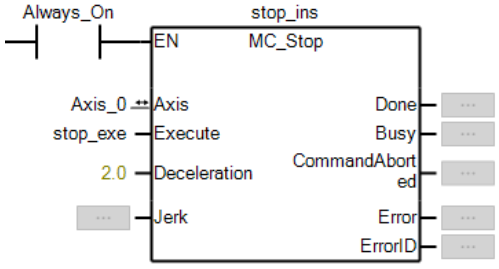
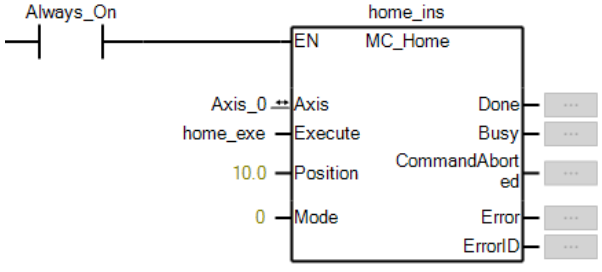
Parameter tuning (Page 875)

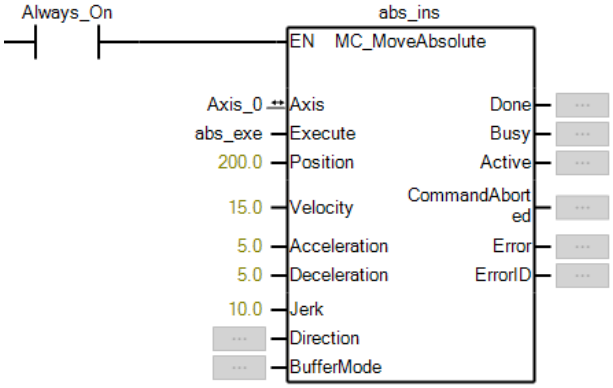
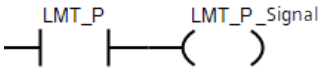
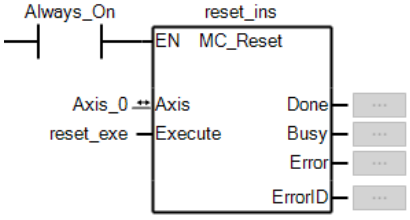
14.10.2 Single Axis example

The following program demonstrates the calling of several single-axis functions, such as Enable, Jog, and Probe.

Table 14-39 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, jog target speed is 25	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 Axis_0 will perform relative positioning based on the following motion parameters: Target position: 300.0 Speed: 10.0 Acceleration: 3.0 Deceleration: 20.0 Jerk: 50.0	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

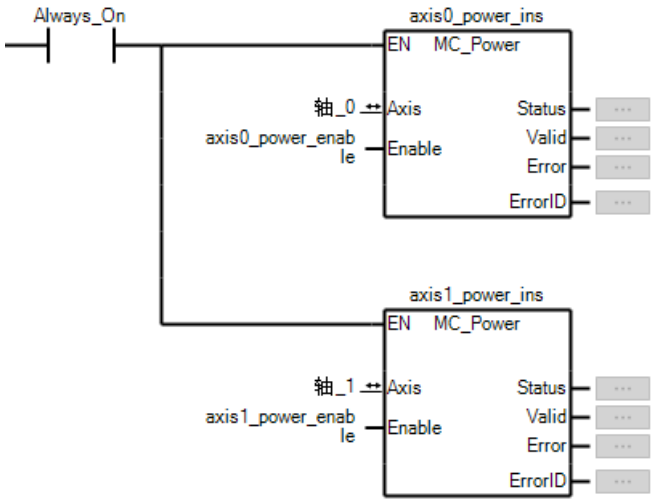
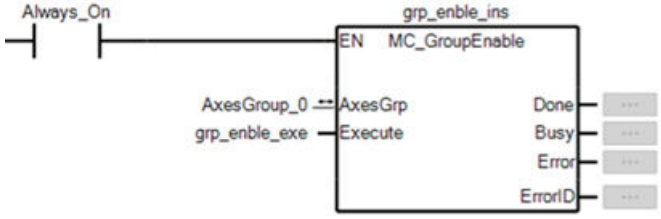
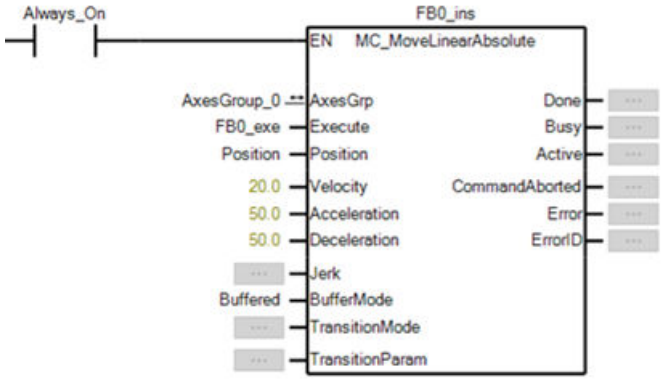
LAD/FBD	Description	STL
Network 4: 	Axis_0, Probe	**LD Always_On** **FBCALL MC_TouchProbe, probe_ins, Axis_0, TriggerInput, probe_exe, Always_On, 30.0, 300.0, probe_ins.Done, probe_ins.Busy, probe_ins.RecordedPosition, probe_ins.CommandAborted, probe_ins.Error, probe_ins.ErrorID**
Network 5: 	Trigger probe	**LDN probe_0** **= CPU_Output11**
Network 6: 	Axis_0 decelerates to a stop with a deceleration of 2.0	**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: 	Operating mode: Mode 0 Home offset: 10.0 Execution result: The current absolute position of Axis_0 has been successfully set to 10.00.	**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**

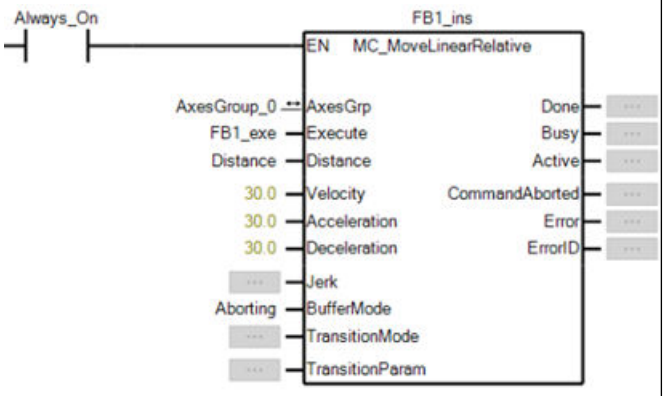
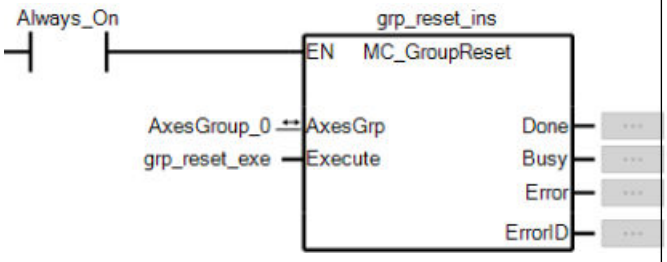
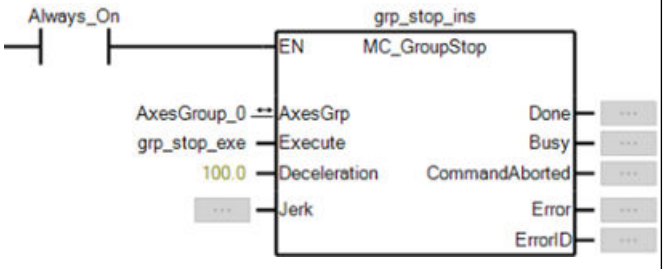
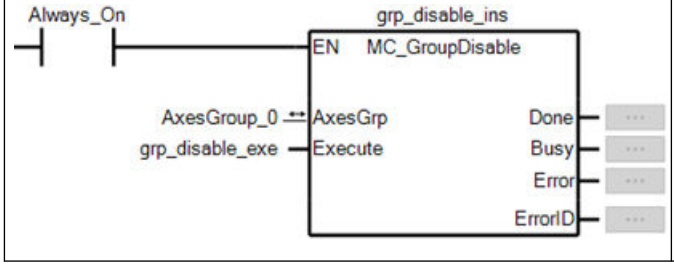
LAD/FBD	Description	STL
Network 8: 	Axis_0 will perform absolute positioning based on the following motion parameters: Target position: 200.0 Speed: 15.0 Acceleration: 5.0 Deceleration: 5.0 Jerk: 10.0	**LD Always_On** **FBCALL MC_MoveAbsolute, abs_ins, Axis_0, abs_exe, 200.0, 15.0, 5.0, 5.0, 10.0, abs_ins.Direction, 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**

14.10.3 Axes group example

The following program demonstrates the calling of axes group.

Table 14-40 Example: Axes group

LAD/FBD	Description	STL
Network 1: 	Enable single axis in axis group	LD Always_On FBCALL MC_Power, axis0_power_ins, 轴_0, axis0_power_enable, axis0_power_ins.Status, axis0_power_ins.Valid, axis0_power_ins.Error, axis0_power_ins.ErrorID FBCALL MC_Power, axis1_power_ins, 轴_1, axis1_power_enable, axis1_power_ins.Status, axis1_power_ins.Valid, axis1_power_ins.Error, axis1_power_ins.ErrorID
Network 2: 	Enable axis 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 3: 	Axis group AxesGroup_0 will perform absolute positioning based on the following motion parameters: Speed: 20.0 Acceleration: 50.0 Deceleration: 50.0	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

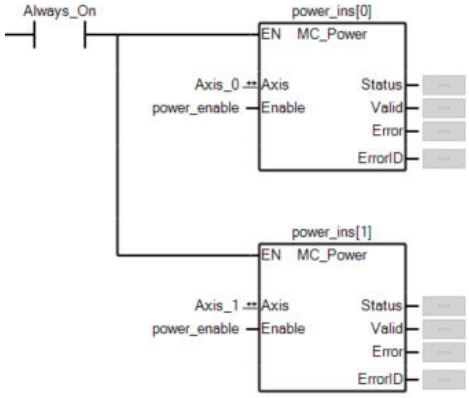
LAD/FBD	Description	STL
Network 4: 	Axis group AxesGroup_0 will perform relative positioning based on the following motion parameters: Speed: 30.0 Acceleration: 30.0 Deceleration: 0.0	`LD Always_On` `FBCALL MC_MoveLinearRelative, FB1_ins, AxesGroup_0, FB1_exe, Distance, 30.0, 30.0, 30.0, FB1_ins.Jerk, 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`
Network 5: 	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 6: 	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 7: 	Disable the axes group function	`LD 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`

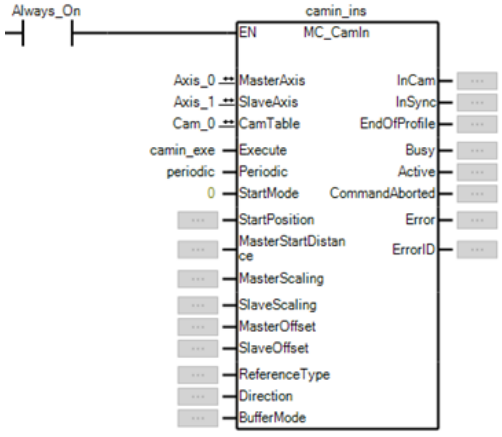
14.10.4 Cam example

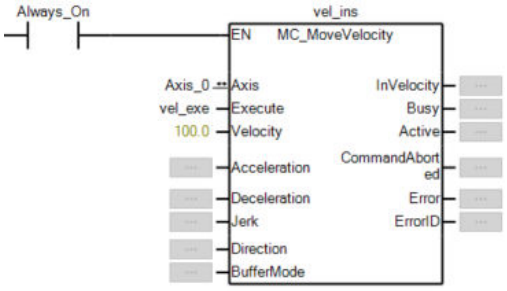
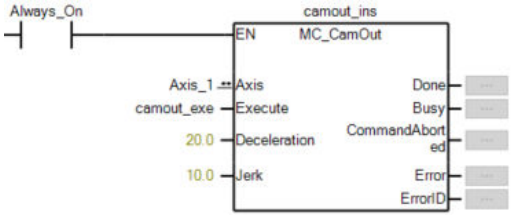
The following program demonstrates how to call cam functions such as Enable, CamIn, CamOut.

Table 14-41 Example: Cam

LAD/FBD	Description	STL

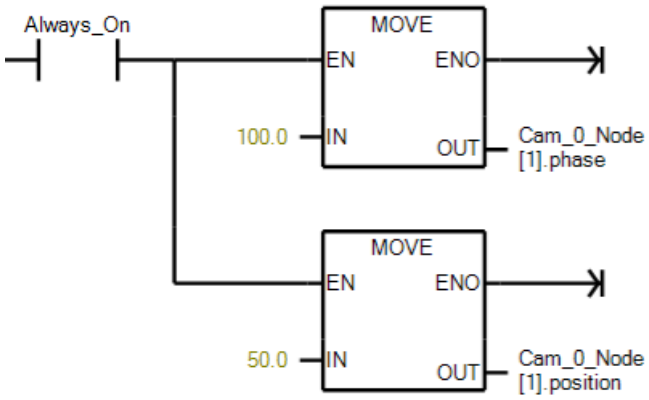
Network 1:
 
	Enable axis	`LD Always_On`
		`FBCALL MC_Power, power_ins[0], Axis_0, power_enable, power_ins[0].Status, power_ins[0].Valid, power_ins[0].Error, power_ins[0].ErrorID`
		`FBCALL MC_Power, power_ins[1], Axis_1, power_enable, power_ins[1].Status, power_ins[1].Valid, power_ins[1].Error, power_ins[1].ErrorID`

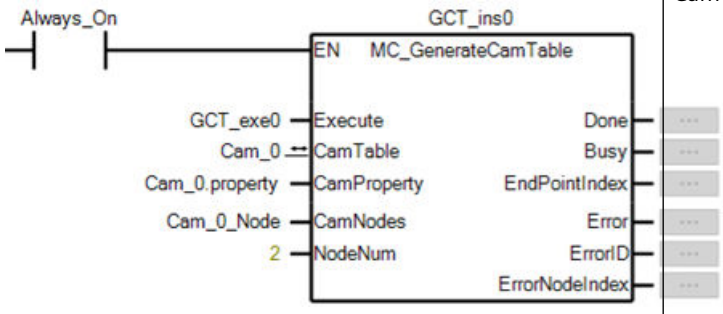
Network 2: 	Trigger slave axis MC_CamIn	`LD Always_On` `FBCALL MC_CamIn, camin_ins, Axis_0, Axis_1, Cam_0, camin_exe, periodic, 0, camin_ins.StartPosition, camin_ins.MasterStartDistance, camin_ins.MasterScaling, camin_ins.SlaveScaling, camin_ins.MasterOffset, camin_ins.SlaveOffset, camin_ins.ReferenceType, 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**

Example: Use the CamProperty, CamNode generated by system

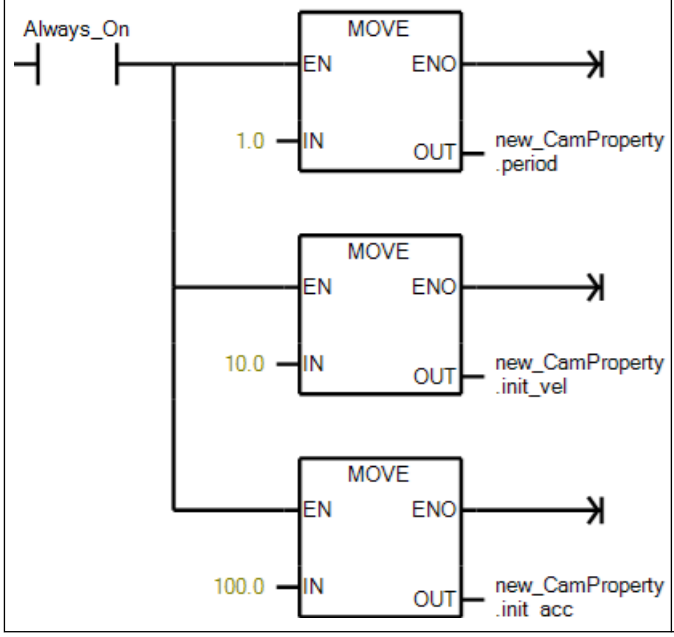
LAD/FBD	Description	STL

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

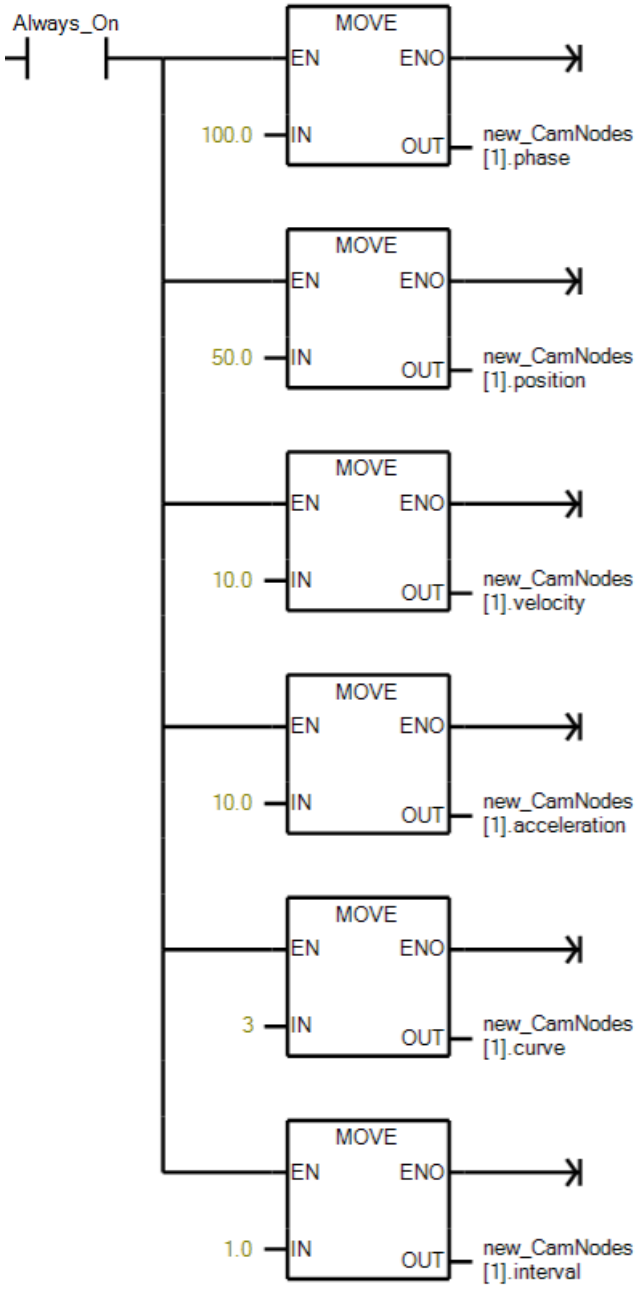
Network 2: 	Trigger MC_Generate-CamTable	**LD Always_On** **FBCALL MC_GenerateCamTable, GCT_ins0, GCT_exe0, Cam_0, Cam_0.property, Cam_0_Node, 2, GCT_ins0.Done, GCT_ins0.Busy, GCT_ins0.EndPointIndex, GCT_ins0.Error, GCT_ins0.ErrorID, GCT_ins0.ErrorNodeIndex**

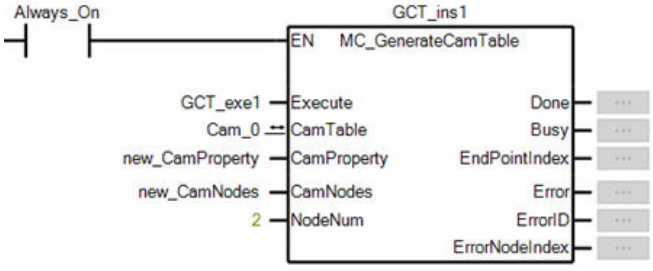
Example: Use the created global variable CamProperty, CamNode

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

Open loop motion control(Legacy) 15

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 Motion control(PLCopen) (Page 765) 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.

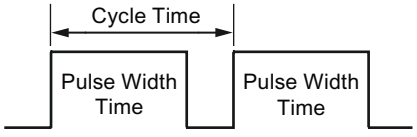
The Axis of Motion provides a single pulse train output with integrated direction control and disable outputs. It also includes programmable inputs which allow the CPU to be configured for several modes of operation, including automatic reference point seek. The Axis of Motion provides a unified solution 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 for configuring Axis of Motion and Axes group of Motion, and a PWM Wizard for configuring 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.

15.1 Using the PWM output

15.1.1 Using the PWM (pulse width modulation) output

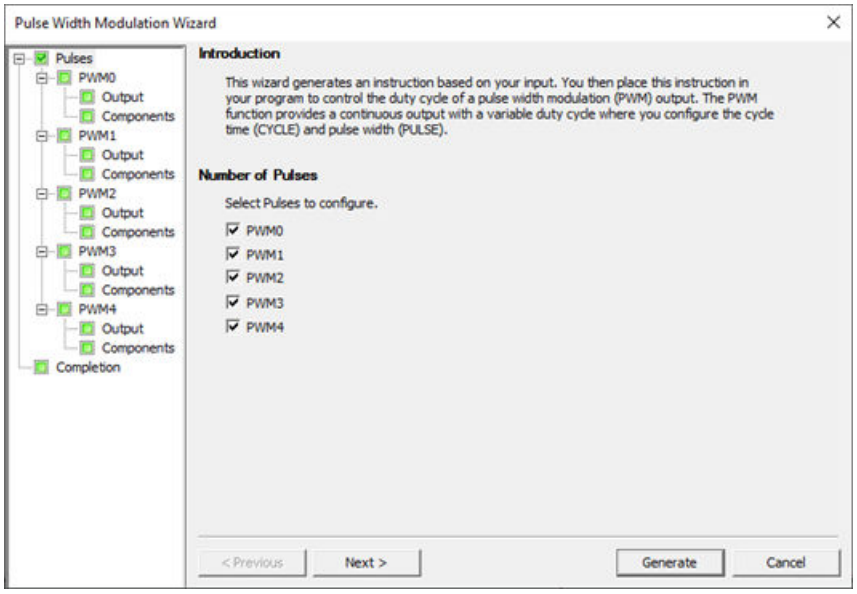
PWM provides a fixed cycle time output with a variable duty cycle. The PWM output runs continuously after being started at the specified frequency (cycle time). The pulse width is varied as required to affect the desired control. Duty cycle can be expressed as a percentage of the cycle time or as a time 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.

15.1.2 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.

15.1.3 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 1041) 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.

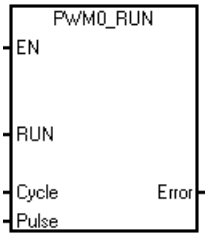
LAD / FBD	STL	Description
	`CALL PWMx_RUN, Cycle, Pulse, Error`	The PWMx_RUN subroutine allows you to control the duty cycle of the output by varying the pulse width from 0 to the pulse width of the cycle time.

Table 15-1 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 15-2 PWMx_RUN instruction error codes

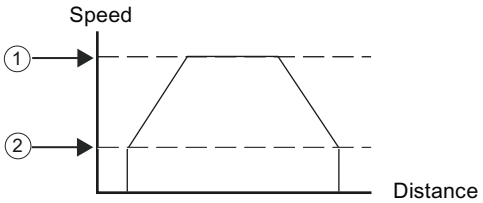
Error code	Description
0	No error, normal completion
131	Pulse generator already in use by another PWM or by a motion axis, or illegal time base change

15.2 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.

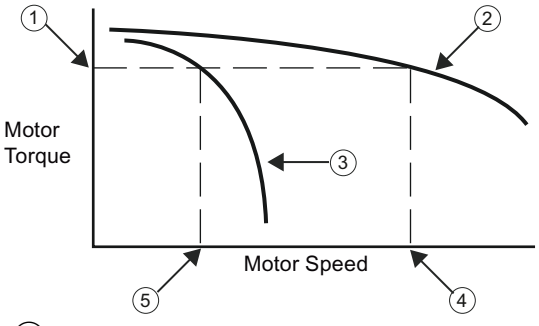
15.2.1 Maximum and start/stop speeds

	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

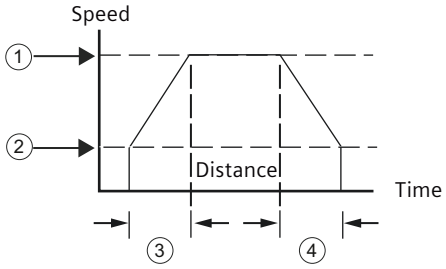
- MAX_SPEED: Enter the value for the optimum operating speed of your application within the torque capability of your motor. The torque required to drive the load is determined by friction, inertia, and the acceleration/deceleration times.
- The Motion wizard calculates and displays the minimum speed 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

15.2.2 Entering the acceleration and deceleration times



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.

① MAX_SPEED

② SS_SPEED

③ ACCEL_TIME

④ DECEL_TIME

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

15.2.3 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

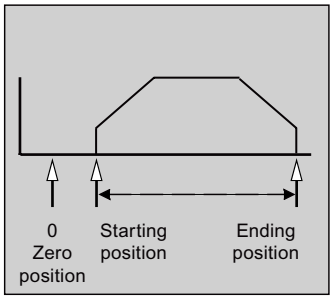
The Motion wizard guides you through a motion profile definition, where you define each motion profile for your application. For each profile, you select the operating mode and define the specifics of each individual step for the profile. The Motion wizard also allows you to define a symbolic name for each profile by simply entering the symbol name as you define the profile.

Selecting the mode of operation for the profile

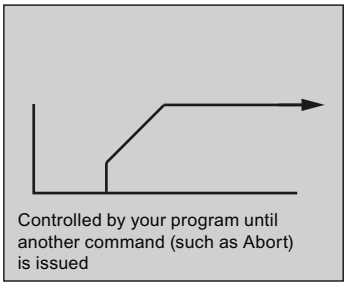
You configure the profile according to the mode of operation desired. The Axis of Motion supports absolute position, relative position, single-speed continuous rotation, and two-speed continuous rotation. The following figure shows the different modes of operation.

Mode selections for the Axis of Motion

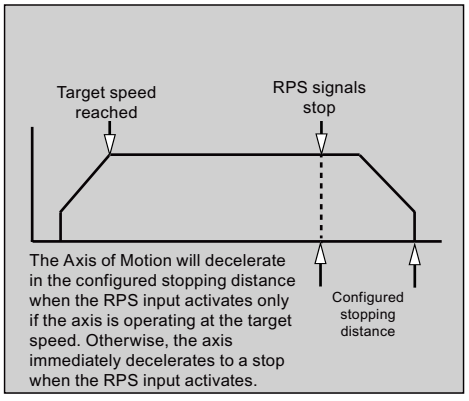
Absolute Position



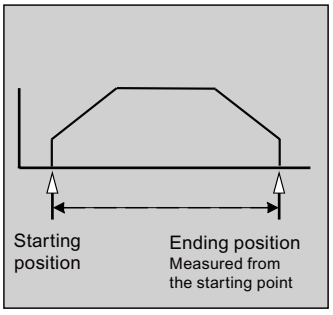
Single-Speed, Continuous Rotation



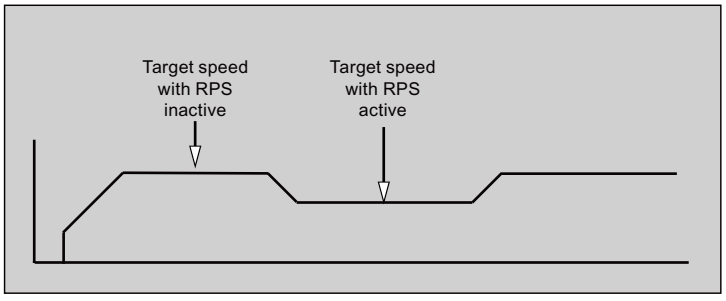
Single-speed, Continuous Rotation, with triggered stop



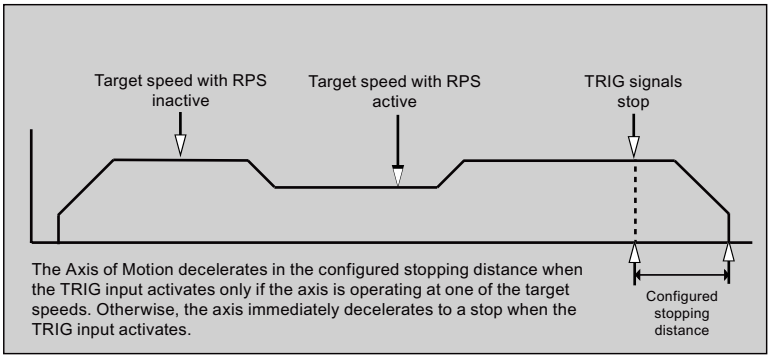
Relative Position



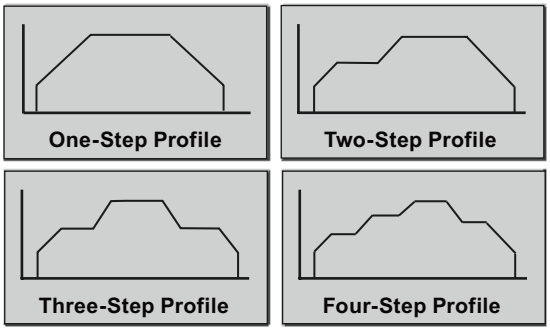
Two-Speed, Continuous Rotation



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.

15.3 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 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 15-3 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 15-4 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.

15.4 Using Axis of motion

15.4.1 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.

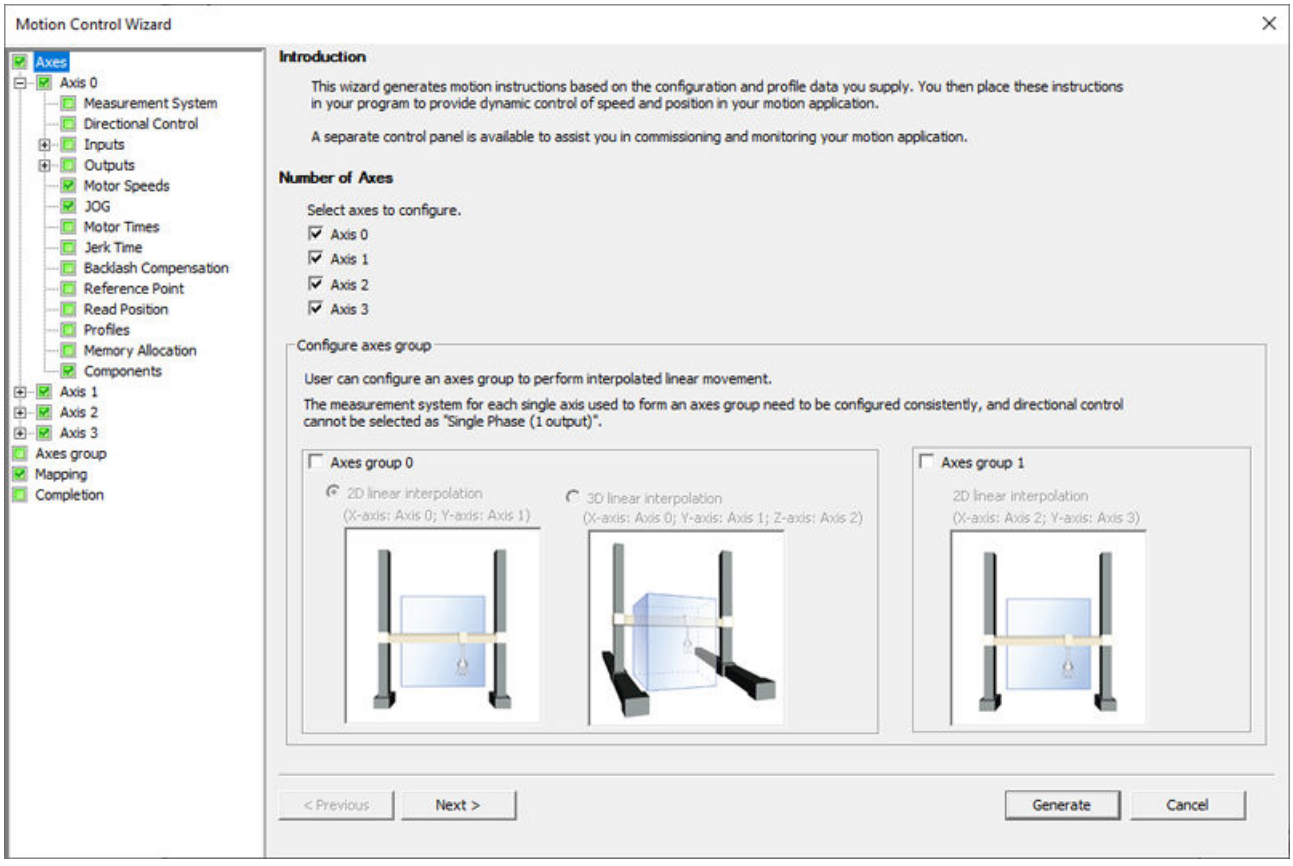
Note

Make sure to match the measurement system configuration to the pulses/revolution and distance/revolution specifications of your stepper/servo motor controller system.

15.4.2 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 15-5 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>Out-put</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="2">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="2">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 rowspan="2">DIS</td><td colspan="2" rowspan="2">Q0.4</td><td colspan="2" rowspan="2">Q0.5</td><td>Q0.6</td><td>Axis 3 is not configured.</td><td rowspan="2">Q1.3</td></tr>
<tr><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.

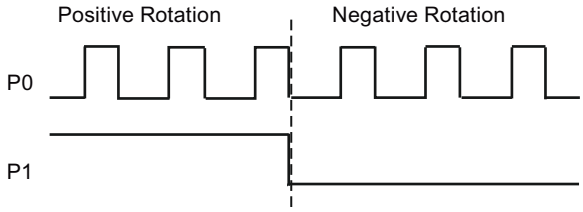
- In the "Directional Control" node, you can select the following "Phasing" modes:
  - Single phase (2 output)
  - Two-phase (2 output)
  - AB quadrature phase (2 output)
  - Single phase (1 output)

  You can also select the "Polarity" (positive or negative) of the outputs.

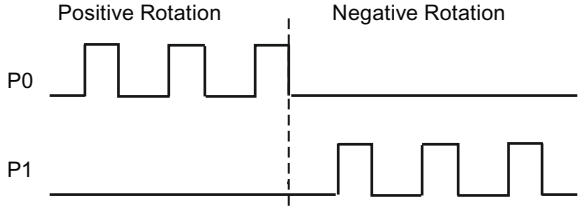
Phasing

You have four options for the "Phasing" interface to the stepper/servo drive. These options are as follows:

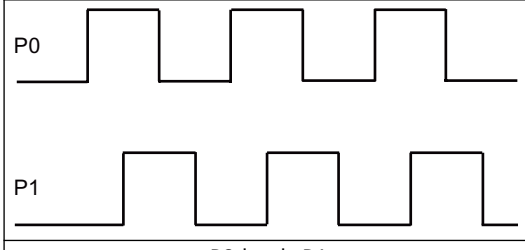
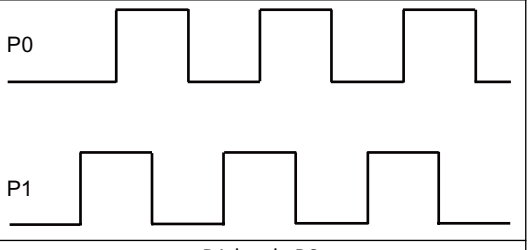
- Single phase (2 output): If you select the single phase (2 output) option, then one output (P0) controls the pulsing, and one output (P1) controls the direction. P1 is high (active) if the pulsing is in the positive direction. P1 is low (inactive) if the pulsing is in the negative direction. Single 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

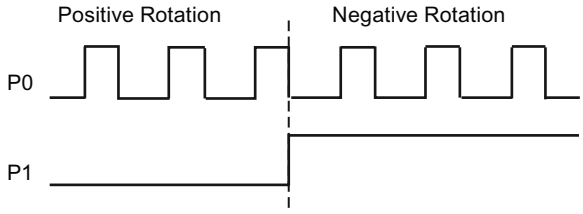
- Single phase (1 output): If you select the single phase (1 output) option, then the output (P0) controls the pulsing. Only positive motion commands are accepted by the CPU in this mode. The Motion wizard restricts you from making illegal negative configurations when you select this mode. You can save an output if your motion application is in one direction only. Single phase (1 output) is shown in the figure below (assuming positive polarity):



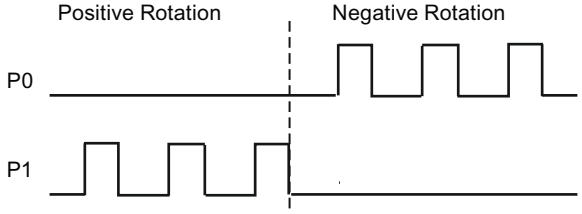
Polarity

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:

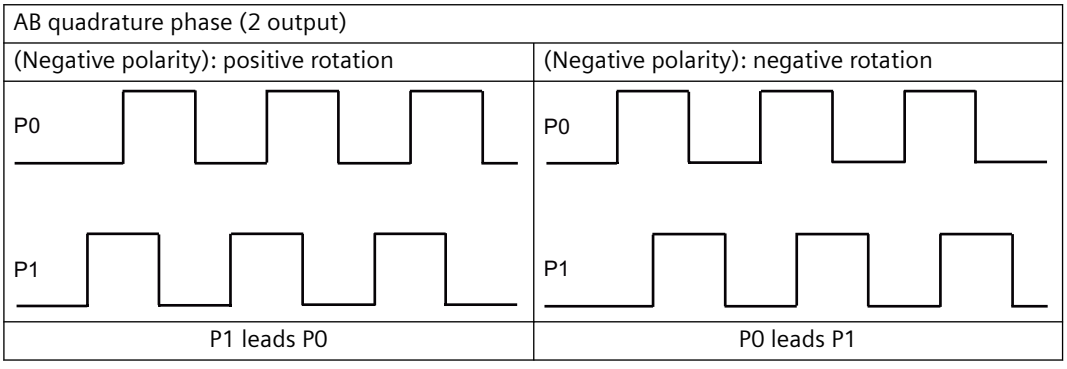
- Single phase (2 output): P1 is low (inactive) if the pulsing is in the positive direction. P1 is high (active) if the pulsing is in the negative direction. This is shown in the figure below:



- 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:



- 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.

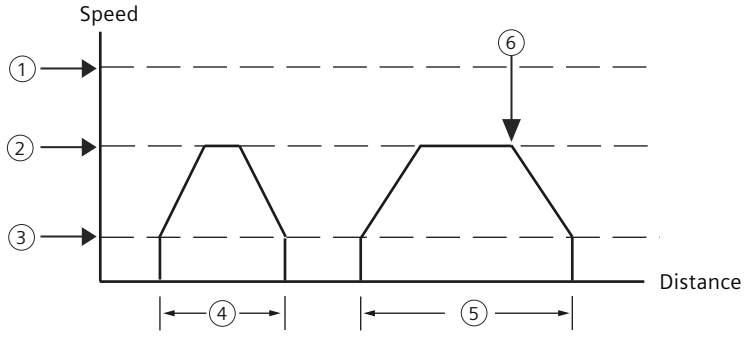
Entering maximum start and stop speed

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.

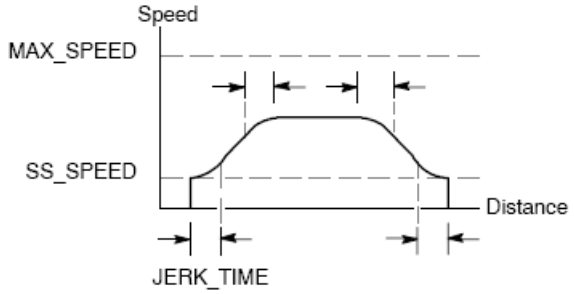
Note

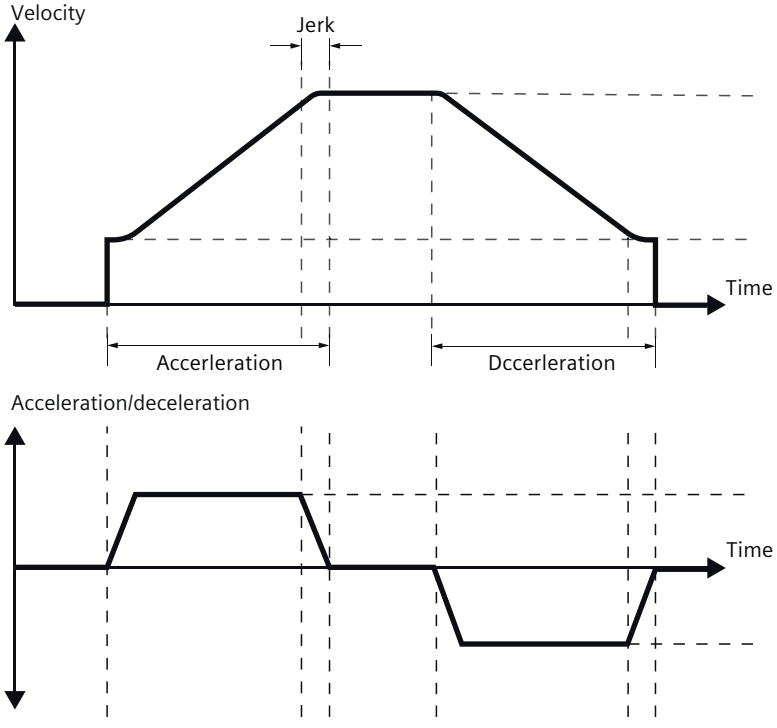
Axes group of motion

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 1123) instruction input parameter, or in the Axes group segment planning (Page 1115) 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 1123) instruction input parameter, or in the Axes group segment planning (Page 1115) 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.

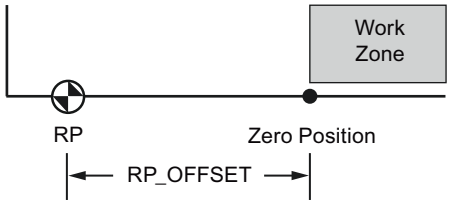
Note

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

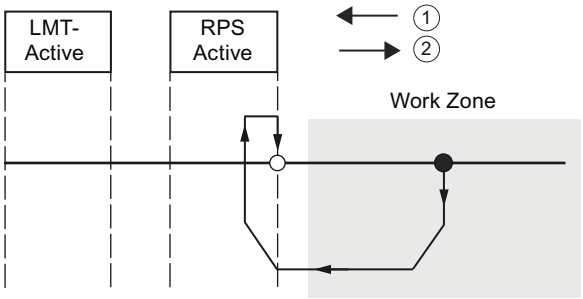


Figure 15-1 ①: RP seek direction②: RP approach direction

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.

15.4.3 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 1064)	Provides initialization and overall control of the axis
AXISx_MAN (Page 1066)	Used for manual mode operation of the axis
AXISx_GOTO (Page 1067)	Commands the axis to go to a specified location
AXISx_RUN (Page 1069)	Commands the axis to execute a configured motion profile
AXISx_RSEEK (Page 1070)	Initiates a reference point seek operation
AXISx_LDOFF (Page 1071)	Establishes a new zero position that is offset from the reference point position
AXISx_LDPOS (Page 1072)	Changes the axis position to a new value
AXISx_SRATE (Page 1073)	Modifies the configured acceleration, deceleration, and jerk compensation times
AXISx_DIS (Page 1074)	Controls the DIS output
AXISx_CFG (Page 1074)	Reads the configuration block and updates the axis setup as required
AXISx_CACHE (Page 1075)	Pre-caches a configured motion profile
AXISx_RDPOS (Page 1076)	Returns the current axis position
AXISx_ABSPOS (Page 1077)	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

15.4.3.1 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.

15.4.3.2 AXISx_CTRL subroutine

Table 15-6 AXISx_CTRL

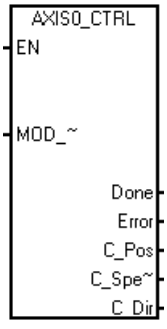
LAD / FBD	STL	Description
	**CALL AXISx_CTRL, MOD_EN, Done, Error, C_Pos, C_Speed, C_Dir**	The AXISx_CTRL subroutine (Control) enables and initializes the Axis of Motion by automatically commanding the Axis of Motion to load the configuration/profile table each time the CPU changes to RUN mode. Use this 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 15-7 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 1099) 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.

15.4.3.3 AXISx_MAN subroutine

Table 15-8 AXISx_MAN

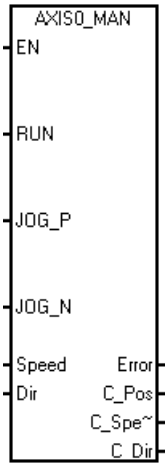
LAD / FBD	STL	Description
	**CALL AXISx_MAN, RUN, JOG_P, JOG_N, Speed, Dir, Error, C_Pos, C_Speed, C_Dir**	The AXISx_MAN subroutine (Manual Mode) puts the Axis of Motion into manual mode. This allows the motor to be run at different speeds or to be jogged in a positive or negative direction. You can enable only one of the RUN, JOG_P, or JOG_N inputs at a time.

Table 15-9 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 1099) 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

15.4.3.4 AXISx_GOTO subroutine

Table 15-10 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 15-11 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 1099) 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.

15.4.3.5 AXISx_RUN subroutine

Table 15-12 AXISx_RUN

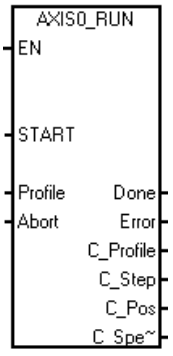
LAD / FBD	STL	Description
	**CALL AXISx_RUN, START, Profile, Abort, Done, Error, C_Profile, C_Step, C_Pos, C_Speed**	The AXISx_RUN subroutine (Run Profile) commands the Axis of Motion to execute the motion operation in a specific profile stored in the configuration/profile table.

Table 15-13 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 1099) 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).

15.4.3.6 AXISx_RSEEK subroutine

Table 15-14 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 15-15 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 1099) contains the result of this subroutine.

15.4.3.7 AXISx_LDOFF subroutine

Table 15-16 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 15-17 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 1099) contains the result of this subroutine.

15.4.3.8 AXISx_LDPOS subroutine

Table 15-18 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 15-19 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 1099) 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).

15.4.3.9 AXISx_SRATE subroutine

Table 15-20 AXISx_SRATE

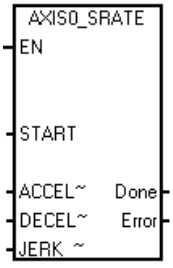
LAD / FBD	STL	Description
	**CALL AXISx_SRATE, START, ACCEL_Time, DECEL_Time, JERK_Time, Done, Error**	The AXISx_SRATE subroutine (Set Rate) commands the Axis of Motion to change the acceleration, deceleration, and jerk times.

Table 15-21 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 1099) contains the result of this subroutine.

15.4.3.10 AXISx_DIS subroutine

Table 15-22 AXISx_DIS

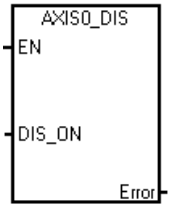
LAD / FBD	STL	Description
	**CALL AXISx_DIS, DIS_ON, Error**	The AXISx_DIS subroutine turns the DIS output of the Axis of Motion on or off. This allows you to use the DIS output for disabling or enabling a motor controller. If you use the DIS output on the Axis of Motion, then this subroutine can be called every scan or only when you need to change the value of the DIS output.

Table 15-23 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 1099) contains the result of this subroutine.

15.4.3.11 AXISx_CFG subroutine

Table 15-24 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 15-25 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 1099) contains the result of this subroutine.

15.4.3.12 AXISx_CACHE subroutine

Table 15-26 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 15-27 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 1099) contains the result of this subroutine.

15.4.3.13 AXISx_RDPOS subroutine

Table 15-28 AXISx_RDPOS

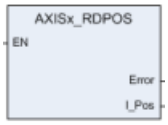
LAD / FBD	STL	Description
	`CALL AXISx_RDPOS, Error, I_Pos`	The AXISx_RDPOS subroutine returns the current motion axis position.

Table 15-29 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 1099) 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.

15.4.3.14 AXISx_ABSPOS subroutine

Table 15-30 AXISx_ABSPOS

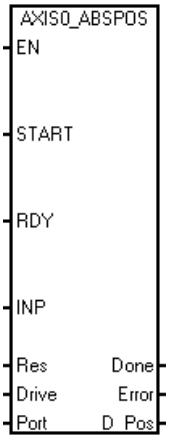
LAD / FBD	STL	Description
	**CALL AXISx_ABSPOS, START, RDY, INP, Res, Drive, Port, Done, Error, D_Pos**	The AXISx_ABSPOS subroutine reads the absolute position from certain Siemens servo drives, such as the V90. You read the absolute position value in order to update the current position value in the 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 15-31 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 1099) 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.

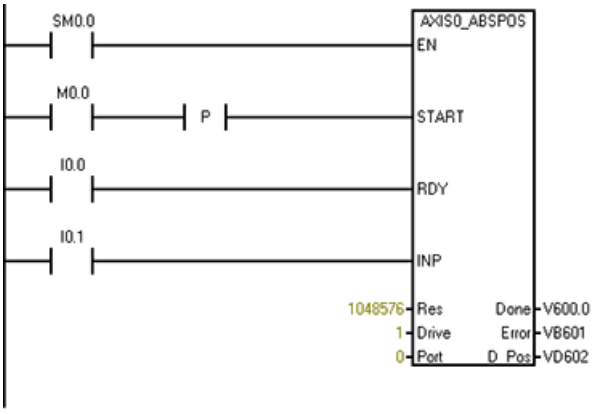
15.4.4 Using the AXISx_ABSPOS subroutine to read the absolute position from a SINAMICS servo drive

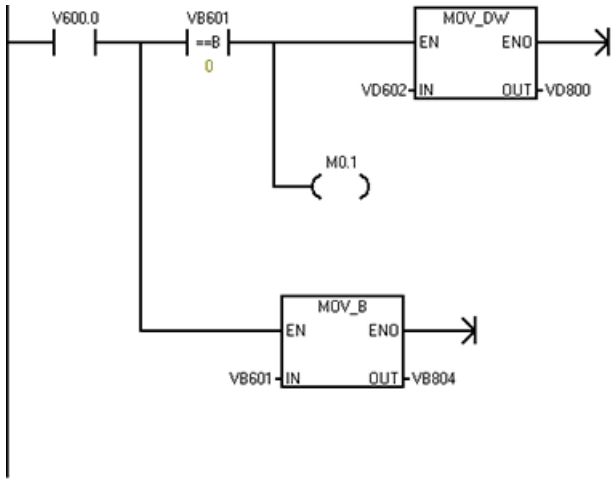
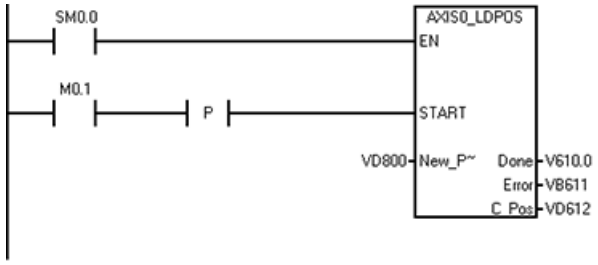
The following sections provide information about how to use the AXISx_ABSPOS subroutine in your project to read the absolute position from a SINAMICS V90 servo drive.

15.4.4.1 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 15-32 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_ABSPOS, 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**

15.4.4.2 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.

15.4.4.3 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.

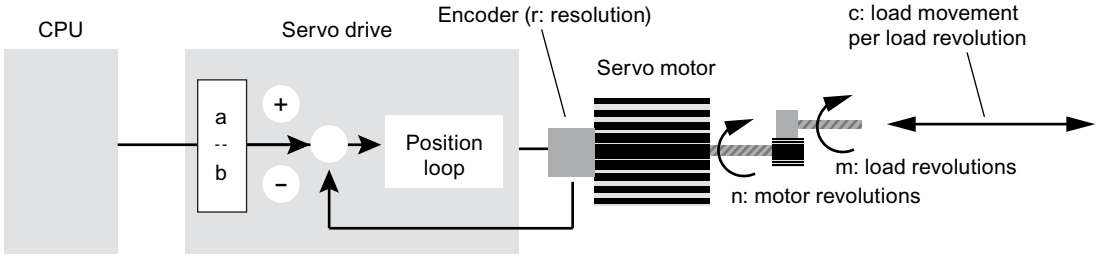
Setpoint pulse train input format

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.

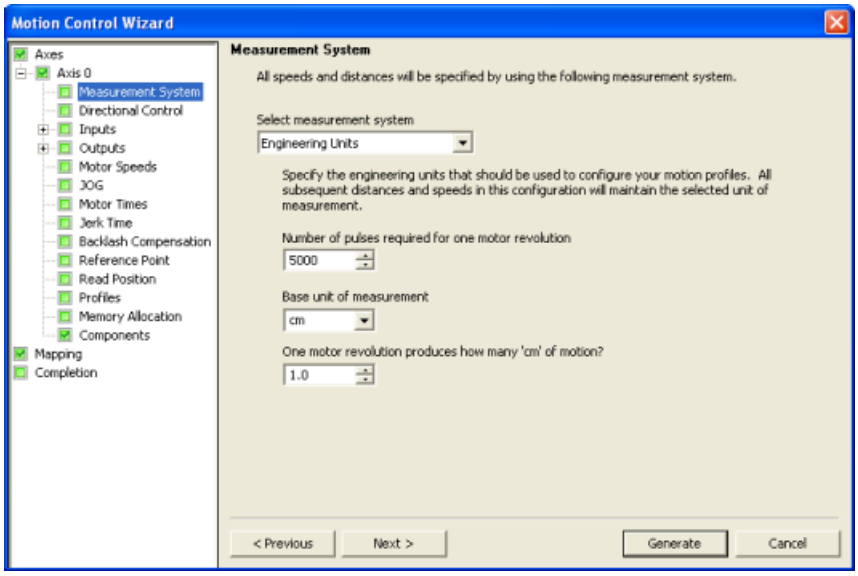
Encoder resolution

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)

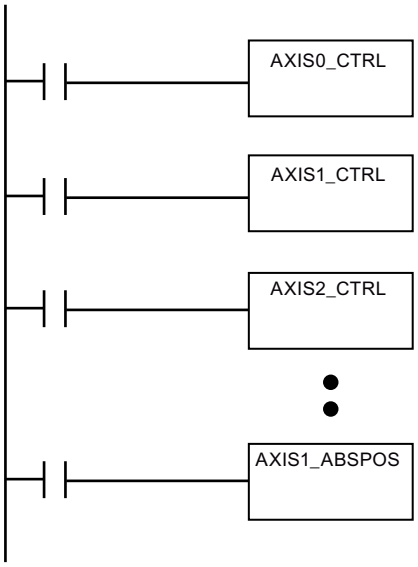
where, "c" = load movement (in the defined engineering unit) per load revolution, "m/n" = external gearing ratio expressed as load revolutions per motor revolution

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**

15.4.4.4 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).

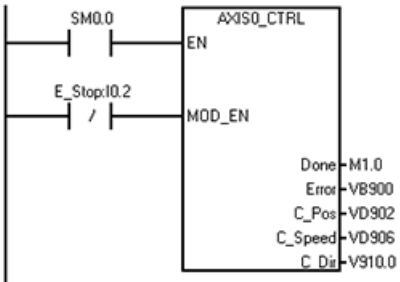
15.4.5 Axis of Motion example programs

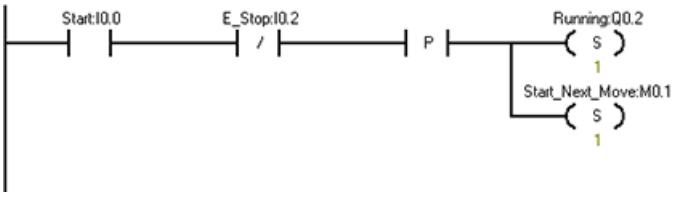
15.4.5.1 Axis of Motion simple relative move (cut-to-length application) example

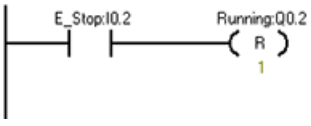
The example program shows a simple relative move that uses the AXISx_CTRL and AXISx_GOTO subroutines to perform a cut-to-length operation. This program does not require an RP seek mode or a motion profile, and the length can be measured in either pulses or engineering units. Enter the length (VD500) 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 15-33 Example: Axis of Motion simple relative move (cut-to-length application)

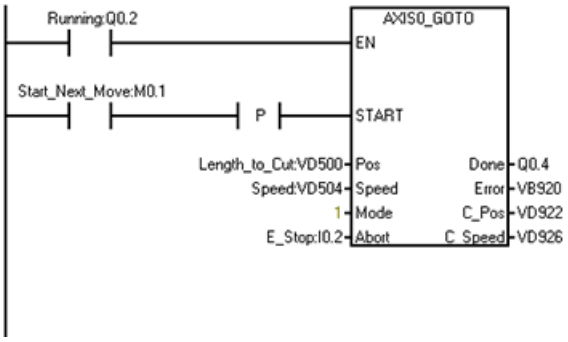
LAD/FBD	Description	STL

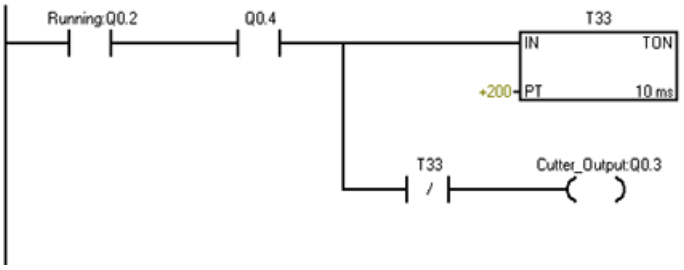
Network 1:
 
	Control instruction	`LD SM0.0`
		`= L60.0`
		`LDN I0.2`
		`= L63.7`
		`LD L60.0`
		`CALL AXIS0_CTRL, L63.7, M1.0, VB900, VD902, VD906, V910.0`

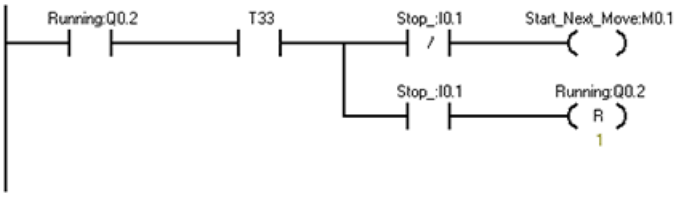
Network 2:
 
	Start puts machine into automatic mode.	`LD I0.0`
		`AN I0.2`
		`EU`
		`S Q0.2, 1`
		`S M0.1, 1`

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`

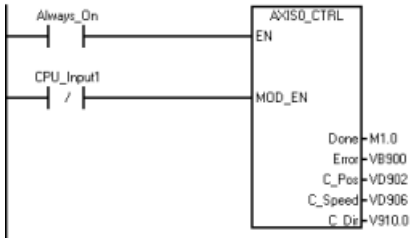
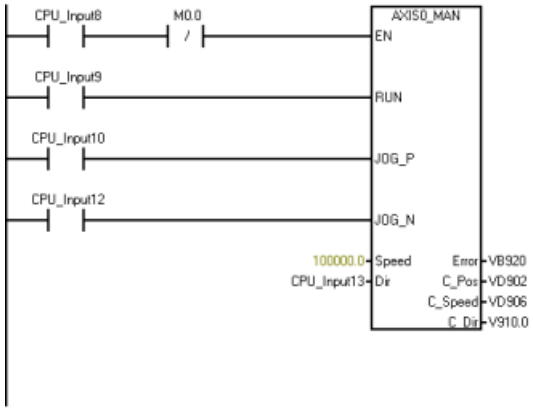
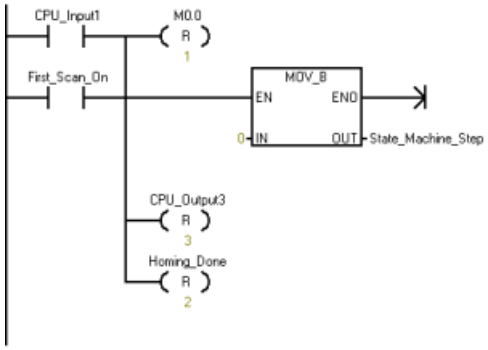
Network 5:
 
	When in position, turn on the cutter for 2 seconds to finish the cut.	`LD Q0.2`
		`A Q0.4`
		`TON T33, +200`
		`AN T33`
		`= Q0.3`

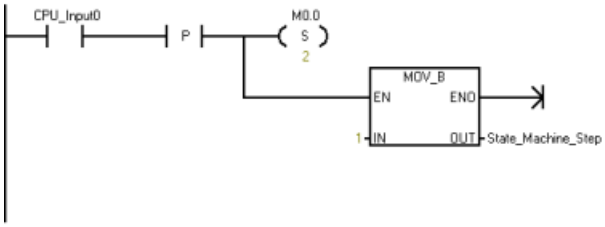
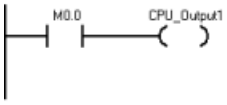
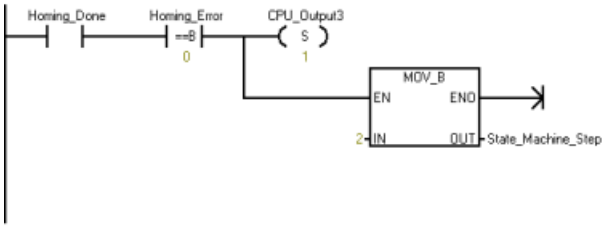
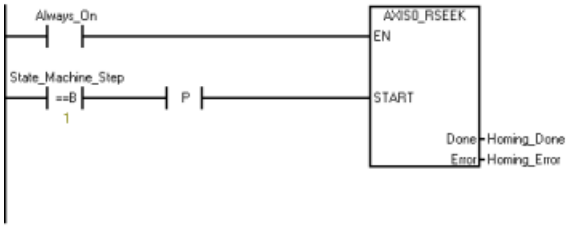
Network 6:
 
	When the cut is finished, then restart unless the Stop is active.	`LD Q0.2`
		`A T33`
		`LPS`
		`AN I0.1`
		`= M0.1`
		`LPP`
		`A I0.1`
		`R Q0.2, 1`

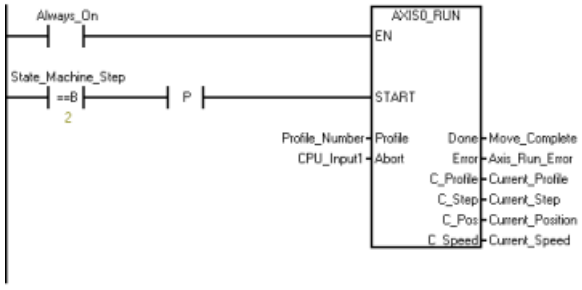
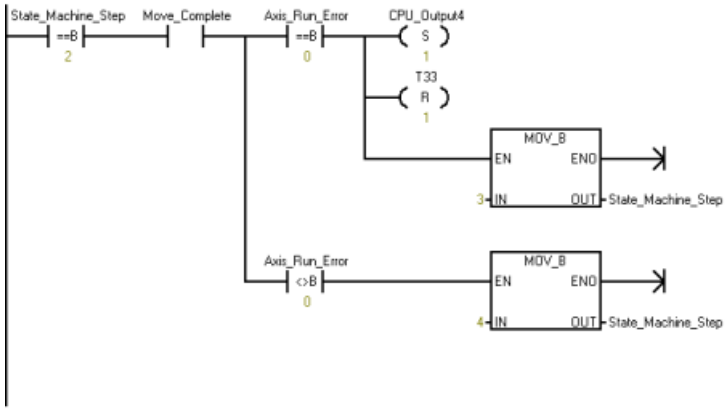
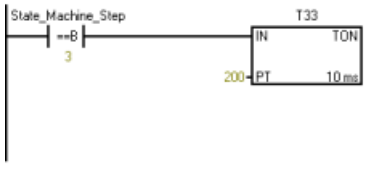
15.4.5.2 Axis of Motion AXISx_CTRL, AXISx_RUN, AXISx_SEEK, and AXISx_MAN example

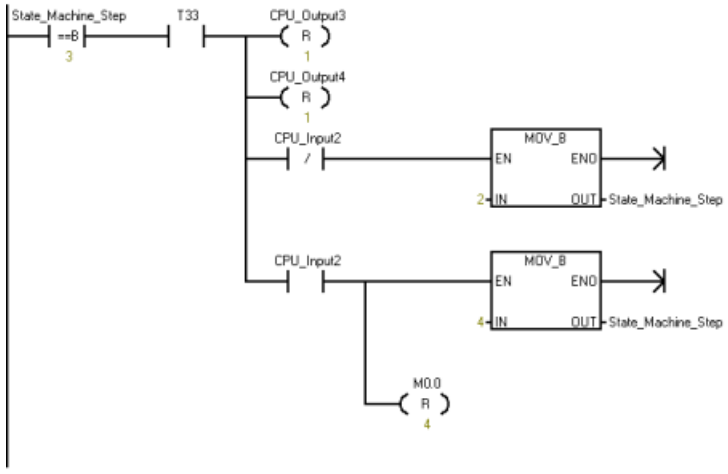
This program provides an example of the AXISx_CTRL, AXISx_RUN, AXISx_RSEEK, and AXISx_MAN subroutines. You must configure the RP seek mode and a motion profile.

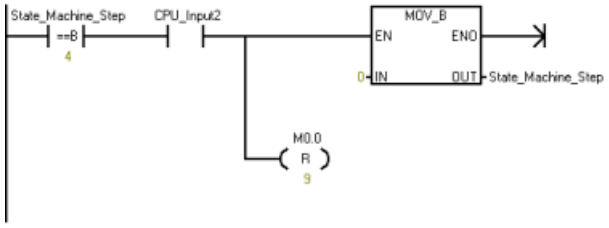
Table 15-34 Example: Axis of Motion AXISx_CTRL, AXISx_RUN, AXISx_SEEK, and AXISx_MAN subroutines application

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**

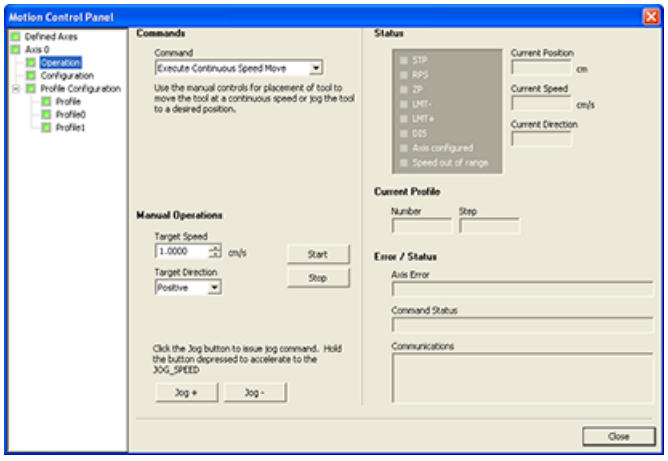
LAD/FBD	Description	STL
Network 13 	If the State Machine is in Step 4, flash output 5 to indicate an error. Symbol and Address: [1] • Clock_1s = SM0.5 (Clock pulse that is ON for 0.5 s and OFF for 0.5 s for a duty cycle time of 1 s.) • CPU_Output5 = Q0.5 • State_Machine_Step = VB1500	**LDB= State_Machine_ 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.

15.4.6 Monitoring the Axis of Motion

15.4.6.1 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.

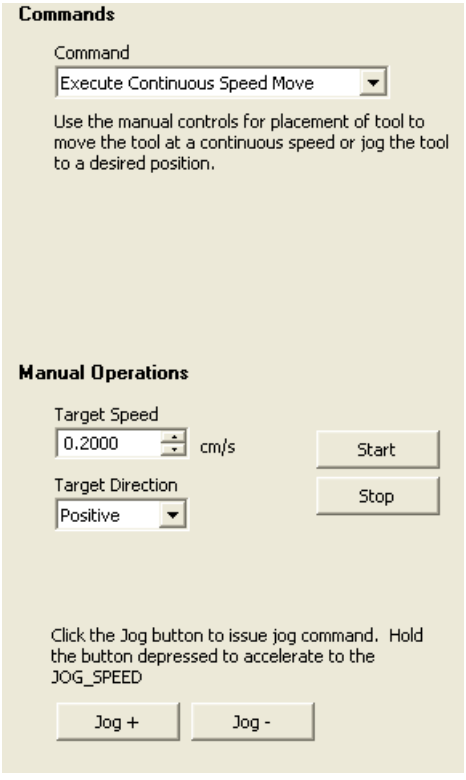
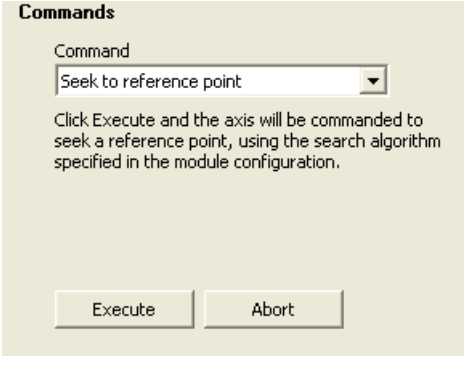
Note

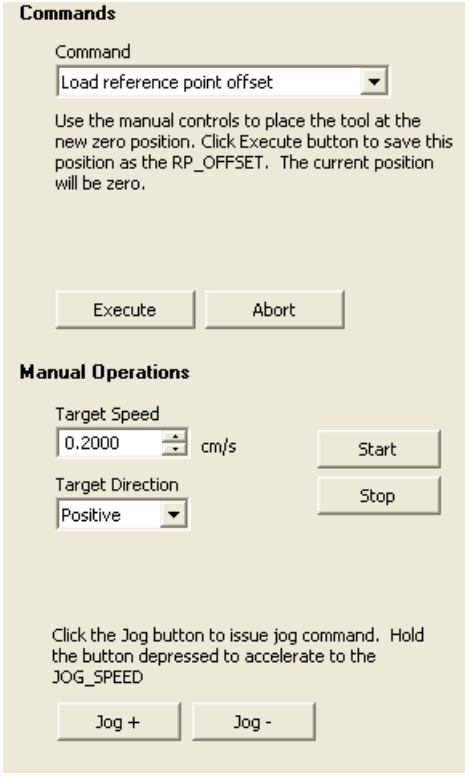
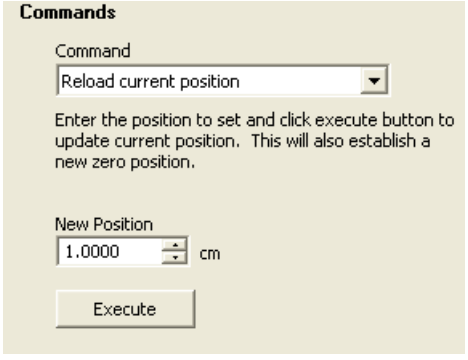
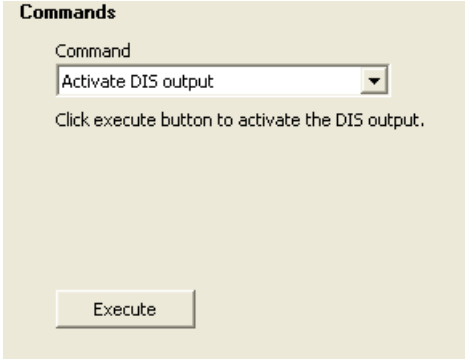
Exiting the Motion control panel or a loss of communications while a motion command is active causes the axis to stop its motion after a 5 second timeout.

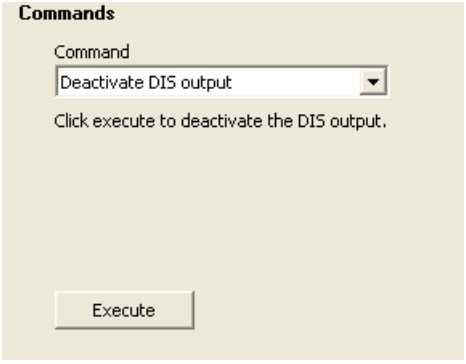
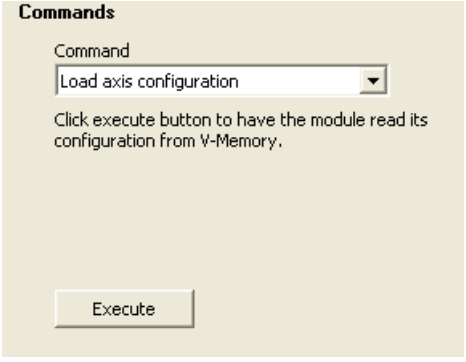
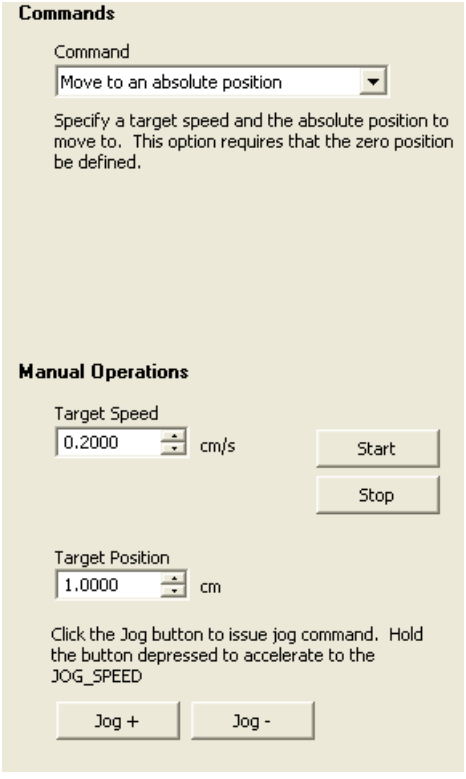
Motion commands

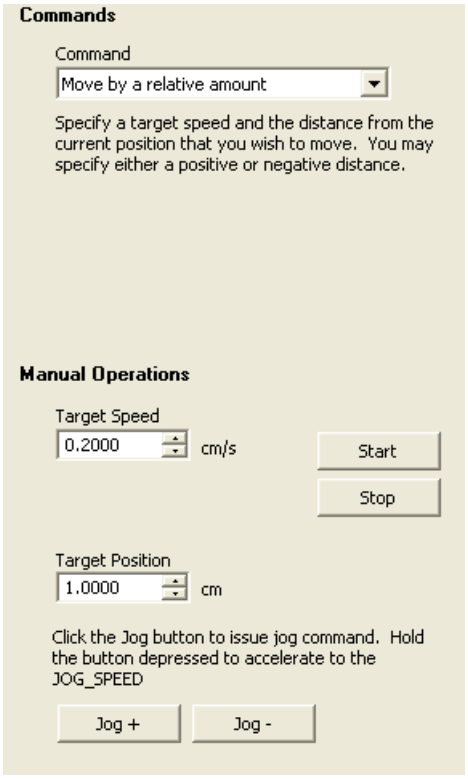
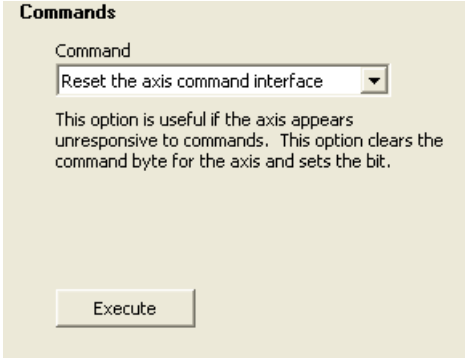
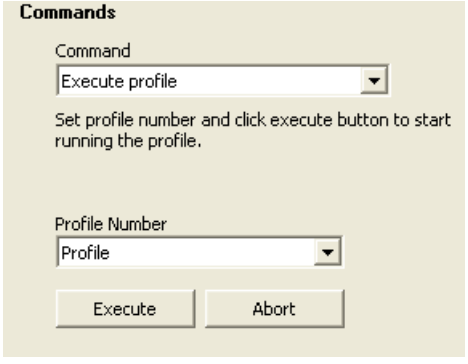
You can also generate the following motion commands:

Table 15-35 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.

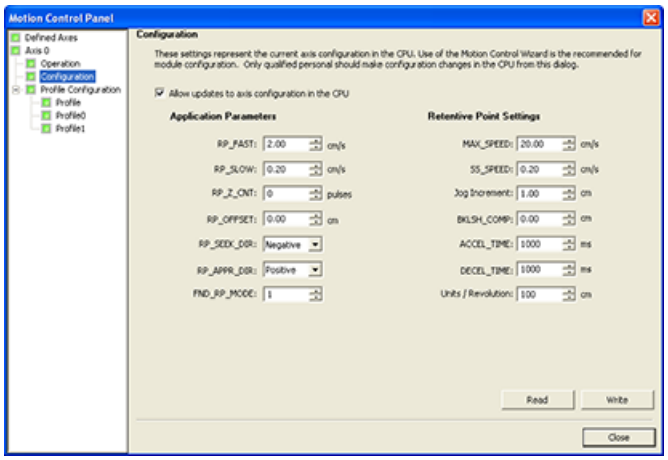
Command	Description
	Load reference point offset: After you use the manual controls to jog the tool to the new position, you then load the "Reference Point Offset". Use the manual controls to place the tool at the new position. Click "Execute" to save this position as the "RP_OFFSET". The current position 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.

15.4.6.2 Displaying and modifying the configuration of the Axis of Motion

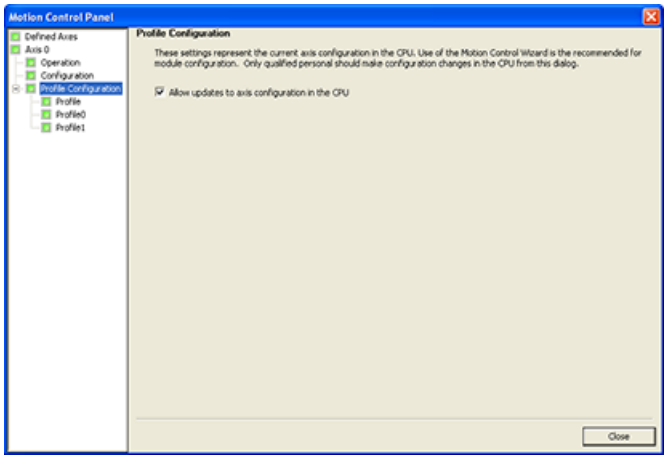
In the Configuration node, you can view and modify the configuration settings for the Axis of Motion that is stored in the data block of the CPU.



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.

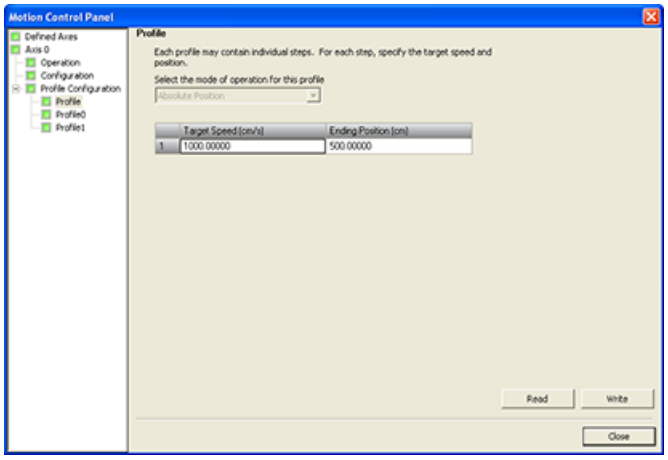
15.4.6.3 Displaying the profile configuration for the Axis of Motion

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.



15.4.6.4 Error codes for the Axis of Motion (WORD at SMW620, SMW670, SMW720 or SMW770)

Table 15-36 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.

15.4.6.5 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 15-37 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.

15.4.7 Advanced topics

15.4.7.1 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.

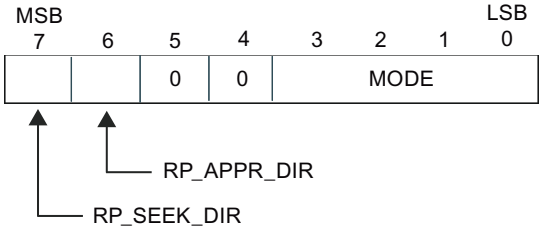
Note

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 15-38 Configuration/Profile table: Configuration block

Configuration/Profile table			
Byte offset	**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 offset	**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)	--

<table>
<tr><th colspan="4">Configuration/Profile table</th></tr>
<tr><th>Byte off-set</th><th>Name</th><th>Function description</th><th>Type</th></tr>
<tr><td colspan="4">Configuration block</td></tr>
<tr><td>32</td><td>RP_CFG</td><td>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).
<table>
<tr><td>MODE</td><td>Specifies the reference point search method</td></tr>
<tr><td>'0000'</td><td>Reference point search disabled.</td></tr>
<tr><td>'0001'</td><td>The reference point is where the RPS input goes active.</td></tr>
<tr><td>'0010'</td><td>The reference point is centered within the active region of the RPS input.</td></tr>
<tr><td>'0011'</td><td>The reference point is outside the active region of the RPS input.</td></tr>
<tr><td>'0100'</td><td>The reference point is within the active region of the RPS input.</td></tr>
<tr><td>'0101' to '1111'</td><td>Reserved (error if selected)</td></tr>
</table></td><td>--</td></tr>
<tr><td>33</td><td>--</td><td>Reserved: Set to 0</td><td>--</td></tr>
<tr><td>34</td><td>RP_Z_CNT</td><td>Number of pulses of the ZP input used to define the reference point (4 bytes)</td><td>DInt</td></tr>
<tr><td>38</td><td>RP_FAST</td><td>Fast speed for the RP seek operation: MAX_SPD or less (4 bytes)</td><td>DInt/Real</td></tr>
<tr><td>42</td><td>RP_SLOW</td><td>Slow speed for the RP seek operation: Maximum speed from which the motor can instantly go to a stop or less (4 bytes)</td><td>DInt/Real</td></tr>
<tr><td>46</td><td>SS_SPEED</td><td>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.</td><td>DInt/Real</td></tr>
<tr><td>50</td><td>MAX_SPEED</td><td>Maximum operating speed of the motor (4 bytes)</td><td>DInt/Real</td></tr>
<tr><td>54</td><td>JOG_SPEED</td><td>Jog speed (4 bytes): MAX_SPEED or less (4 bytes)</td><td>DInt/Real</td></tr>
</table>

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 15-39 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 15-40 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 15-41 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 15-42 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 15-43 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 15-44 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 15-45 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 15-46 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 15-47 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

15.4.7.2 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.

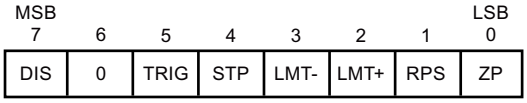
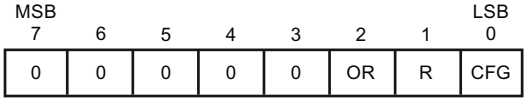
Table 15-48 Special memory bytes SMB600 to SMB799

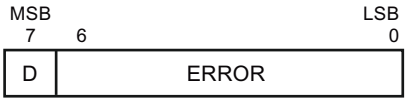
SM bytes for Axes of Motion:			
Axis of Motion 0	**Axis of Motion 1**	**Axis of Motion 2**	**Axis of Motion 3**
SMB600 to SMB649	SMB650 to SMB699	SMB700 to SMB749	SMB750 to SMB799

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 15-49 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 1098) list.)

SM address	Description
SMB622	Axis 0: Input/output status: Reflects the status of the inputs and outputs  • 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
SMB623	Axis 0 Instantaneous status: Reflects the status of the configuration and rotation direction status  • 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

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 1099) 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.

15.4.8 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.

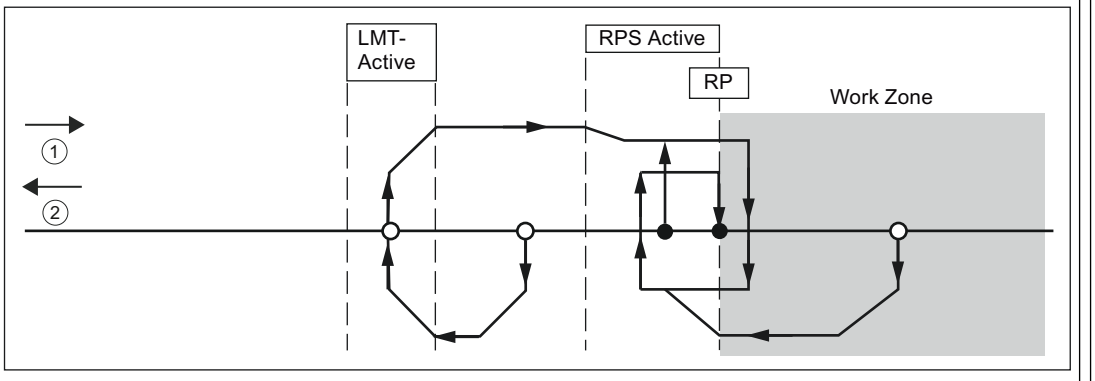
By selecting the location of the work zone in this way, all the backlash of the mechanical gearing system is removed for the first move to the work zone after a reference point seek.

Note

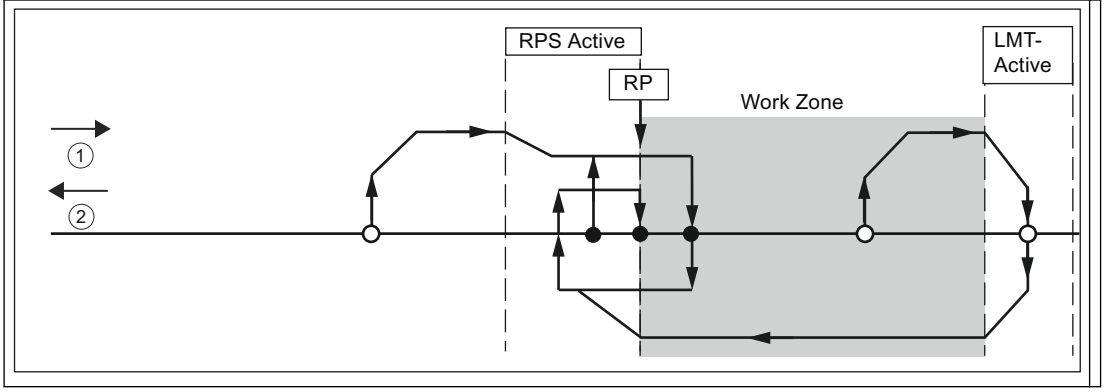
The RPS input must be enabled to use the RP Seek functionality. The ZP input must also be enabled if RP Seek modes 3 or 4 are to be used, unless you configure the number of ZP pulses to be received after entering the RPS active region to a value of "0".

RP seek mode 1

Default configuration: RP seek direction: negative and RP approach direction: positive



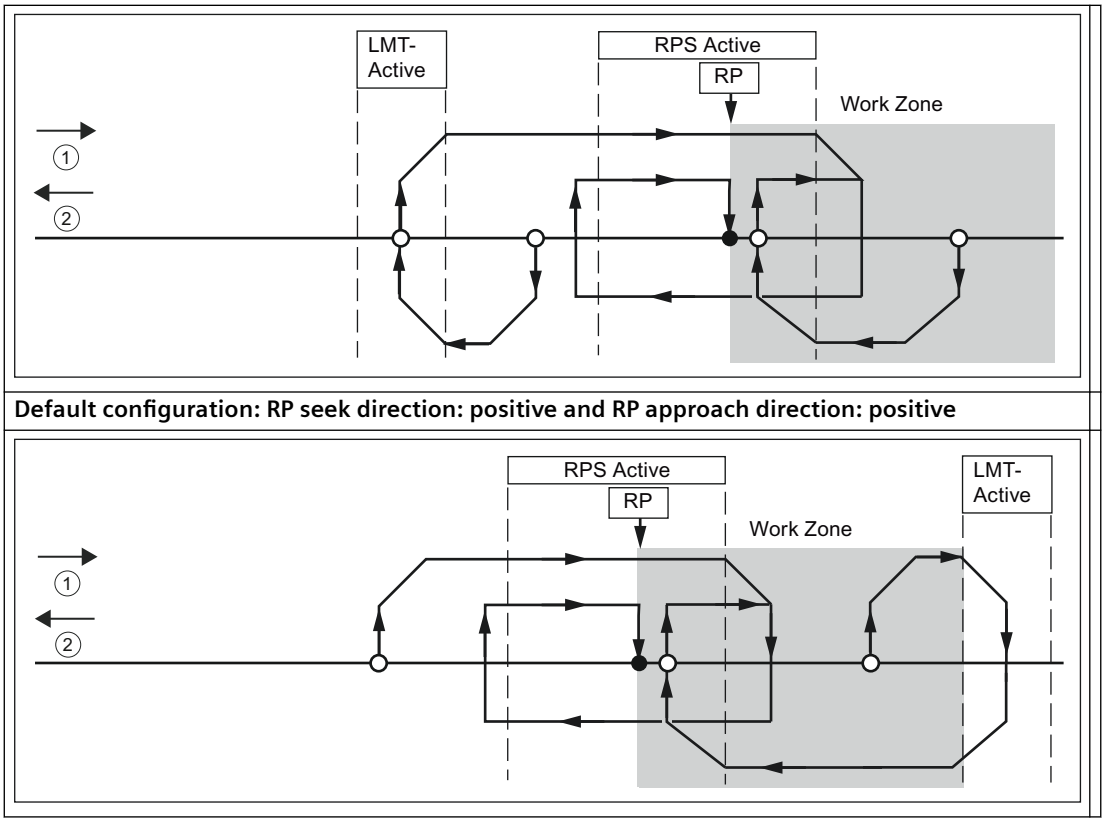
Default configuration: RP seek direction: positive and RP approach direction: positive



①: Positive motion
②: Negative motion

RP seek mode 2

Default configuration: RP seek direction: negative and RP approach direction: positive

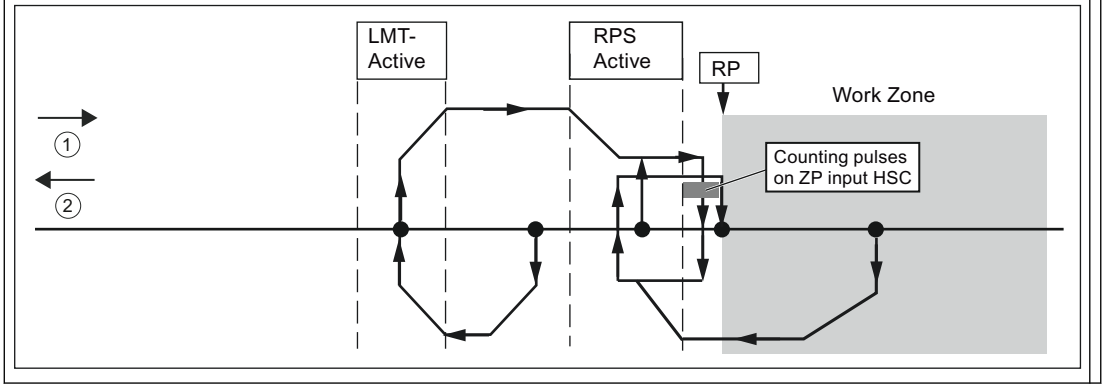


Default configuration: RP seek direction: positive and RP approach direction: positive

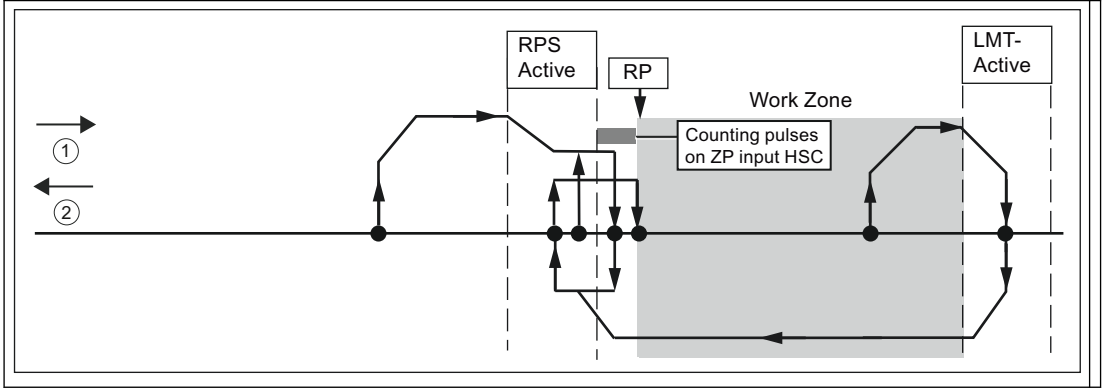
①: Positive motion
②: Negative motion

RP seek mode 3

Default configuration: RP seek direction: negative and RP approach direction: positive



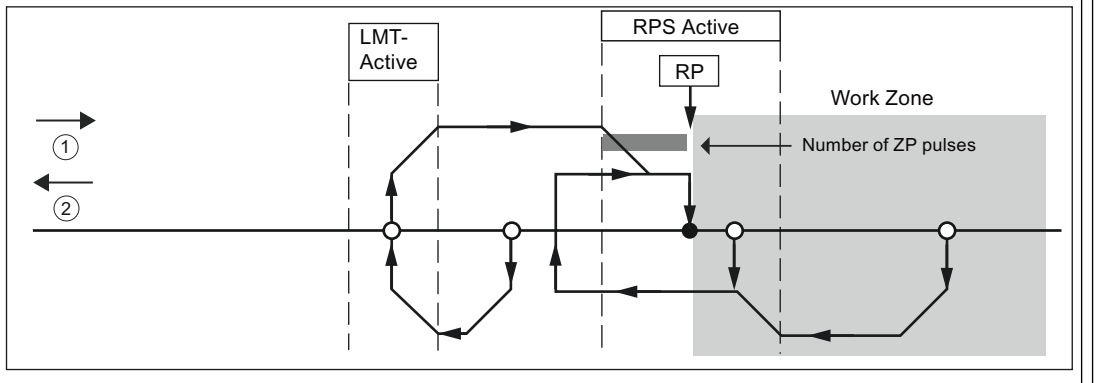
Default configuration: RP seek direction: positive and RP approach direction: positive



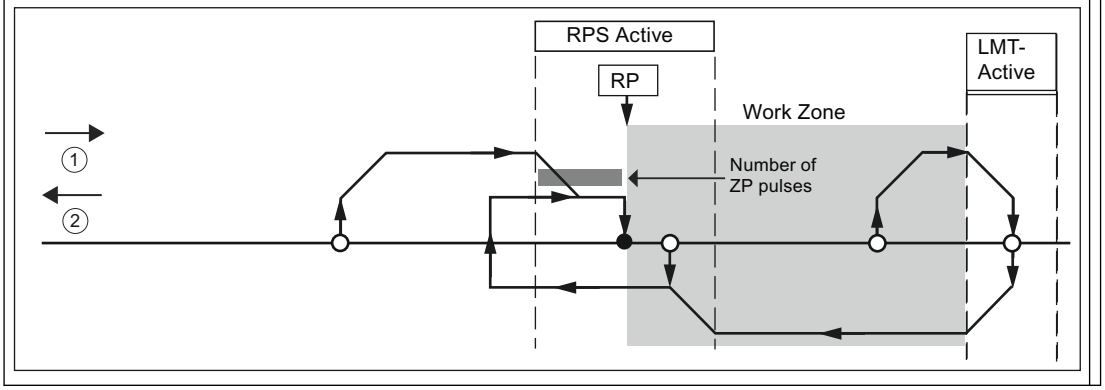
①: Positive motion
②: Negative motion

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



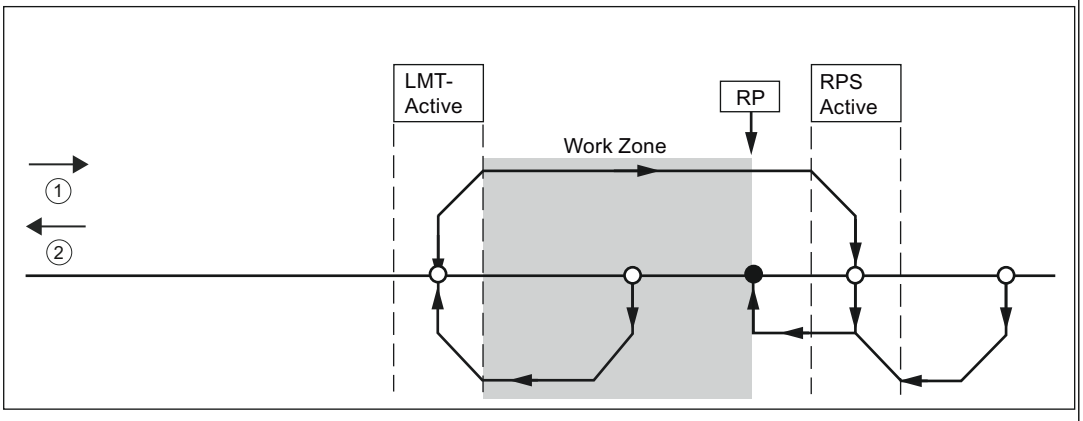
①: Positive motion
②: Negative motion

15.4.8.1 Selecting the work zone location to eliminate backlash

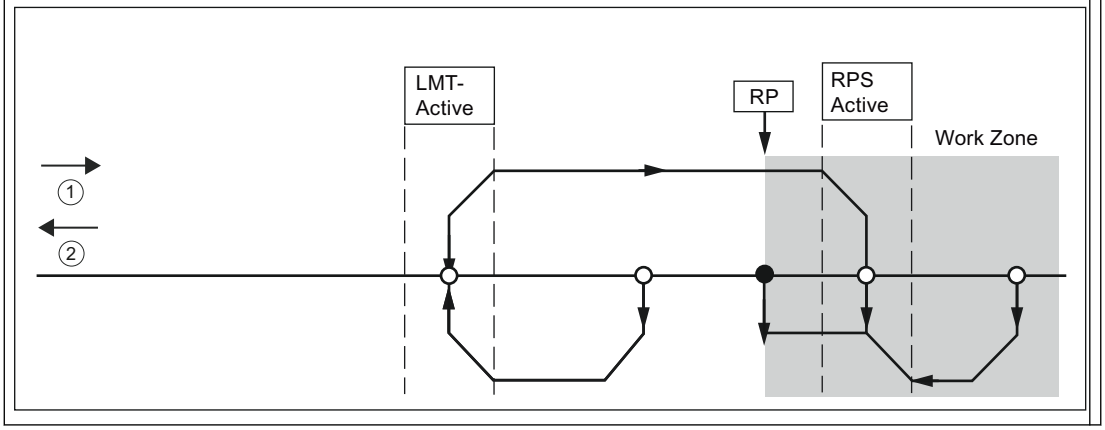
The following figure shows the work zone in relationship to the reference point (RP), the RPS Active zone, and the limit switches (LMT+ and LMT-) for an approach direction that eliminates the backlash. The second part of the illustration places the work zone so that the backlash is not eliminated. The 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

15.5 Using Axes group of Motion

15.5.1 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 1115) 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 1122) 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.

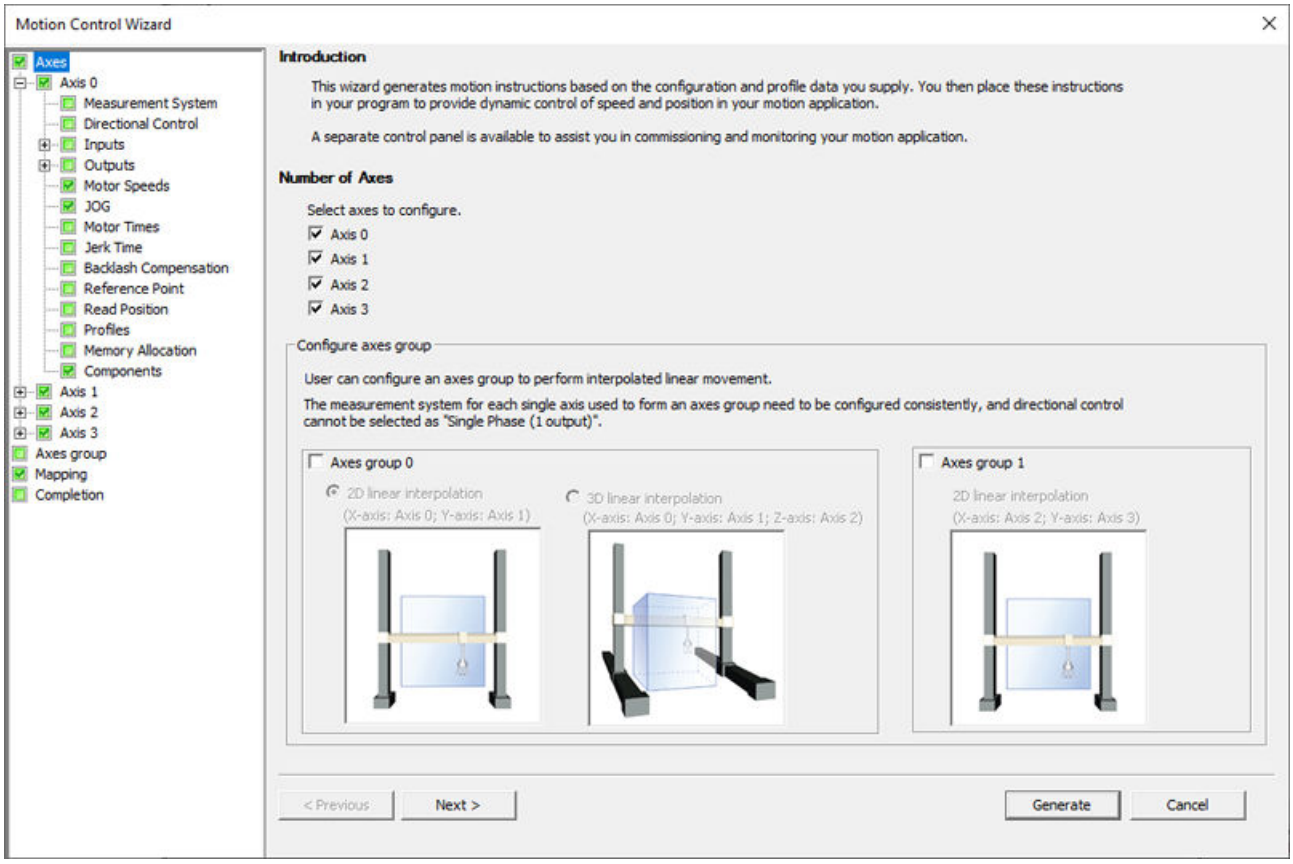
Note

Make sure to match the measurement system configuration to the pulses/revolution or distance/revolution specifications of your stepper/servo motor controller system.

15.5.2 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 1049) 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.

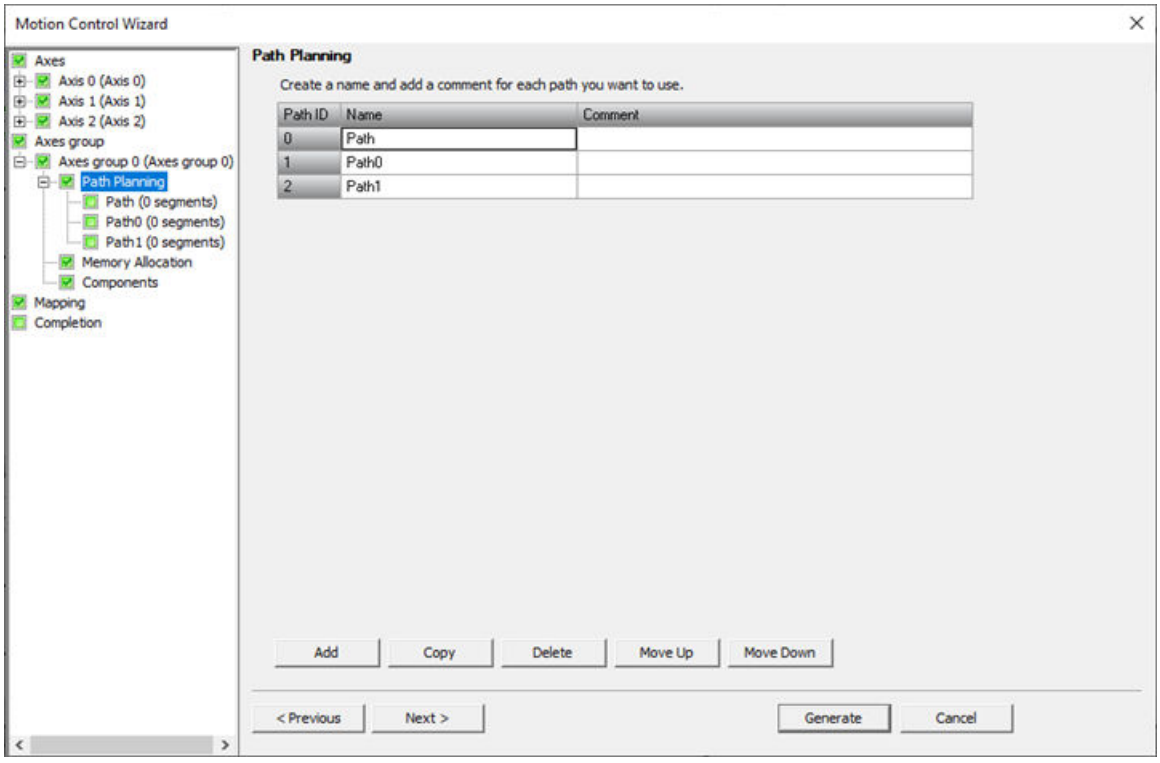
A path is a pre-defined motion description of a space curve that the tool center point (TCP) of an axes group moves along. It consists of a set of continuous positions and orientation information in 2/3D axes group movement space.

Note

The CPU only supports absolute position linear interpolation movement in path planning function for axes group..

Follow below steps to plan path:

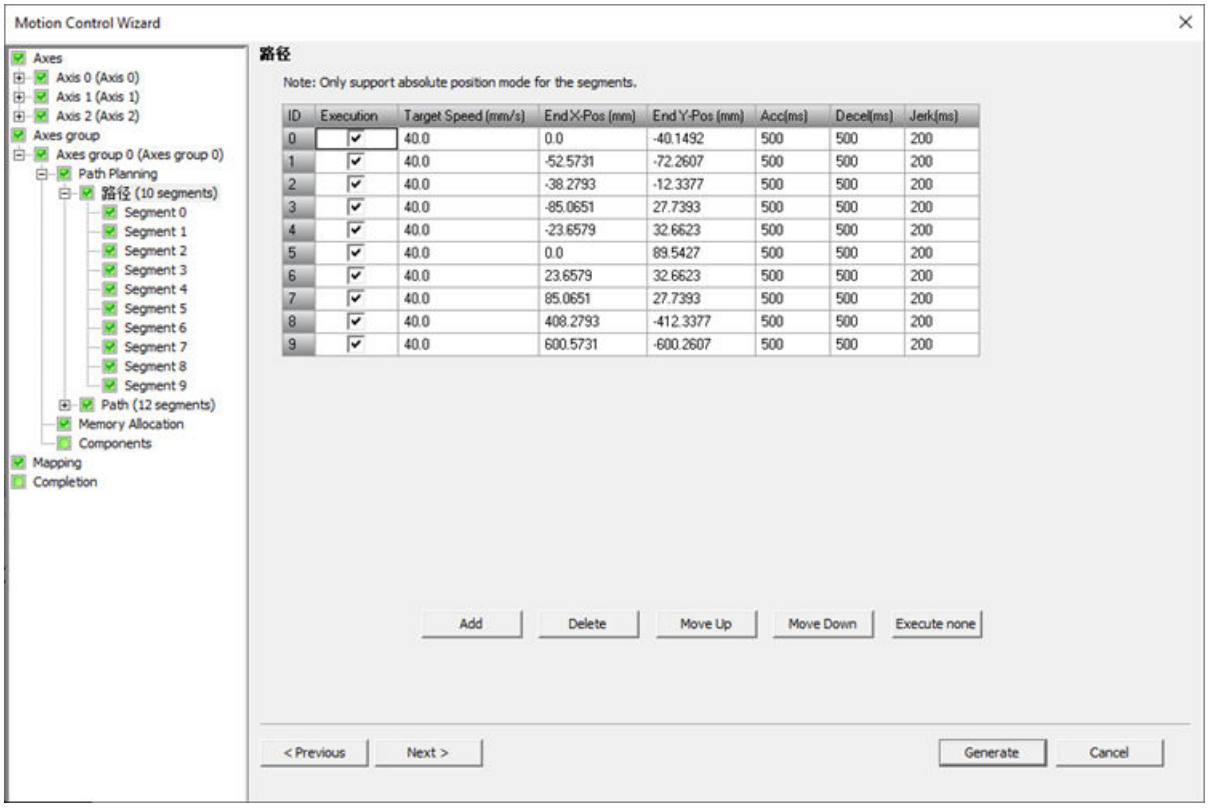
1. Add paths according to the linear interpolation movement you want the axes group to run. You can create a name and add a comment for each path. You can add 128 paths at most.



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.

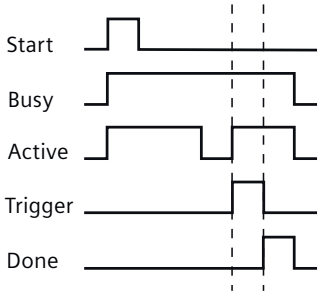
   – Aborting: the next function block controls an ongoing segment to decelerate to stop and controls the new segment to accelerate to full execution.

   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 1125) 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.

15.5.3 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 1123)	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 1125)	Command an axes group to move according to the path specified in multi-segment configuration
GRPx_RESET (Page 1126)	Command an axes group to transit from ErrorStop state to the state of getting ready for linear interpolation movement.
GRPx_STOP (Page 1128)	Command an axes group to stop the movement.

15.5.3.1 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 1126) to the state of getting ready for linear interpolation movement.

15.5.3.2 GRPx_2/3D_MOVELINEAR subroutine

Table 15-50 GRPx_2/3D_MOVELINEAR

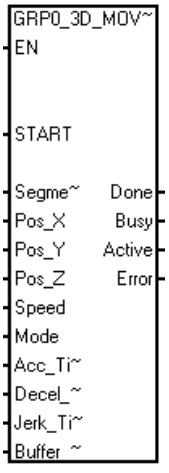
LAD / FBD	STL	Description
	**CALL GRPx_2D_MOVELINEAR, GRPx_3D_MOVELINEAR**	The GRPx_3D_MOVELINEAR subroutine commands the configured axes group on the linear interpolation movement from the actual position of the TCP (Tool Center point) to a target absolute or relative position. For axes group configured with 2 singles axes, the CPU will generate GRPx_2D_MOVELINEAR subroutine instruction.

Table 15-51 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

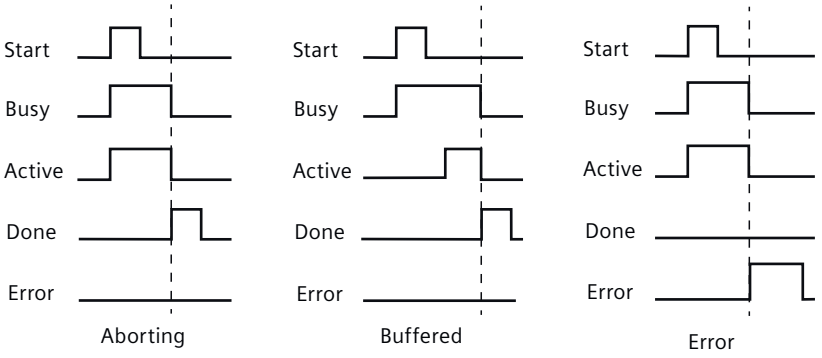
Use edge signal as the trigger signal for the linear interpolation movement subroutine execution to ensure the same linear interpolation movement subroutine is triggered only once.

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.

15.5.3.3 GRPx_MOVEPATH subroutine

Table 15-52 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 15-53 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.

15.5.3.4 GRPx_RESET subroutine

Table 15-54 GRPx_MOVEPATH

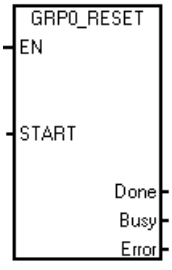
LAD / FBD	STL	Description
	`CALL GRPx_RESET`	The GRPx_RESET subroutine commands an axes group to transit from ErrorStop state to the state of getting ready for linear interpolation movement.

Table 15-55 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.

15.5.3.5 GRPx_STOP subroutine

Table 15-56 GRPx_STOP

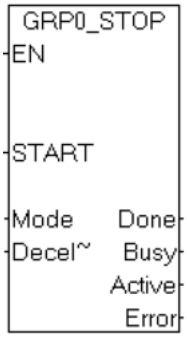
LAD/FBD	STL	Description
	**Call GRPx_STOP, START, Mode, Decel_Time, Done, Busy, Active, Error**	The **GRPx_STOP** subroutine commands an axes group to stop the movement.

Table 15-57 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.

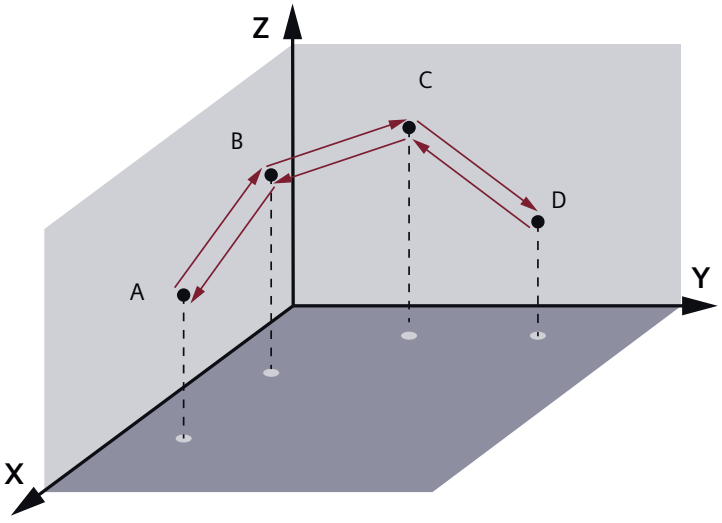
Note

After triggering the GRPx_STOP command, the axes group stops in the straight line where the current movement is.

15.5.4 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.

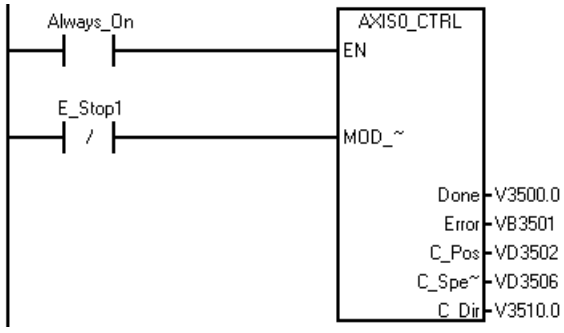
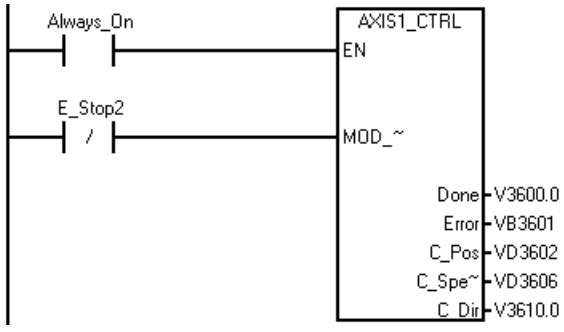
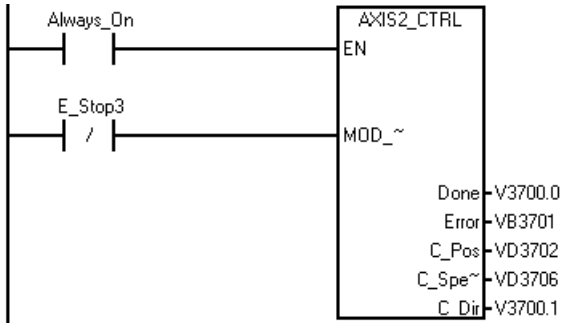
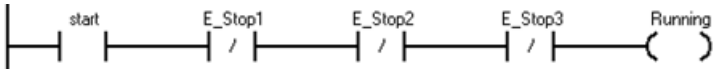
Below is the moving path of the tong on the pick-and-place machine:

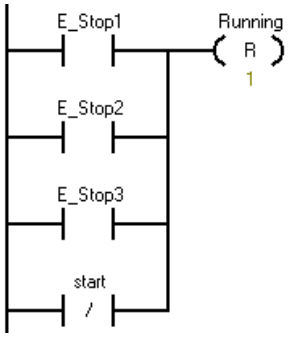
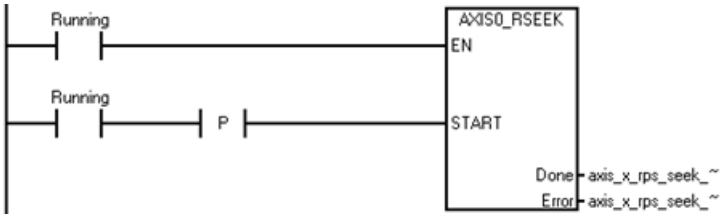
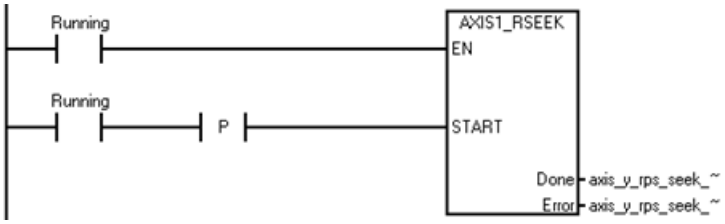
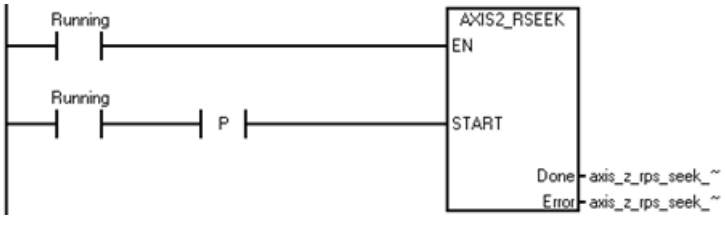


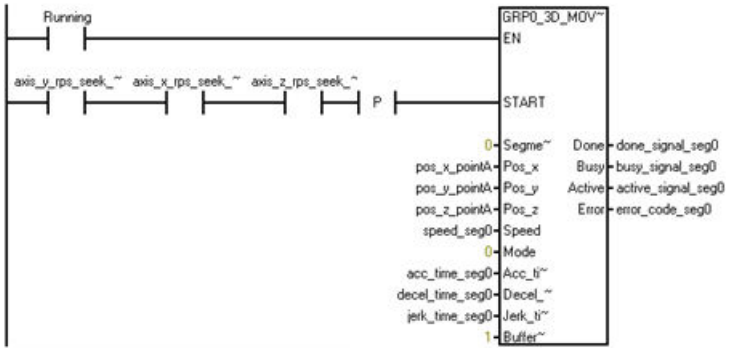
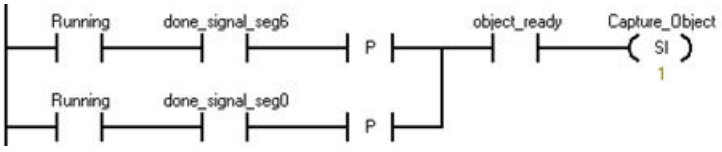
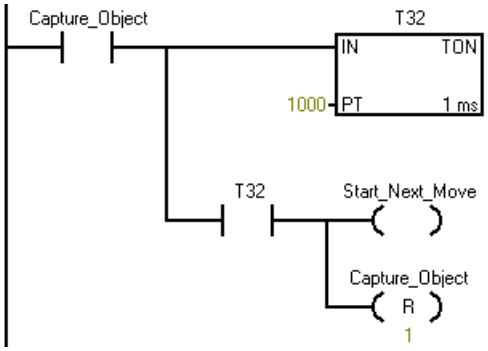
When the machine starts, the system seeks for zero position point first, and the tong moves to the pick position A. The system waits for the signal I0.1 that indicates the capture object is ready. The tong picks up the object by turning on Q1.1 (turn on for 1 second) and follow the track 1, 2 and 3 to position D. When the tong reaches to position D, it releases the object by

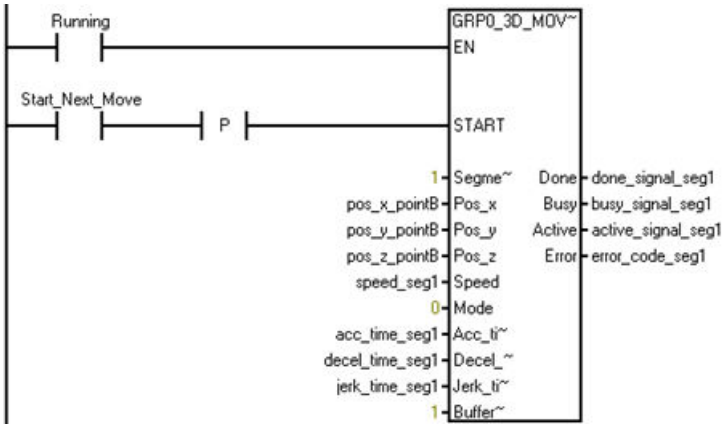
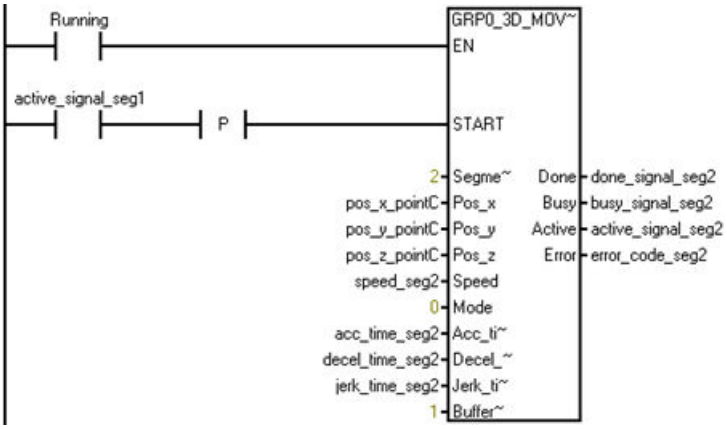
turning on Q1.2 (turn on for 1 second). The tong moves back to position A by following the track 4, 5 and 6 and waits for next capture.

Table 15-58 Example: Axes group of Motion simple absolute move (pick-and-place application)

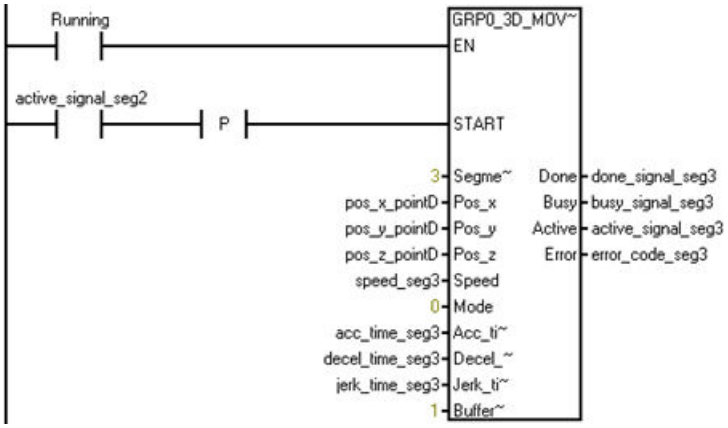
LAD/FBD	Description	STL
Network 1: 	Initialize Axis X.	**LD Always_On** **= L60.0** **LDN E_Stop1** **= L63.7** **LD L60.0** **CALL AXIS0_CTRL,** **L63.7, V3500.0,** **VB3501, VD3502,** **VD3506, V3510.0**
Network 2: 	Initialize Axis Y.	**LD Always_On** **= L60.0** **LDN E_Stop2** **= L63.7** **LD L60.0** **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**

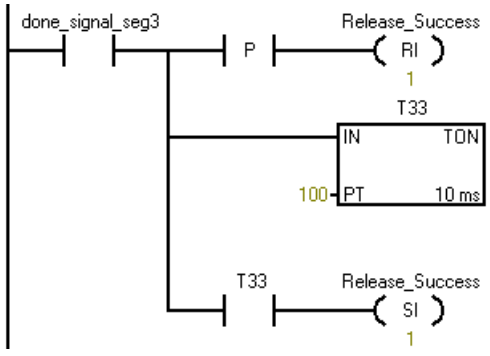
LAD/FBD	Description	STL
Network 5: 	When the start signal is OFF and the start signal is ON, the system stops.	**LD start** **O E_Stop1** **O E_Stop2** **ON start** **= Running, 1**
Network 6: 	When the start signal is ON, the system starts to seek zero position. This network is used for Axis X to seek 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**

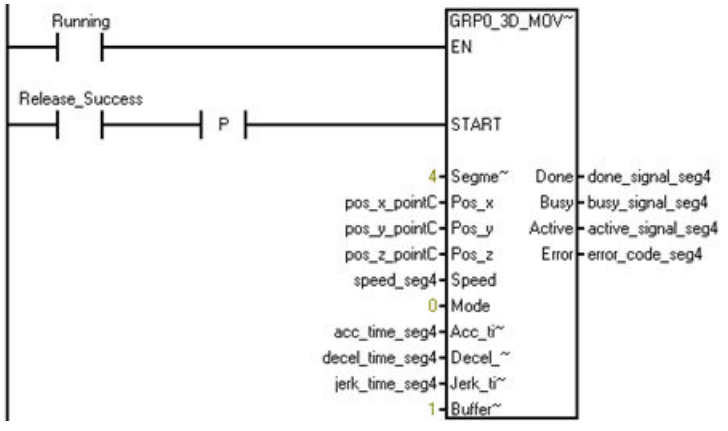
LAD/FBD	Description	STL
Network 9: 	When Axis X, Axis Y and Axis Z return to the zero position, the tong moves to the position over point A.	**LD Running** **= L60.0** **LD** **axis_y_rps_seek_done** **A** **axis_x_rps_seek_done** **A** **axis_z_rps_seek_done** **EU** **= L63.7** **LD L60.0** **CALL** **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**

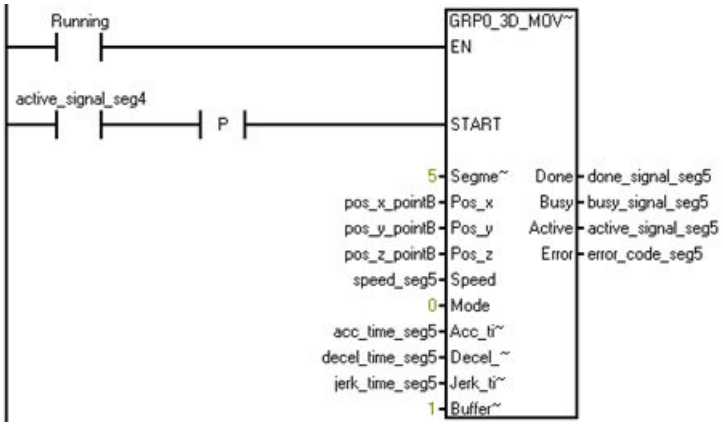
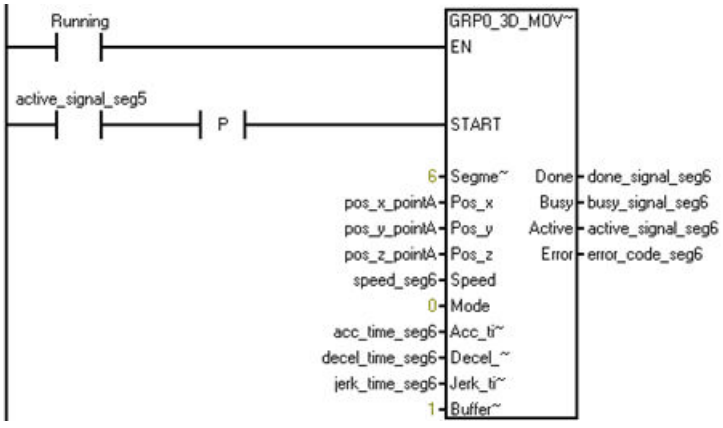
LAD/FBD	Description	STL
Network 12: 	The tong moves from point A to point B in the 3D space after the picking is finished. The moving path number equals to the Segment_ID, 1.	**LD Running** **= L60.0** **LD Start_Next_Move** **EU** **= L63.7** **LD L60.0** **CALL** **GRP0_3D_MOVELINEAR,** **L63.7, 1,** **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`

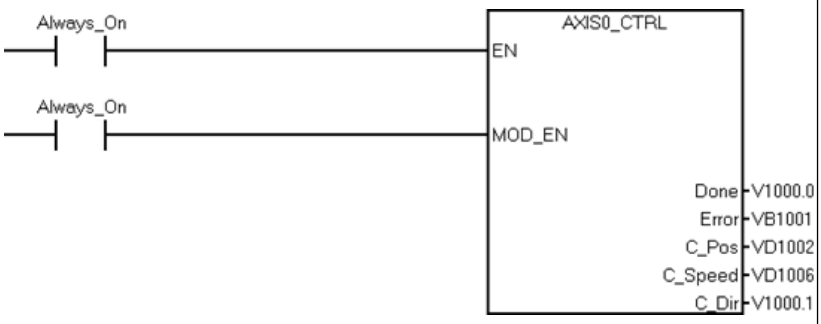
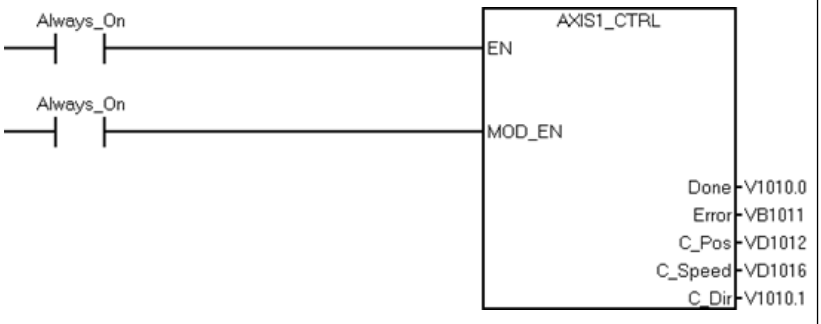
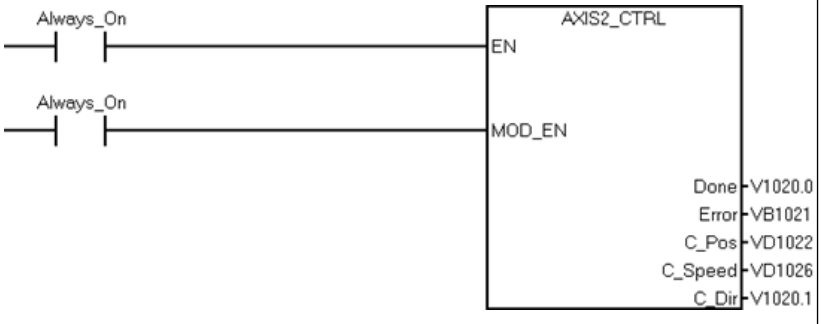
Network 16:
 
	The tong moves from point D to point C in the 3D space after the placing is finished. The moving path number equals to the Segment_ID, 4.	`LD Running`
		`= L60.0`
		`LD Release_Success`
		`EU`
		`= L63.7`
		`LD L60.0`
		`CALL GRP0_3D_MOVELINEAR, L63.7, 4, pos_x_pointC, pos_y_pointC, pos_z_pointC, 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`

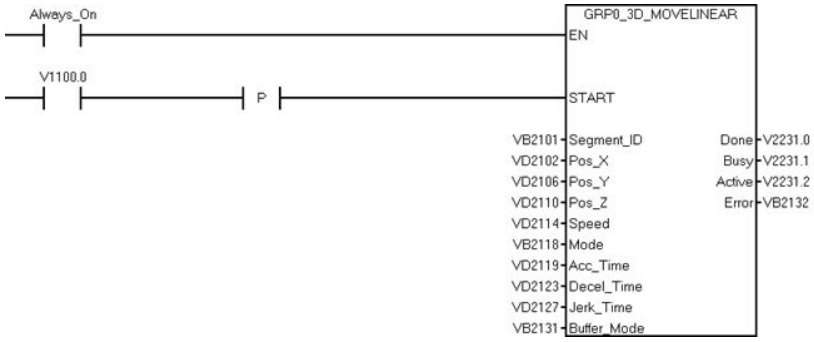
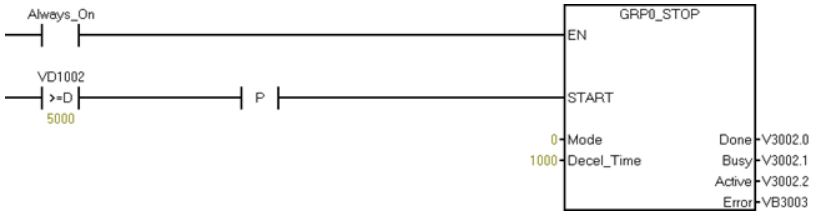
LAD/FBD	Description	STL
Network 17: 	The tong moves from point C to point B in the 3D space. The moving path number equals to the Segment_ID, 5.	**LD Running** **= L60.0** **LD** **active_signal_seg4** **EU** **= L63.7** **LD L60.0** **CALL** **GRP0_3D_MOVELINEAR,** **L63.7, 5,** **pos_x_pointB,** **pos_y_pointB,** **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**

15.5.4.1 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 15-59 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_move-ment:V1100.0** **EU** **= L63.7** **LD L60.0** **CALL GRP0_3D_MOVELIN-EAR:SBR34, L63.7, seg-ment: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 pul-ses, the command GPRO-STOP triggers a deceler-ated stop. The decel-eration 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_ac-tive:V3002.2, stp_er-ror:VB3003**

15.6 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 1049) 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 1064) in the AXISx_CTRL instruction to abort the axes group motion instruction under execution.
- Use the GRPx_STOP (Page 1128)subroutine.

15.7 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

15.8 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.

15.9 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 1064) 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 1070) or AXISx_LDPOS (Page 1072) 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 1123) or GRP0_MOVEPATH (Page 1125) to command the CPU for axes group movement.

Technical specifications

A

A.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.

- 2014/30/EU "Electromagnetic compatibility" (EMC directive)
- 2011/65/EU "Restriction of the use of certain hazardous substances in electrical and electronic equipment" (RoHS Directive)

The EC Declaration of Conformity can be obtained from the following responsible body and is stored at the following address:

Siemens Aktiengesellschaft

DI FA

Gleiwitzer Straße 555

DE-90475 Nürnberg

The EC Declaration of Conformity can be found at Internet (https://support.industry.siemens.com/cs/ww/zh/ps/14031/cert?ct=444&ci=526).

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:

- Electromagnetic Compatibility Regulations 2016 (EMC)
- Regulations on the restriction of the use of certain hazardous substances in electrical and electronic equipment 2012 (RoHS)

The Declaration of Conformity can be obtained from the following responsible body and is stored at the following address:

Siemens plc
Princess Road
Manchester
M20 2UR
United Kingdom

The UK Declaration of Conformity can be found at Internet (https://support.industry.siemens.com/cs/ww/zh/ps/14031/cert?ct=444&ci=5596).

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	Air discharge voltage for all surfaces: ±2kV; ±4kV; ±8 kV Contact discharge voltage for exposed conductive surfaces: ±4 kV
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
EN 61000-4-5 Surge immunity	DC system ±1 kV common mode, ±0.5 kV differential mode
EN 61000-4-6 Immunity to conducted disturbances	150 kHz to 80 MHz, 10 V RMS, 80% AM at 1 kHz
EN 61000-4-29 Voltage dips	DC system (CPU module) 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

Electromagnetic compatibility - Conducted and radiated emissions per EN 61000-6-4		
Radiated Emissions EN 55011, Class A, Group 1	30 MHz to 230 MHz	<50dB (µV/m) quasi-peak; measured at 3 m
	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

Ambient 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, 5 cycles
EN 60068-2-31, 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 Conditions of operations

Environmental conditions - Operation	
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	SO_2: < 0.5 ppm; H_2S: < 0.1 ppm; RH < 60% non-condensing
EN 60068-2-14, Test Nb, temperature change	-20 °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

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

- Polution 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 voltage

Table A-7 Rated voltage

Module	Rated voltage	tolerance
For DC/DC/DC modules	24 V DC	20.4 V DC to 28.8 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

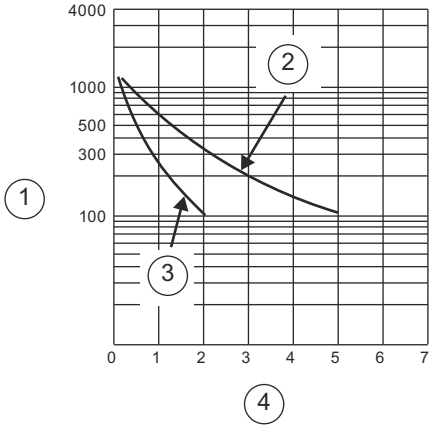
When a mechanical contact turns on output power to the S7-200 SMART CPU, or any digital expansion module, it sends a "1" signal to the digital outputs for approximately 150 microseconds.

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 CPU

A.2.1 CPU ST32

A.2.1.1 General specifications and features

General specifications and features of CPU ST32

Table A-8 General specifications

Technical data	**CPU ST32 DC/DC/DC**
Article number	6ES7288-1ST32-0AA0
Dimensions W x H x D (mm)	100 x 118 x 75
Weight	305 g
Power consumption	24 W
Current available (EM bus)	Maximum 1680 mA(5 V DC)
Current available (24 V DC)	300 mA max. (sensor power)
Digital input current consumption (24 V DC)	**I0.0-I0.7:** • 10mA @24v • 6mA @5v **I1.0-1.7:** • 4mA @24v

Table A-9 CPU features

Technical data		**CPU ST32 (V3.2) DC/DC/DC**
User memory	Program	120 KB[1]
	User-defined area	896 KB
	User data (V)	100 KB
	Retentivity	40 KB max.[2]
On-board digital I/O		16 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 relay (S)		256 bits
Expansion modules expansion		max. 16
Signal board expansion		2 max.

Technical data		**CPU ST32 (V3.2) DC/DC/DC**
High-speed counter	total number	8
	Single phase	8 at 200 kHz
	A/B phase	8 at 200 kHz
Pulse outputs [3]		• Single phase: 5 at 200 kHz • A/B phase: 4 at 200 kHz
Pulse catch inputs		16
Cyclic interrupts		10 at 1 ms resolution including interrupt number 10 (timer interrupt 0), interrupt number 11 (timer interrupt 1), and 8 interrupts triggered by a 1ms timer (CT=PT)
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] The user program size for CPU ST32 (V3.1) is 100 KB.

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

[3] 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 ST32 (V3.2) DC/DC/DC**
Support Controller function for PROFINET communication	✓
Support I-Device function for PROFINET communication	✓
Maximum number of PROFINET device	10
Device number of PROFINET device	1 to 10
Maximum number of devices supported by PN IRT	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 send clock supported by PROFINET	1 ms
CPU address range of PROFINET process image input register	I128.0 to I1535.7
CPU address range of PROFINET process image output register	Q128.0 to Q1535.7
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

Description	CPU ST32 (V3.2) DC/DC/DC
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
CPU address of PROFINET process image input and output register for device #9 [1]	I1128.0 to I1407.7 Q1128.0 to Q1407.7
CPU address of PROFINET process image input and output register for device #10 [1]	I1408.0 to I1535.7 Q1408.0 to Q1535.7

[1] Not applicable to CPU ST32 (V3.1).

Table A-11 Performance

Instruction type	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
POU	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
Timer	Type/quantity	Non-retentive (TON, TOF): 320 Retentive (TONR): 192
Counter	Quantity	512

Table A-13 Communication

Technical data	**CPU ST32 DC/DC/DC**
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 transmission rate	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, 707 V DC; RS485 signal to CPU logic common, 707 V DC
Cable type	PROFINET (LAN): CAT5e shielded cable; RS485: PROFIBUS network cable
PROFINET Communication	
PROFINET controller	✓
PROFINET device	✓
PROFINET controller	
Services	
PG/OP communication	✓
S7 routing	✓
Isochronous mode	✓
Open IE communication	✓
IRT	✓
MRP	-
PROFIenergy	-
Max. number of PROFINET devices that you can connect for RT	10
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.
Send clock	1 ms to 4 ms

Table A-14 Power supply

Technical data	**CPU ST32 DC/DC/DC**
Voltage range	20.4 to 28.8 V DC [1]
Mains frequency	--

Technical data		CPU ST32 DC/DC/DC
Input current (max. load)	CPU only	120 mA at 24 V DC (without driving 300 mA sensor power) 420 mA at 24 V DC (with driving 300 mA sensor power)
	CPU with all expansion accessories	1 A at 24 V DC
Hold up time (loss of power)		10 ms at 24 V DC
Inrush current (max.)		15 A at 28.8 V DC
Isolation (input power to logic)		--
Ground leakage, AC line to functional earth		--
Internal fuse, not user replaceable		3.5 A, 125 V, fast-acting fuse

[1] The rise time of the power supply voltage from 0 V to the normal operating voltage range must not exceed 1.5 seconds.

Table A-15 Sensor Supply

Technical data	CPU ST32 DC/DC/DC
Voltage range	19.2 to 28.8 V DC
Output current rating (max.)	300 mA
Maximum ripple noise (<10 MHz)	< 1 V peak to peak
Isolation (CPU logic to sensor power)	Not isolated

A.2.1.2 Digital inputs and outputs

Table A-16 Digital inputs

Technical data	CPU ST32 DC/DC/DC
Number of inputs	16
Type	Sink/Source (IEC Type 1 sink, except I0.0 to I0.7)
Rated voltage	**I0.0-I0.7:** • 10mA @24v • 6mA @5v **I1.0-1.7:** • 4mA @24v
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 6 mA • Other input points: 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
Isolation (field side to logic)	707 V DC for 1 minute
Isolation groups	2

Technical data	CPU ST32 DC/DC/DC
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
HSC clock input rates	Single phase: 8 HSCs at 200 kHz; A/B phase: 8 HSCs at 200 kHz;
Number of inputs on simultaneously	16
Cable length (max.), in meters	I0.0 to I1.7: Shielded (only): • 50 m HSC (high-speed) inputs
	Non-HSC usage: Shielded: • 500 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
- If the level transition speed of the input signal is slow, it is recommended to add a pull-down or pull-up resistor on the input side to accelerate the signal edge
- It is recommended to use twisted-pair cables, as pairing the signal line with the return line by twisting them together can effectively reduce signal crosstalk

Table A-17 Digital outputs

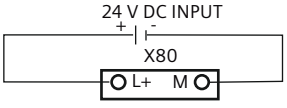
Technical data	CPU ST32 DC/DC/DC
Number of outputs	16
Type	Solid state - MOSFET (sourcing)
Voltage range	20.4 to 28.8 V DC
Logic 1 signal at max. current	20 V DC min.
Logic 0 signal with 10 KΩ load	0.1 V DC max.
Rated current per point (max.)	0.5 A
Rated current per common (max.)	8 A
Lamp load	5 W
ON state resistance	0.4 Ω max.
Leakage current per point	10 µA max.
Surge current	8 A for 100 ms max.
Overload protection	-
Isolation (field side to logic)	707 V DC for 1 minute

Technical data	CPU ST32 DC/DC/DC
Isolation resistance	--
Isolation between open contacts	--
Isolation groups	1
Inductive clamp voltage	L+ minus 48 V DC, 1 W dissipation
Switching delay (Qa.0 to Qb.7)	0.5 µs max., off to on 2 µs max., on to off
Maximum relay switching frequency	--
Lifetime mechanical (no load)	--
Lifetime contacts at rated load	--
Output state in STOP mode	Last value or substitute value (default value 0)
Number of outputs on simultaneously	16
Cable length (max.), in meters	Shielded: 500 m Unshielded: 150 m

A.2.1.3 Wiring diagram

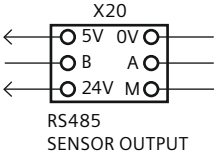
The wiring diagram and pin connector locations are shown below:

Connector pin assignments for CPU ST32 DC/DC/DC (6ES7288-1ST32-0AA0)



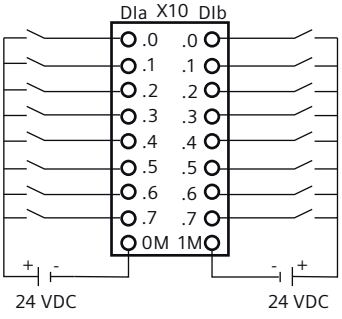
X80: DC power supply terminal

DC power supply	L+	M	
Pin number			



X20: RS485, sensor output

DC power supply	5 V	0 V	
Signals	B	A	
Sensor power supply output	24 V	M	



X10: DC input terminal

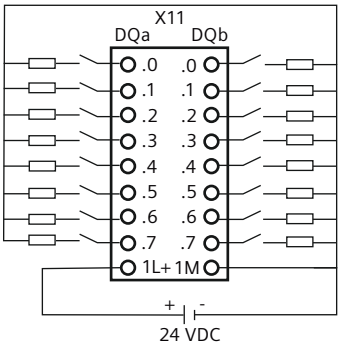
Signals	DI a.0	DI a.1	DI a.2	DI a.3	DI a.4	DI a.5	DI a.6	DI a.7	0M
Pin number	1	3	5	7	9	11	13	15	17
Pin number	2	4	6	8	10	12	14	16	18
Signals	DI b.0	DI b.1	DI b.2	DI b.3	DI b.4	DI b.5	DI b.6	DI b.7	1M

Both group a and group b support sourcing and sinking wiring methods, without affecting each other

For sinking inputs, connect 24 VDC "-" to "M" (shown).

For sourcing inputs, connect 24 VDC "+" to "M"

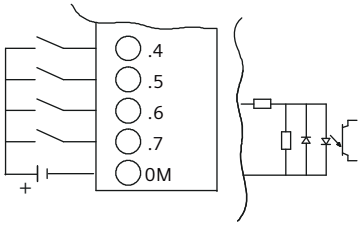
Connector pin assignments for CPU ST32 DC/DC/DC (6ES7288-1ST32-0AA0)



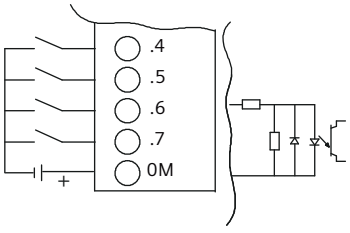
X11: DC outputs terminal

Signals	DQ a.0	DQ a.1	DQ a.2	DQ a.3	DQ a.4	DQ a.5	DQ a.6	DQ a.7	L+
Pin number	1	3	5	7	9	11	13	15	17
Pin number	2	4	6	8	10	12	14	16	18
Signals	DQ b.0	DQ b.1	DQ b.2	DQ b.3	DQ b.4	DQ b.5	DQ b.6	DQ b.7	M

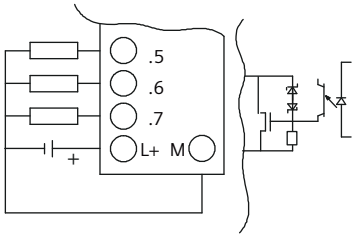
24 V DC Input
Used as Sinking Inputs



24 V DC Input
Used as Sourcing Inputs



24 V DC Output
(Sourcing)



A.3 Digital inputs and outputs expansion modules (EMs)

A.3.1 EM DE16 digital input specifications

Table A-18 General specifications

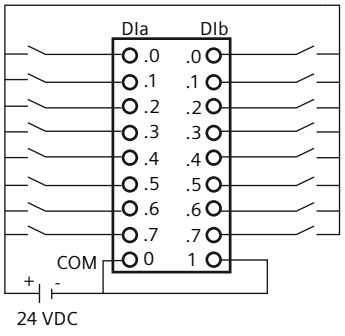
Model	EM Digital 16 x Inputs (EM DE16)
Part number	6ES7288-2DE16-0AA1
Dimensions W x H x D (mm)	15 x 118 x 74.5
Weight	78 g
Power consumption	2.2 W

Model	EM Digital 16 x Inputs (EM DE16)
Current consumption (SM Bus)	80 mA
Current consumption (24 V DC)	4 mA/input used

Table A-19 Digital inputs

Model	EM Digital 16 x Inputs (EM DE16)
Number of inputs	16
Type	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.)	15 V DC at 2.5 mA
Logic 0 signal (max.)	5 V DC at 1 mA
Isolation (field side to logic)	707 V DC for 1 minute
Isolation groups	2
Filter times	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	16
Cable length (max.), in meters	Shielded: 500 m normal inputs Unshielded: 300 m normal inputs

EM DE16 16-point digital input (6ES7288-2DE16-0AA1) wiring diagram



DC input terminal									
Signals	DI a.0	DI a.1	DI a.2	DI a.3	DI a.4	DI a.5	DI a.6	DI a.7	0
Pin number	1	3	5	7	9	11	13	15	17
Pin number	2	4	6	8	10	12	14	16	18
Signals	DI b.0	DI b.1	DI b.2	DI b.3	DI b.4	DI b.5	DI b.6	DI b.7	1

A.3.2 EM QT16 digital output specifications

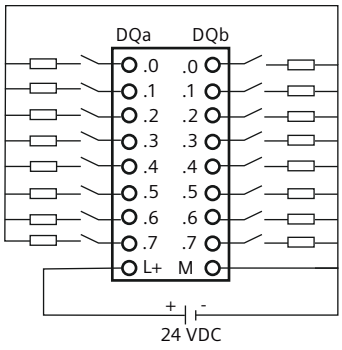
Table A-20 General specifications

Model	**EM Digital 16 x Outputs Transistor (EM QT16)**
Article number	6ES7288-2QT16-0AA1
Dimensions W x H x D (mm)	15 x 118 x 74.5
Weight	79 g
Power consumption	1.5 W
Current consumption (SM Bus)	80 mA
Current consumption (24 V DC)	Control circuit maximum 40 mA, L+ maximum 10 A

Table A-21 Digital outputs

Model	**EM Digital 16 x Outputs Transistor (EM QT16)**
Number of outputs	16
Type	Solid state - MOSFET (sourcing)
Voltage range	20.4 to 28.8 V DC
Logic 1 signal at max. current	20 V DC
Logic 0 signal with 10 KΩ load	0.1 V DC
Rated current per point (max.)	0.75 A
Rated current per common (max.)	10 A
Lamp load	5 W
ON state contact resistance	0.6 Ω max.
Leakage current per point	10 µA
Surge current	-
Overload protection	Yes
Isolation (field side to logic)	707 V DC for 1 minute
Isolation resistance	-
Isolation between open contacts	-
Isolation groups	1
Inductive clamp voltage	L+ minus 52 V, 1 W dissipation
Switching delay	Switch on less than 100 µs and switch off less than 100 µs
Output behavior in STOP	Last value or substitute value (default value 0)
Number of outputs on simultaneously	16
Cable length (max.), in meters	Shielded: 500 m Unshielded: 150 m

EM QT16 Digital 16 x Outputs Transistor (6ES7288-2QT16-0AA1)



DC outputs terminal									
Signals	DQ a.0	DQ a.1	DQ a.2	DQ a.3	DQ a.4	DQ a.5	DQ a.6	DQ a.7	L+
Pin number	1	3	5	7	9	11	13	15	17
Pin number	2	4	6	8	10	12	14	16	18
Signals	DQ b.0	DQ b.1	DQ b.2	DQ b.3	DQ b.4	DQ b.5	DQ b.6	DQ b.7	M

A.4 Analog inputs and outputs expansion modules (EMs)

A.4.1 EM AE04 and EM AE08 analog input specifications

Table A-22 General specifications

Model	EM Analog 4 x Inputs (EM AE04)	EM Analog 8 x Inputs (EM AE08)
Article number	6ES7288-3AE04-0AA1	6ES7288-3AE08-0AA1
Dimensions W x H x D (mm)	15 x 118 x 74.5	15 x 118 x 74.5
Weight	81 g	83 g
Power consumption	1.3 W	1.9 W
Current consumption (SM Bus)	70 mA	80 mA
Current consumption (24 V DC)	40 mA	60 mA

Table A-23 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 input), optionally grouped in pairs	Voltage or current (differential input), optionally grouped in pairs
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 (Refer to Analog input representation for voltage and Analog input representation for current (Page 1165).)	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 1165).)

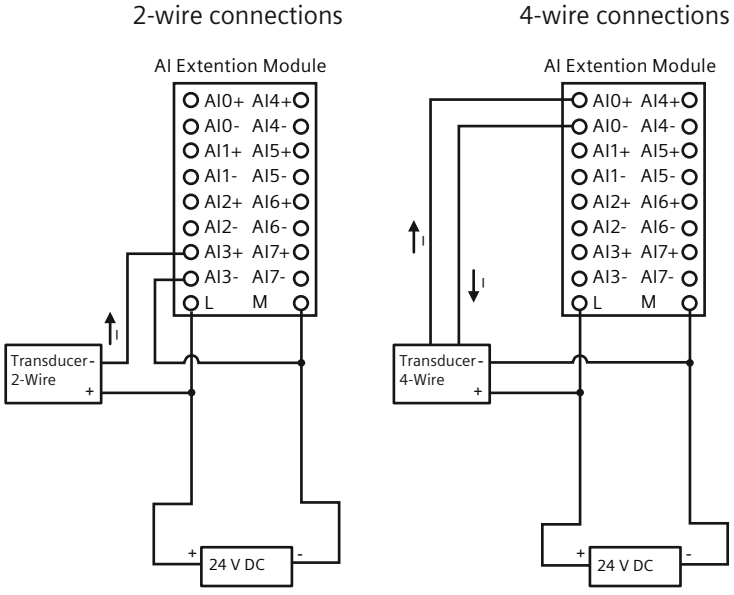
Model	EM Analog 4 x Inputs (EM AE04)	EM Analog 8 x Inputs (EM AE08)
Overflow/underflow (data word)	Voltage: 32,512 to 32,767 / -32,513 to -32,768 Current: 32,512 to 32,767 / -4,865 to -32,768 (Refer to Analog input representation for voltage and Analog input representation for current (Page 1165).)	Voltage: 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 1165).)
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 (Refer to Analog input response times for step response time.) (Page 1165)	None, weak, medium or strong (Refer to Analog input response times for step response time.) (Page 1165)
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
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 shielded cable	100 m shielded cable

Table A-24 Diagnostics

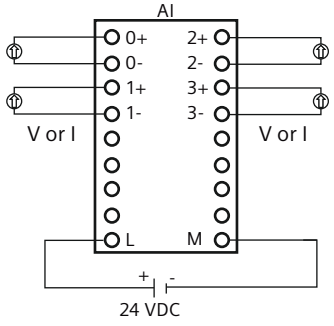
Model	EM Analog 4 x Inputs (EM AE04)	EM Analog 8 x Inputs (EM AE08)
Overflow/underflow	Yes	Yes
24 V DC low voltage	Yes	Yes

EM AE04 and EM AE08 wiring current transducers

The wiring current transducers can be used as both 2-wire and 4-wire transducers, as shown in the diagram below.

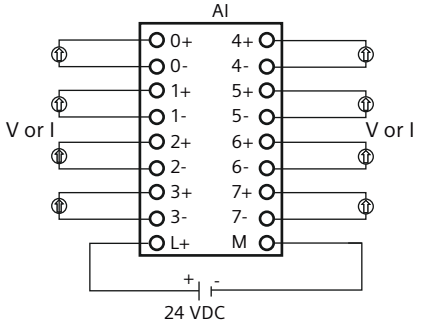


EM AE04 Analog 4 x Inputs (6ES7288-3AE04-0AA1)



DC input terminal									
Signals	AI 0+	AI 0-	AI 1+	AI 1-					L+
Pin number	1	3	5	7	9	11	13	15	17
Pin number	2	4	6	8	10	12	14	16	18
Signals	AI 2+	AI 2-	AI 3+	AI 3-					M

EM AE08 Analog 8 x Inputs (6ES7288-3AE08-0AA1)



DC input terminal									
Signals	AI 0+	AI 0-	AI 1+	AI 1-	AI 2+	AI 2-	AI 3+	AI 3-	L+
Pin number	1	3	5	7	9	11	13	15	17
Pin number	2	4	6	8	10	12	14	16	18
Signals	AI 4+	AI 4-	AI 5+	AI 5-	AI 6+	AI 6-	AI 7+	AI 7-	M

Note

Unused analog inputs should be shorted.

A.4.2 EM AQ04 Analog output module specifications

Table A-25 General specifications

Technical data	EM Analog 4 x Outputs (EM AQ04)
Article number	6ES7288-3AQ04-0AA1
Dimensions W x H x D (mm)	15 x 118 x 74.5
Weight	79 g
Power consumption	1.3 W (no load)
Current consumption (SM Bus)	60 mA
Current consumption (24 V DC)	40 mA (no load)
	120 mA (20 mA load per channel)

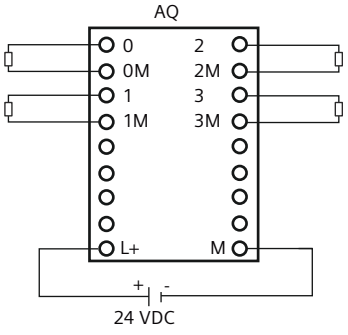
Table A-26 Analog outputs

Technical data	EM Analog 4 x Outputs (EM AQ04)
Number of outputs	4
Type	Voltage or current
Range	±10 V or 0 to 20 mA
Resolution	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 (Refer to the output ranges for voltage and current (Page 1166).)
Accuracy (25 °C / -20 to 60 °C)	±0.5% / ±1.0% of full scale
Settling time (95% of new value)	Voltage: 300 µs (R), 750 µs (1 uF) Current: 600 µs (1 mH), 2 ms (10 mH)
Load impedance	Voltage: ≥ 1000 Ω Current: ≤ 600 Ω
Output behavior in STOP	Last value or substitute value (default value 0)
Isolation (field side to logic)	None
Cable length (max.), in meters	100 m twisted and shielded

Table A-27 Diagnostics

Technical data	EM Analog 4 x Outputs (EM AQ04)
Overflow/underflow	Yes
Short to ground (voltage mode only)	Yes
Wire break (current mode only)	Yes
24 V DC low voltage	Yes

EM AQ04 Analog 4 x Outputs (6ES7288-3AQ04-0AA1)



DC input terminal

Signals	AQ 0	AQ 0M	AQ1	AQ1 M					L+
Pin number	1	3	5	7	9	11	13	15	17
Pin number	2	4	6	8	10	12	14	16	18
Signals	AQ 2	AQ 2M	AQ3	AQ3 M					M

Note

Leave unused channels in default voltage mode (0V voltage output), with no wiring required. If an unused channel is set to current mode (0mA current output), short-circuit the output; otherwise, the channel will report an open-circuit alarm.

AQ04 module current mode derating trend chart

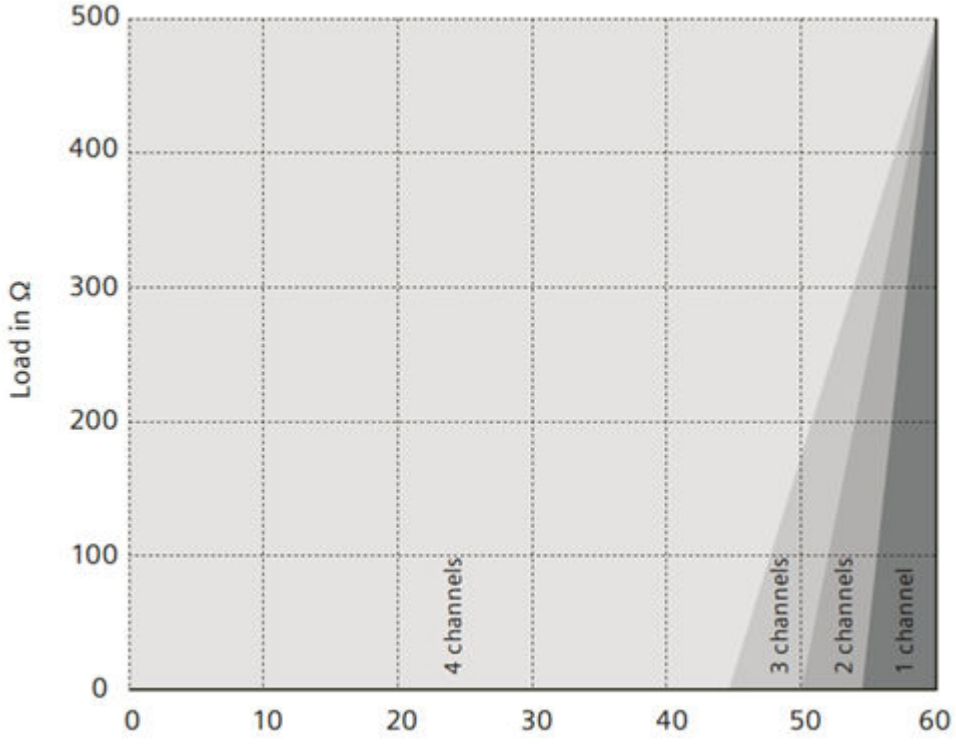


Figure A-1 Horizontal installation, ambient temperature (unit: ℃)

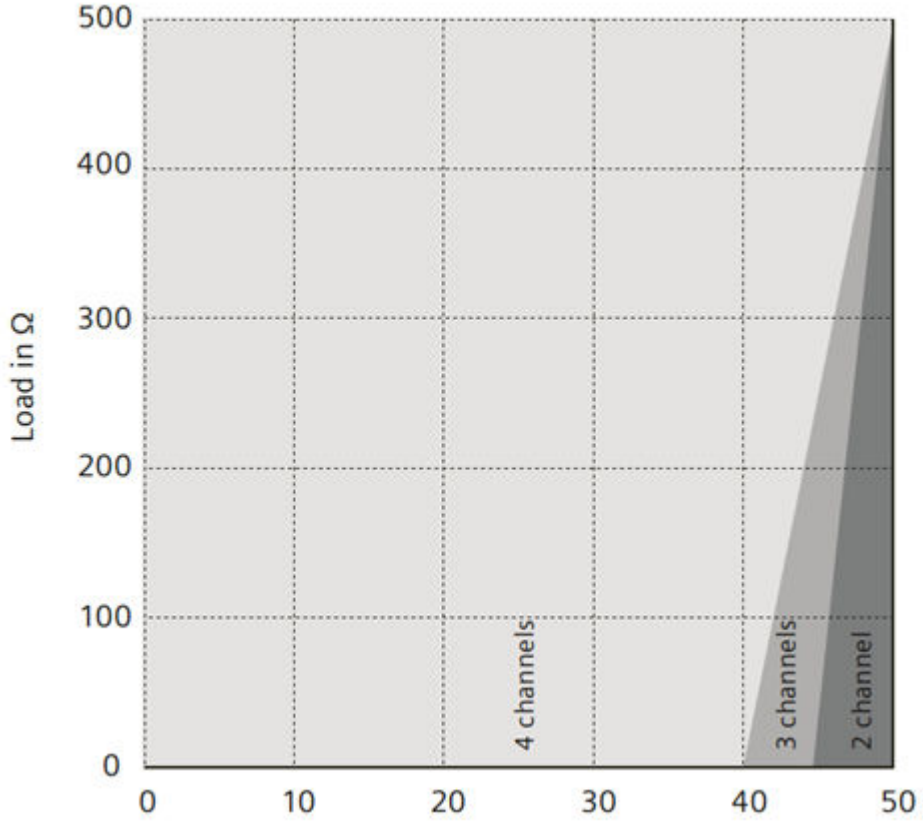


Figure A-2 Vertical installation, ambient temperature (unit: ℃)

A.4.3 EM AM06 analog input/output module specifications

Table A-28 General specifications

Technical data	EM 4 x Analog Inputs / 2 x Analog Outputs (AM06)
Article number	6ES7288-3AM06-0AA1
Dimensions W x H x D (mm)	15 x 118 x 74.5
Weight	82 g
Power consumption	1.5 W (no load)
Current consumption (SM Bus)	60 mA
Current consumption (24 V DC)	50 mA (no load)
	90 mA (20 mA load per channel)

Table A-29 Analog inputs

Model	**EM 4 x Analog Inputs / 2 x Analog Outputs (AM06)**
Number of inputs	4
Type	Voltage or current (differential): Selectable in groups of 2
Range	±10 V, ±5 V, ±2.5 V, or 0 to 20 mA
Full scale range (data word)	-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 (Refer to Analog input representation for voltage and Analog input representation for current (Page 1165).)
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 1165).)
Resolution	Voltage mode: 12 bits + sign Current mode: 12 bits
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 1165)
Noise rejection	400, 60, 50, or 10 Hz
Input impedance	≥1 M Ω (voltage) / 290 Ω (current)
Isolation (field side to logic)	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
Analog to digital conversion time	625 µs (400 Hz rejection)
Common mode rejection	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
Cable length (max.), in meters	100 m shielded cable

Table A-30 Analog outputs

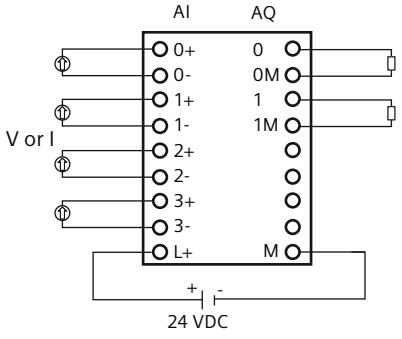
Technical data	**EM 4 x Analog Inputs / 2 x Analog Outputs (AM06)**
Number of outputs	2
Type	Voltage or current
Range	±10 V or 0 to 20 mA
Resolution	Voltage mode: 11 bits + sign Current mode: 11 bits
Full scale range (data word)	Voltage: -27,648 to 27,648 Current: 0 to 27,648
Accuracy (25 °C / -20 to 60 °C)	±0.5% / ±1.0% of full scale
Settling time (95% of new value)	Voltage: 300 µs (R), 750 µs (1 uF) Current: 600 µs (1 mH), 2 ms (10 mH)

Technical data	EM 4 x Analog Inputs / 2 x Analog Outputs (AM06)
Load impedance	Voltage: ≥ 1000 Ω Current: ≤ 00 Ω
Output behavior in STOP	Last value or substitute value (default value 0)
Isolation (field side to logic)	None
Cable length (max.), in meters	100 m shielded cable

Table A-31 Diagnostics

Model	EM 4 x Analog Inputs / 2 x Analog Outputs (AM06)
Overflow/underflow	✓
Short to ground (voltage mode only)	✓
Wire break (current mode only)	✓
24 V DC low voltage	✓

EM AM06 4 x Analog Inputs / 2 x Analog Outputs (6ES7288-3AM06-0AA1)



DC input terminal									
Signals	AI 0+	AI 0-	AI 1+	AI 1-	AI 2+	AI 2-	AI 3+	AI 3-	L+
Pin number	1	3	5	7	9	11	13	15	17
Pin number	2	4	6	8	10	12	14	16	18
Signals	AQ 0	AQ 0M	AQ 1	AQ 1M					M

Note

Keep unused output channels in default voltage mode (0V voltage output), with no wiring required.
If an unused output channel is set to current mode (0mA current output), short-circuit the output; otherwise, the channel will report an open-circuit alarm.

Note

Unused analog inputs should be shorted.

A.4.4 Step response of the analog inputs

Table A-32 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
Sampling time				
• 4 AI x 13 bits	• 0.625 ms	• 4.17 ms	• 5 ms	• 25 ms
• 8 AI x 13 bits	• 1.25 ms	• 4.17 ms	• 5 ms	• 25 ms

A.4.5 Sample time and update times for the analog inputs

Table A-33 Sample time and update time

Rejection frequency (Integration time)	**Sampling 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 SM)

Table A-34 Analog input representation for voltage (SB and EM)

System		**Voltage Measuring Range**				
dec	**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					

System		Voltage Measuring Range				
dec	**Hexadecimal**	**±10 V**	**±5 V**	**±2.5 V**	**±1.25 V**	
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-35 Analog input representation for current (SB and EM)

System		Current measuring range	
dec	**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	Rated 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-36 Analog output representation for voltage (SB and EM)

System		Output ranges, voltage	
dec	**Hexadecimal**	**±10 V**	
32767	7FFF	See note 1	Overflow
32512	7F00	See note 1	

System		Output ranges, voltage	
dec	**Hexadecimal**	**±10 V**	
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-37 Analog output representation for current (SB and EM)

System		Current output range	
dec	**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		Undershoot range
-6912	E500		
-6913	E4FF		Not possible. Output value limited to 0 mA.
-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 expansion modules (EMs)

A.5.1 EM AT08 thermocouple specifications

Table A-38 General specifications

Model	**EM AT08 AI 8 x 16 bit TC**
Part number	6ES7288-3AT08-0AA1
Dimensions W x H x D (mm)	15 x 118 x 74.5
Weight	79 g
Power consumption	0.525 W
Current consumption (SM Bus)	105 mA

Table A-39 Analog inputs

Model		**EM AT08 AI 8 x 16 bit TC**
Number of inputs		8
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 V
Noise rejection		For the selected filter setting 85 dB (10 Hz/50 Hz/60 Hz/400 Hz)
Common mode rejection		greater than 120 dB
Impedance		≥ 10 MΩ
Isolation	Field side to logic	707 V DC
Channel to channel isolation		120 V AC
Resolution		See Thermocouple selection table.
Repeatability		±0.05% FS
Measuring principle		Integrating
Module update time		See Filter selection table.
Cold junction error		±2.5 °C
Cable length (meters)		100 m to sensor max.
Wire resistance		100 Ω max.

Table A-40 Diagnostics

Model	EM AT08 AI 8 x 16 bit TC
Overflow/underflow [1]	x
Wire break [2]	x

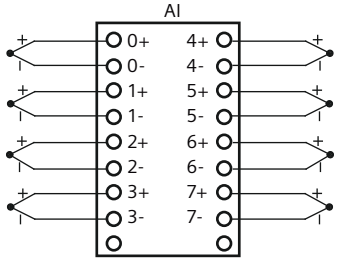
[1] The overflow, underflow and low voltage diagnostic alarm information will be reported in the analog data values even if the alarms are disabled in the module configuration.

[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 AT08 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.

EM AT08 RTD thermocouple 8-point 16-bit (6ES7288-3AT08-0AA1)



DC input terminal									
Signals	AI 0+/TC	AI 0-/TC	AI 1+/TC	AI 1-/TC	AI 2+/TC	AI 2-/TC	AI 3+/TC	AI 3-/TC	
Pin number	1	3	5	7	9	11	13	15	17
Pin number	2	4	6	8	10	12	14	16	18
Signals	AI 4+/TC	AI 4-/TC	AI 5+/TC	AI 5-/TC	AI 6+/TC	AI 6-/TC	AI 7+/TC	AI 7-/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 AT08 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 AT08 Thermocouple expansion module are shown in the table below.

Table A-41 EM AT08 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	±0.6 °C	±1.2 °C
K	-270.0 °C	-200.0 °C	1372.0 °C	1622.0 °C	±0.7 °C	±1.5 °C
T	-270.0 °C	-200.0 °C	400.0 °C	540.0 °C	±0.7 °C	±1.5 °C
E	-270.0 °C	-200.0 °C	1000.0 °C	1200.0 °C	±0.6 °C	±1.2 °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	±3.5 °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 ±2.5 °C for all ranges. This error must be added to the error values in this table. The module requires at least 30 minutes of warm-up time to meet this specification.

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-42 Noise reduction and update times for the EM AT08 thermocouple

Rejection frequency selection	Integral time	8 Channel module update time (seconds)
400 Hz (2.5 ms)	10 ms [1]	0.286
60 Hz (16.6 ms)	16.67 ms	0.446

Rejection frequency selection	Integral time	8 Channel module update time (seconds)
50 Hz (20 ms)	20 ms	0.526
10 Hz (100 ms)	100 ms	2.450

[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 or 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-43 Representation of analog values for Thermocouple Type J

Type J in °C	Function units		Type J in °F	Function units		Range
	dec	Hexadecimal		dec	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	High limit violated
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	Low limit violated
< -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.6 RTD expansion modules (EMs)

A.6.1 EM AR04 RTD specifications

EM RTD specifications

Table A-44 General specifications

Technical data	EM RTD 4 x 16 bit (EM AR04)
Part number	6ES7288-3AR04-0AA1
Dimensions W x H x D (mm)	15 x 118 x 74.5
Weight	77 g
Power consumption	0.525 W
Current consumption (SM Bus)	105 mA

Table A-45 Analog inputs

Technical data		EM RTD 4 x 16 bit (EM AR04)
Number of inputs		4
Type		Module referenced RTD and Ω
Range Nominal range (data word) Overshoot/undershoot range (data word) Overflow/underflow (data word)		See RTD Sensor selection table
Resolution	Temperature	0.1 °C / 0.1 °F
	Resistor	15 bits + sign bit
Maximum withstand voltage		±35 V
Noise rejection		85 dB (10 Hz/50 Hz/60 Hz/400 Hz)
Common mode rejection		>120 dB
Impedance		≥10 M Ω
Isolation	Field side to logic	707 V DC
Channel to channel isolation		None
Resolution		See RTD Sensor selection table
Repeatability		±0.05% FS
Maximum sensor dissipation		0.5 m W
Measuring principle		Sigma-delta
Module update time		See Noise reduction selection table.
Cable length (max.), in meters		100 m to sensor max.
Wire resistance (max.)	except 10 Ω RTD	20 Ω
	10 Ω RTD	2.7 Ω

Table A-46 Diagnostics

Technical data	EM RTD 4 x 16 bit (EM AR04)
Overflow/underflow [1,2]	Yes
Wire break [3]	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-47 Ranges and accuracy for the different sensors supported by the RTD expansion module

Temperature coefficient	RTD types	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
Ni 0.006170	Ni 100	-105.0 °C	-60.0 °C	180.0 °C	212.4 °C	±0.5 °C	±1.0 °C

Temperature coefficient	RTD types	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
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-48 Resistor

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
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.

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-49 Noise reduction and update times for the RTD module

Rejection frequency selection	Integral 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

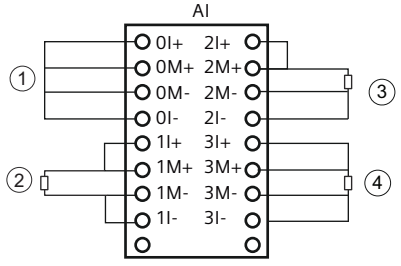
Rejection frequency selection	Integral time	Update time (seconds)
50 Hz (20 ms)	20 ms	4-/2-wire: 0.262 3-wire: 0.505
10 Hz (100 ms)	100 ms	4-/2-wire: 1.222 3-wire: 2.445

[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.

Wiring diagram for EM AR04 RTD 4-point 16-bit (6ES7288-3AR04-0AA1)



DC input terminal

Signals	AI 0I+	AI 0M+	AI 0M-	AI 0I-	AI 1I+	AI 1M+	AI 1M-	AI 1I-	
Pin number	1	3	5	7	9	11	13	15	17
Pin number	2	4	6	8	10	12	14	16	18
Signals	AI 2I+	AI 2M+	AI 2M-	AI 2I-	AI 3I+	AI 3M+	AI 3M-	AI 3I-	

① Loop-back unused RTD inputs ② 2-wire RTD ③ 3-wire RTD ④ 4-wire RTD

NOTICE

External voltage reference generation

This module utilizes an excitation current source to generate an external voltage reference. To successfully generate an external voltage reference, the external current loop must be well connected. If issues such as wire breakage occur during the self-calibration process, the excitation current source cannot generate the reference properly, leading to abnormal calibration parameters and consequently preventing normal data sampling.

A.7 RS485/RS232 signal board

A.7.1 SB RS485/RS232 specifications

Table A-50 General specifications

Technical data	SB RS485/RS232
Part number	6ES7288-5CM01-0AA1
Dimensions W x H x D (mm)	14.5 x 62.45 x 51.5
Weight	20 grams
Power consumption	0.35 W
Current consumption (5 V DC)	30 mA[1]
Current consumption (24 V DC)	8 mA[2]

[1] CPU internal power supply

[2] CPU internal power supply, applicable only in RS485 mode

Table A-51 RS485 Transmitter and receiver

Technical data	SB RS485
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	96 K Ω min.
Receiver threshold/sensitivity	20 mV/200 mV, typical hysteresis 40 mV
Isolation	RS485 signal to chassis ground, 707 V DC; RS485 signal to CPU logic common, 707 V DC;
Cable length, shielded	With isolated repeater: 1000 m up to 187.5 Kbps Without isolated repeater: 50 m

Table A-52 RS232 Transmitter and receiver

Technical data	SB RS232
Transmitter output voltage	+/-5 V min. at RL = 3K Ω
Transmit output voltage	+/-5.5 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	+/- 25 V DC max.

Technical data	**SB RS232**
Isolation	RS232 signal to chassis ground, 707 V DC; RS232 signal to CPU logic common, 707 V DC
Cable length, shielded	10 m max.

SB CM01 RS485/RS232 (6ES7288-5CM01-0AA1)		
SB CM01 5V Tx/B Rx/A M RTS	**Signals**	**X12: CM01 Signal Board (SB) port (Port 1)**
	+5 V DC	+5 V with internal 100 Ω series resistor
	Tx/B	RS232-Tx/RS485-B
	Rx/A	RS232-Rx/RS485-A
	M-ground	Logic common
	RTS (TTL)	Request to send

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 of the specific CPU (Page 1146) to determine the 24 V DC sensor power budget. Refer to the Calculating power budget (Page 1146) to determine how much power (or current) the CPU can provide for your configuration.

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 CPU also provides a 24 V DC sensor provide 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.

See also

Calculating a sample power requirement (Page 1180)

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 ST32
- 1 EM analog 4 x inputs (AE04)
- 1 EM analog 8 x inputs (AE08)
- 1 EM analog output (AQ04)

In this example, the CPU sensor power supply provides sufficient 24 V DC current for all expansion modules. The I/O requires 230 mA and the CPU provides 300 mA. This installation does not require an additional power supply to provide power to the expansion modules.

Table B-1 Calculation of the power budget for a sample configuration

CPU Power Budget	**24 V DC**
CPU ST32 sensor supply	300 mA
minus	
System requirements	
Slot 0: EM AE04	40 mA
Slot 1: EM AE08	50 mA
Slot 1: EM AQ04	150 mA
Total requirements	240 mA

CPU Power Budget	**24 V DC**
equals	
Current balance	**24 V DC**
Current balance total	60 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 1146) 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	**24 V DC**
minus	
System requirements	**24 V DC**
Total requirements	
equals	
Current balance	**24 V DC**
Current balance total	

Error code

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 (Page 172) 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 compilation errors) will prevent the program from being downloaded to the PLC. STEP 7-Micro/WIN SMART V3 detects compilation errors during program compilation (Page 81) or downloading (Page 81) and displays them in the Output Window (Page 157). 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 1191) 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
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
0016	HSC or interrupt input point already assigned to motion

Hexadecimal Error Code	Non-fatal run-time programming problem
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
00C0	Trace 0 configuration verification failed
00C1	Trace 1 configuration verification failed
00C2	Trace 2 configuration verification failed
00C3	Trace 3 configuration verification failed
00C4	Motion control track task failed to load
00CF	Exceeded the maximum number of motion control function block instances
00D0	Insufficient Motion resources
00D1	MC_USER_TASK timeout

C.3 PLC non-fatal error SM flags

Overview

Non-fatal errors are those errors that may partially degrade PLC performance but do not prevent the PLC from executing the user program and updating the I/O. To assist you in debugging the program, information related to the error condition is stored in a dedicated memory (SM) unit (Page 1191) that is accessible from 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 Bits	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
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
		SMB2031	Module 8 I/O Error Byte
		SMB2033	Module 9 I/O Error Byte
		SMB2035	Module 10 I/O Error Byte
		SMB2037	Module 11 I/O Error Byte
		SMB2039	Module 12 I/O Error Byte
		SMB2041	Module 13 I/O Error Byte
		SMB2043	Module 14 I/O Error Byte
		SMB2045	Module 15 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.

- Transition to STOP Mode
- Turns on both the System Fault LED and the STOP LED
- Shutdown

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
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
0028	A trace SDB checksum error was detected in the internal flash memory
0029	Trace SDB compilation failed

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 1191), 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, SMB950 to SMB1999, and SMB2014 to SMB2849 (MBS7-200 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 1193) System status bits
- SMB1 (Page 1194) Instruction execution status bits
- SMB2 (Page 1195) Freeport receive character
- SMB3 (Page 1195) Freeport parity error
- SMB4 (Page 1196) Interrupt queue overflow, run-time program error, interrupts enabled, freeport transmitter idle, and forced value
- SMB5 (Page 1196) I/O error status bits
- SMB6-SMB7 (Page 1197) CPU ID, error status, and digital I/O points
- SMB8-SMB19 (Page 1197) I/O module ID and errors
- SM20.0 (Page 1196) Freeport transmitter idle
- SMW22-SMW26 (Page 1199) Scan times
- SMB28-SMB29 (Page 1199) Signal board 0 ID and errors
- SMB32-SMB33 (Page 1199) Signal board 1 ID and errors
- SMB220-SMB235 (Page 1217) PID control error code

- SMB250-SMB349 (Page 1225) EM (expansion module) hardware/firmware ID
- SMB480-SMB515 (Page 1218) Data log status (read only)
- SMB929-SMB936 Trace
- SMB950-SMB999 (Page 1224) SB (signal board 1) hardware/firmware ID
- SMB1000-SMB1049 (Page 1224) CPU hardware/firmware ID
- SMB1050-SMB1099 (Page 1224) SB (signal board 0) hardware/firmware ID
- SMB1100-SMB1399 (Page 1225) EM (expansion module) hardware/firmware ID
- SMB1400-SMB1799 (Page 1231) EM (expansion module) module-specific data
- SMB1800-SMB1939 (Page 1232) PROFINET device status
- SMB1940-SMB1946 (Page 1233) Web server status
- SMB2050-SMB2449 (Page 1225) EM (expansion module) hardware/firmware ID
- SMB2450-SMB2999 (Page 1231) EM (expansion module) module-specific data

SMB30 to SMB194, SMB566 to SMB858, SMB870 to SMB928, and SMB2014 to SMB2028 (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 1201) 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 1202) Time intervals for timed interrupts
- SMB195-SMB206 (Page 1202) 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 1202): High-speed counter configuration and operation
- SMB66-SMB85 (Page 1204) PLS0 and PLS1 high-speed outputs
- SMB86-SMB94 and SMB186-SMB194 (Page 1211) Receive message control

- SMW98 (Page 1215) I/O expansion bus communication errors
- SMW100-SMW120 (Page 1215) System alarms
- SMW2014-SMW2028 (Page 1215) system alarm
- SMB130 (Page 1201) Port configuration for the CM01 Signal Board (SB) RS232/RS485 port (Port 1) (See SMB30)
- SMB166-SMB169 (Page 1204) PTO0 profile definition table
- SMB176-SMB179 (Page 1204) PTO1 profile definition table
- SMB186-SMB194 (Page 1211) Receive message control (See SMB86-SMB94)
- SMB195-SMB206 (Page 1217) HSC Type and Frequency measurement sampling period
- SMB380-SMB404 (Page 1218) HSC User-defined pin mapping enable
- SMB566-SMB575 (Page 1204) PLS2 high-speed output
- SMB576-SMB579 (Page 1204) PTO2 profile definition table
- SMB580-SMB589 (Page 1204) PLS3 high-speed output
- SMB590-SMB593 (Page 1204) PTO3 profile definition table
- SMB600-SMB649 (Page 1219) Axis 0 open loop motion control
- SMB650-SMB699 (Page 1221) Axis 1 open loop motion control
- SMB700-SMB749 (Page 1221) Axis 2 open loop motion control
- SMB750-SMB799 (Page 1221) Axis 3 open loop motion control
- SMB800-SMB858 (Page 1221) Axes group 0 open loop motion control
- SMB870-SMB928 (Page 1222) 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.

S7-200 SMART variable name	SM address	Description
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.
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, SM20.0: 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 398) 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.

Table D-6 SM20.0 system status

S7-200 SMART variable name	SM address	Description
Xmit2_Idle	SM20.0	TRUE: Port 2 transmitter is idle (FALSE: Transmission in progress).

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-7 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 names	SM addresses	Read-only SMB6 and SMB7 (CPU ID, error status, and digital I/O point)								
CPU_ID	SMB6	MSB							LSB	
		7							0	
		x	x	x	x	c	d	0	0	
	SM6.4 to SM6.6	1	0	0	0	= CPU CR20s				
		1	0	0	1	= CPU CR40s				
		1	0	1	0	= CPU CR60s				
		1	0	1	1	= CPU SR20 / ST20				
		1	1	0	0	= CPU SR40 / ST40				
		1	1	0	1	= CPU SR60 / ST60				
		1	1	1	0	= CPU CR30s				
		1	1	1	1	= CPU SR30 / ST30				
		0	0	0	0	= CPU ST32				
	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 1215)

D.9 SMB8-SMB19, SMB131-SMB134, SMB2030-SMB2045: I/O module ID and errors

SMB8 through SMB19 and SMB131 to SMB134 are organized in byte pairs for expansion modules 0 to 15.

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)
EM8_ID	SMB2030	Expansion module 8 ID register
EM8_Err	SMB2031	Expansion module 8 error register (See SMW2014 for diagnostic alarm code)
EM9_ID	SMB2032	Expansion module 9 ID register
EM9_Err	SMB2033	Expansion module 9 error register (See SMW2016 for diagnostic alarm code)
EM10_ID	SMB2034	Expansion module 10 ID register
EM10_Err	SMB2035	Expansion module 10 error register (See SMW2018 for diagnostic alarm code)
EM11_ID	SMB2036	Expansion module 11 ID register
EM11_Err	SMB2037	Expansion module 11 error register (See SMW2020 for diagnostic alarm code)
EM12_ID	SMB2038	Expansion module 12 ID register
EM12_Err	SMB2039	Expansion module 12 error register (See SMW2022 for diagnostic alarm code)
EM13_ID	SMB2040	Expansion module 13 ID register
EM13_Err	SMB2041	Expansion module 13 error register (See SMW2024 for diagnostic alarm code)
EM14_ID	SMB2042	Expansion module 14 ID register
EM14_Err	SMB2043	Expansion module 14 error register (See SMW2026 for diagnostic alarm code)
EM15_ID	SMB2044	Expansion module 15 ID register
EM15_Err	SMB2045	Expansion module 15 error register (See SMW2028 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 1215)

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-8 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							

| | | |

	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 400). 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-9 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
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

SM definition	S7-200 SMART variable name	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	HSC6	HSC7
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
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**

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.

The CPU uses SMB2000-SMB2013 to monitor and control pulse train output PTO4 and pulse width modulation output PWM4.

Table D-10 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

S7-200 SMART variable name	SM address	Function
PLS0_Select	SM67.6	PTO0/PWM0 mode select: • FALSE: PWM • TRUE: PTO
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-11 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

S7-200 SMART variable name	SM address	Function
PTO1_PC_Update	SM77.2	PTO1 update the pulse count value: • FALSE: No update • TRUE: Write new pulse count
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-12 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

S7-200 SMART variable name	SM address	Description
PLS2_Idle	SM566.7	PTO2 idle: • FALSE: PTO in progress • TRUE: PTO is idle
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-13 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-14 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.

<table>
<tr><th rowspan="2">S7-200 SMART variable name</th><th colspan="3">SM address</th><th rowspan="2">Bit for-mat</th><th colspan="9">Receive message status byte</th></tr>
<tr><td></td><td>Port 0</td><td>Port 1</td><td>Port 2</td><td>MSB</td><td></td><td></td><td></td><td></td><td></td><td></td><td>LSB</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td>7</td><td></td><td></td><td></td><td></td><td></td><td></td><td>0</td></tr>
<tr><td>P0_Stat_Rcv</td><td>SMB86</td><td></td><td></td><td rowspan="3"></td><td colspan="9" rowspan="3">n r e 0 0 t c p</td></tr>
<tr><td>P1_Stat_Rcv</td><td></td><td>SMB186</td><td></td></tr>
<tr><td>P2_Stat_Rcv</td><td></td><td></td><td>SMB405</td></tr>
<tr><td>P0_Stat_Rcv_7</td><td>SM86.7</td><td></td><td></td><td rowspan="3">n:</td><td>1</td><td colspan="8" rowspan="3">= Receive message was terminated by user disable command
else n = 0</td></tr>
<tr><td>P1_Stat_Rcv_7</td><td></td><td>SM186.7</td><td></td></tr>
<tr><td>P2_Stat_Rcv_7</td><td></td><td></td><td>SM405.7</td></tr>
</table>

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	SM87.6			sc:	0	= Ignore SMB88 or SMB188							
P1_Ctrl_Rcv_6		SM187.6			1	= Use the value of SMB88 or SMB188 to detect start of message							
P2_Ctrl_Rcv_6			SM406.6										

P0_Ctrl_Rcv_5	SM87.5			ec:	0	= Ignore SMB89 or SMB189
P1_Ctrl_Rcv_5		SM187.5			1	= Use the value of SMB89 or SMB189 to detect end of message
P2_Ctrl_Rcv_5			SM406.5			
P0_Ctrl_Rcv_4	SM87.4			il:	0	= Ignore SMB90 or SMB190
P1_Ctrl_Rcv_4		SM187.4			1	= Use the value of SMB89 or SMB189 to detect an idle line condition
P2_Ctrl_Rcv_4			SM406.4			
P0_Ctrl_Rcv_3	SM87.3			c/m:	0	= Timer is an inter-character timer
P1_Ctrl_Rcv_3		SM187.3			1	= Timer is a message timer
P2_Ctrl_Rcv_3			SM406.3			
P0_Ctrl_Rcv_2	SM87.2			Tmr:	0	= Ignore SMW92 or SMW192
P1_Ctrl_Rcv_2		SM187.2			1	= Terminate receiving if the time period in SMW92 or SMW192 is exceeded
P2_Ctrl_Rcv_2			SM406.2			
P0_Ctrl_Rcv_1	SM87.1				0	= Ignore break conditions
P1_Ctrl_Rcv_1		SM187.1			1	= Use break condition as start of message detection
P2_Ctrl_Rcv_1			SM406.1			

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 for-mat	**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-15 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, SMW2014-SMW2028 system alarms

Special memory words SMW100-SMW120 and SMW2014-SMW2028 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
EM8_Alarm	SMW2014	Expansion module bus slot 8 diagnostic alarm code
EM9_Alarm	SMW2016	Expansion module bus slot 9 diagnostic alarm code
EM10_Alarm	SMW2018	Expansion module bus slot 10 diagnostic alarm code
EM11_Alarm	SMW2020	Expansion module bus slot 11 diagnostic alarm code
EM12_Alarm	SMW2022	Expansion module bus slot 12 diagnostic alarm code
EM13_Alarm	SMW2024	Expansion module bus slot 13 diagnostic alarm code
EM14_Alarm	SMW2026	Expansion module bus slot 14 diagnostic alarm code
EM15_Alarm	SMW2028	Expansion module bus slot 15 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 Noise reduction for high-speed inputs (Page 346)

Table D-16 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-17 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

S7-200 SMART variable name	SM address	Function
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

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		

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**	
SM622.1	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.
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.

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 1219)" 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 1219)" 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 1219)" 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 1219)" 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-18 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
SMB822	Reserved for instantaneous axes group status.

SM Bits	Description								
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.						
	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								
SMB857	Last completed path ID								
SMB858	Last completed segment ID								

D.29 SMB870-SMB928: Axes group 1 open loop motion control

Table D-19 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								
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.						
	Done	0	Motion on the axes group is in progress.						
		1	Operation completes (set by the axes group during initialization).						

SM Bits	Description
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 SMB929-SMB936: Trace control and trace status

The S7-200 SMART CPU uses the following special memory for monitoring and control traces.

SM definitions	Trace 0	Trace 1	Trace 2	Trace 4
Is the trace active? • 0 = No • 1 = Yes	SMB929	SMB931	SMB933	SMB935
Trace status: • 0 = Not configured • 1 = Configured • 2 = Logging • 3 = Triggered • 4 = Stop	SMB930	SMB932	SMB934	SMB936

D.31 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.32 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.33 SMB1100-SMB1399, SMB250-SMB349, SMB2050-SMB2449: EM (expansion module) hardware/firmware ID

The CPU writes expansion module information to special memory after a power-up or warm restart transition. The special memory ranges SMB1100-SMB1399, SMB250 to SMB349, and SMB2050-SMB2449 are 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: Function 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: Function 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: Function 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: Function 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: Function 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: Function 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: Function 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

SM addresses for slot 8	Description
SMW2050	EM bus slot 8 vendor ID: 0x002A if a Siemens EM is present; 0x0000 if no EM is present
SMB2052 to SMB2071	EM bus slot 8 vendor ID: If a Siemens article number (MLFB) exists: ASCII characters, left-justified in field, padded with spaces
SMB2072 to SMB2087	EM bus slot 8 vendor ID: If a Siemens article number (M serial number) exists: ASCII characters, left-justified in field, padded with spaces
SMW2088	EM bus slot 8 vendor ID: If a Siemens hardware version exists: Represents the hardware E-stand; range = 0x0001 to 0xFFFD 0x0000, 0xFFFE, and 0xFFFF are reserved values.
SMD2090	EM bus slot 8 firmware version: • Byte 0: ASCII 'V' • Byte 1: Function version • Byte 2: Minor change version • Byte 3: Bug fix version
SMW2094	EM bus slot 8 firmware version counter (range 0x0000 to 0x00FF)
SMW2096	Reserved, always 0x0000
SMW2098	EM bus slot 8 vendor ID: If a Sie device type exists: I/O=0x0003, communications=0x0004, all other values reserved

SM addresses for slot 9	Description
SMW2100	EM bus slot 9 vendor ID: 0x002A if a Siemens EM is present; 0x0000 if no EM is present
SMB2102 to SMB2121	EM bus slot 9 vendor ID: If a Siemens article number (MLFB) exists: ASCII characters, left-justified in field, padded with spaces
SMB2122 to SMB2137	EM bus slot 9 vendor ID: If a Siemens article number (M serial number) exists: ASCII characters, left-justified in field, padded with spaces
SMW2138	EM bus slot 9 vendor ID: If a Siemens hardware version exists: Represents the hardware E-stand; range = 0x0001 to 0xFFFD 0x0000, 0xFFFE, and 0xFFFF are reserved values.
SMD2140	EM bus slot 9 firmware version: • Byte 0: ASCII 'V' • Byte 1: Function version • Byte 2: Minor change version • Byte 3: Bug fix version

SM addresses for slot 9	Description
SMW2144	EM bus slot firmware version counter (range 0x0000 to 0x00FF)
SMW2146	Reserved, always 0x0000
SMW2148	EM bus slot 9 device type: I/O=0x0003, communications=0x0004, all other values reserved

SM addresses for slot 10	Description
SMW2150	EM bus slot 10 vendor ID: 0x002A if a Siemens EM is present; 0x0000 if no EM is present
SMB2152 to SMB2171	EM bus slot 10 vendor ID: If a Siemens article number (MLFB) exists: ASCII characters, left-justified in field, padded with spaces
SMB2172 to SMB2187	EM bus slot 10 vendor ID: If a Siemens article number (M serial number) exists: ASCII characters, left-justified in field, padded with spaces
SMW2188	EM bus slot 10 vendor ID: If a Siemens hardware version exists: Represents the hardware E-stand; range = 0x0001 to 0xFFFD 0x0000, 0xFFFE, and 0xFFFF are reserved values.
SMD2190	EM bus slot 10 firmware version: • Byte 0: ASCII 'V' • Byte 1: Function version • Byte 2: Minor change version • Byte 3: Bug fix version
SMW2194	EM bus slot 10 firmware version counter (range 0x0000 to 0x00FF)
SMW2196	Reserved, always 0x0000
SMW2198	EM bus slot 10 device type: I/O=0x0003, communications=0x0004, all other values reserved

SM addresses for slot 11	Description
SMW2200	EM bus slot 11 vendor ID: 0x002A if a Siemens EM is present; 0x0000 if no EM is present
SMB2202 to SMB2221	EM bus slot 11 vendor ID: If a Siemens article number (MLFB) exists: ASCII characters, left-justified in field, padded with spaces
SMB2222 to SMB2237	EM bus slot 11 vendor ID: If a Siemens article number (M serial number) exists: ASCII characters, left-justified in field, padded with spaces
SMW2238	EM bus slot 11 vendor ID: If a Siemens hardware version exists: Represents the hardware E-stand; range = 0x0001 to 0xFFFD 0x0000, 0xFFFE, and 0xFFFF are reserved values.
SMD2240	EM bus slot 11 firmware version: • Byte 0: ASCII 'V' • Byte 1: Function version • Byte 2: Minor change version • Byte 3: Bug fix version
SMW2244	EM bus slot 11 firmware version counter (range 0x0000 to 0x00FF)
SMW2246	Reserved, always 0x0000
SMW2248	EM bus slot 11 device type: I/O=0x0003, communications=0x0004, all other values reserved

SM addresses for slot 12	Description
SMW2250	EM bus slot 12 vendor ID: 0x002A if a Siemens EM is present; 0x0000 if no EM is present
SMB2252 to SMB2271	EM bus slot 12 vendor ID: If a Siemens article number (MLFB) exists: ASCII characters, left-justified in field, padded with spaces
SMB2272 to SMB2287	EM bus slot 12 vendor ID: If a Siemens article number (M serial number) exists: ASCII characters, left-justified in field, padded with spaces
SMW2288	EM bus slot 12 vendor ID: If a Siemens hardware version exists: Represents the hardware E-stand; range = 0x0001 to 0xFFFD 0x0000, 0xFFFE, and 0xFFFF are reserved values.
SMD2290	EM bus slot 12 firmware version: • Byte 0: ASCII 'V' • Byte 1: Function version • Byte 2: Minor change version • Byte 3: Bug fix version
SMW2294	EM bus slot 12 firmware version counter (range 0x0000 to 0x00FF)
SMW2296	Reserved, always 0x0000
SMW2298	EM bus slot 12 device type: I/O=0x0003, communications=0x0004, all other values reserved

SM addresses for slot 13	Description
SMW2300	EM bus slot 13 vendor ID: 0x002A if a Siemens EM is present; 0x0000 if no EM is present
SMB2302 to SMB2321	EM bus slot 13 vendor ID: If a Siemens article number (MLFB) exists: ASCII characters, left-justified in field, padded with spaces
SMB2322 to SMB2337	EM bus slot 13 vendor ID: If a Siemens article number (M serial number) exists: ASCII characters, left-justified in field, padded with spaces
SMW2338	EM bus slot 13 vendor ID: If a Siemens hardware version exists: Represents the hardware E-stand; range = 0x0001 to 0xFFFD 0x0000, 0xFFFE, and 0xFFFF are reserved values.
SMD2340	EM bus slot 13 firmware version: • Byte 0: ASCII 'V' • Byte 1: Function version • Byte 2: Minor change version • Byte 3: Bug fix version
SMW2344	EM bus slot 13 firmware version counter (range 0x0000 to 0x00FF)
SMW2346	Reserved, always 0x0000
SMW2348	EM bus slot 13 device type: I/O=0x0003, communications=0x0004, all other values reserved

SM addresses for slot 14	Description
SMW2350	EM bus slot 14 vendor ID: 0x002A if a Siemens EM is present; 0x0000 if no EM is present
SMB2352 to SMB2371	EM bus slot 14 vendor ID: If a Siemens article number (MLFB) exists: ASCII characters, left-justified in field, padded with spaces
SMB2372 to SMB2387	EM bus slot 14 vendor ID: If a Siemens article number (M serial number) exists: ASCII characters, left-justified in field, padded with spaces

SM addresses for slot 14	Description
SMW2388	EM bus slot 14 vendor ID: If a Siemens hardware version exists: Represents the hardware E-stand; range = 0x0001 to 0xFFFD 0x0000, 0xFFFE, and 0xFFFF are reserved values.
SMD2390	EM bus slot 14 firmware version: • Byte 0: ASCII 'V' • Byte 1: Function version • Byte 2: Minor change version • Byte 3: Bug fix version
SMW2394	EM bus slot 14 firmware version counter (range 0x0000 to 0x00FF)
SMW2396	Reserved, always 0x0000
SMW2398	EM bus slot 14 device type: I/O=0x0003, communications=0x0004, all other values reserved

SM addresses for slot 15	Description
SMW2400	EM bus slot 15 vendor ID: 0x002A if a Siemens EM is present; 0x0000 if no EM is present
SMB2402 to SMB2421	EM bus slot 15 vendor ID: If a Siemens article number (MLFB) exists: ASCII characters, left-justified in field, padded with spaces
SMB2422 to SMB2437	EM bus slot 15 vendor ID: If a Siemens article number (M serial number) exists: ASCII characters, left-justified in field, padded with spaces
SMW2438	EM bus slot 15 vendor ID: If a Siemens hardware version exists: Represents the hardware E-stand; range = 0x0001 to 0xFFFD 0x0000, 0xFFFE, and 0xFFFF are reserved values.
SMD2440	EM bus slot 15 firmware version: • Byte 0: ASCII 'V' • Byte 1: Function version • Byte 2: Minor change version • Byte 3: Bug fix version
SMW2444	EM bus slot 15 firmware version counter (range 0x0000 to 0x00FF)
SMW2446	Reserved, always 0x0000
SMW2448	EM bus slot 15 device type: I/O=0x0003, communications=0x0004, all other values reserved

D.34 SMB1400-SMB1799, SMB2450-2999: EM (expansion module) module-specific data

The CPU reserves an additional 50 bytes for each expansion module for module-specific read-only data:

SM address	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

SM address	Description
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
SMB2450 to SMB2499	EM bus slot 8: Module specific information
SMB2500 to SMB2549	EM bus slot 9: Module specific information
SMB2550 to SMB2599	EM bus slot 10: Module specific information
SMB2600 to SMB2649	EM bus slot 11: Module specific information
SMB2650 to SMB2699	EM bus slot 12: Module specific information
SMB2700 to SMB2749	EM bus slot 13: Module specific information
SMB2750 to SMB2799	EM bus slot 14: Module specific information
SMB2800 to SMB2849	EM bus slot 15: Module specific information

D.35 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.36 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

Reference

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

Instruction

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 TRIGR	Detection of Rising Edge
ED TRIGF	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	Load result of String Compare IN1 (x: =, <>) IN2
ASx IN1, IN2	AND result of String Compare IN1 (x: =, <>) IN2
OSx IN1, IN2	OR result of String Compare IN1 (x: =, <>) IN2
ALD	AND Load
OLD	OR Load
LPS	Logic Push (stack control)
LRD	Logic Read (stack control)
LPP	Logic Pop (stack control)
LDS n	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 loops

Timer and counter instructions	
STL	**Description**
TON Txxx, PT	On-delay timer
TOF Txxx, PT	Off-delay timer
TONR Txxx, PT	Retentive on-delay timer
IECTON XXX, PT, ET	On-delay timer (IEC)
IECTOF XXX, PT, ET	Off-delay timer (IEC)
IECTONR XXX, XXX.RESET PT, ET	Retentive on-delay timer (IEC)
BITIM OUT	Beginning Interval Timer
CITIM IN, OUT	Calculate Interval Timer
CTU Cxxx, PV	Count Up
CTD Cxxx, PV	Count Down
CTUD Cxxx, PV	Count Up/Down
CTUDINT XXX, XXX.RESET, XXX.PV, XXX.CV	Count Up (IEC)
CTDDINT XXX, XXX.LOAD, XXX.PV, XXX.CV	Count Down (IEC)
CTUDDINT XXX, XXX.CD, XXX.RESET, XXX.LOAD, XXX.PV, XXX.QD, XXX.CV	Count Up/Down (IEC)

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	Read the time-of-day clock
TODW T	Write the time-of-day clock
TODRX T	Read Real Time Clock Extended
TODWX T	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 Subroutines [N1, ... up to 16 optional parameters] Conditional Return from SBR
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	Convert BCD to Integer
IBCD OUT	Convert Integer to BCD
BTI IN, OUT	Convert Byte to Integer
ITB IN, OUT	Convert Integer to Byte
ITD IN, OUT	Convert Integer to Double Integer
DTI IN, OUT	Convert Double Integer to Integer
DTR IN, OUT	Convert DWord to Real
TRUNC IN, OUT	Convert Real to Double Integer
ROUND IN, OUT	Convert Real to Double Integer
ATH IN, OUT, LEN	Convert ASCII to Hex
HTA IN, OUT, LEN	Convert Hex to ASCII
ITA IN, OUT, FMT	Convert Integer to ASCII
DTA IN, OUT, FM	Convert Double Integer to ASCII
RTA IN, OUT, FM	Convert Real to ASCII
DECO IN, OUT	Decoding
ENCO IN, OUT	Encode
SEG IN, OUT	Generate 7-segment pattern
ITS IN, FMT, OUT	Convert Integer to String
DTS IN, FMT, OUT	Convert Double Integer to String
RTS IN, FMT, OUT	Convert Real to String
STI STR, INDX, OUT	Convert Substring to Integer
STD STR, INDX, OUT	Convert Substring to Double Integer
STR STR, INDX, OUT	Convert Substring to Real

Interrupt instructions	
STL	**Description**
CRETI	Conditional Return from Interrupt
ENI	Enable Interrupts
DISI	Disable Interrupts
ATCH INT, EVNT	Attach Interrupt routine to event
DTCH EVNT	Detach event
CEVENT EVNT	Clear all interrupt events of type EVNT

Communications instructions	
STL	**LAD/FBD**
GET	Reads remote station data
PUT	Writes data to a remote station
XMT TBL, PORT	Freeport transmission
RCV TBL, PORT	Freeport receive message

Communications instructions	
STL	**LAD/FBD**
GIP ADDR, MASK, GATE	Get CPU address, subnet mask, and gateway
SIP ADDR, MASK, GATE	Set CPU address, subnet mask, and gateway
GPA ADDR, PORT	Get Port Address
SPA ADDR, PORT	Set Port Address

E.5 CPU memory ranges and features

The following table lists the CPU memory ranges and features supported by S7-200 SMART CPU firmware version V3.2 (ST32).

Description		CPU ST32
User program size		100 KB
User data size		120 KB
Variable memory (V)		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 number	SMB0 to SMB2999
	Read only	SMB0 to SMB29 SMB32 to SMB33 SMB250 to SMB349 SMB480 to SMB515 SMB950 to SMB1999 SMB2014 to SMB2849
Timer		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
Counter		512 (C0 to C511)
High-speed counter		HC0 to HC7
Sequential control relay (S)		S0.0 to S31.7
Accumulator register		AC0 to AC3
Jump/label		0 to 255

Description	CPU ST32
Call/subroutine	0 to 127
Interrupt routine	0 to 127
Positive/negative transition	1024
PID control loop	0 to 15
Port	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 V3.2 (ST32) supports the PROFINET communication on the following CPU models. For the detailed parameters and PROFINET process image information, refer to the following table.

Description	CPU ST32
Maximum number of PROFINET device	10
Device number of PROFINET device	1 to 10
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 device #9	I1128.0 to I1407.7
CPU address of PROFINET process image input register for device #10	I1408.0 to I1535.7
CPU address of PROFINET process image input register for I-Device	I1152.0 to I1279.7

Description	CPU ST32
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
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 device #9	Q1128.0 to Q1407.7
CPU address of PROFINET process image output register for device #10	Q1408.0 to Q1535.7
CPU address of PROFINET process image output register for I-Device	Q1152.0 to Q1279.7

Ordering information

F

F.1 CPU module

CPU model	Article number
CPU ST32, DC/DC/DC	6ES7288-1ST32-0AA0

F.2 Expansion modules (EMs) and signal boards (SBs)

Expansion modules and signal boards	Article number
EM Digital 16 x Inputs (EM DE16)	6ES7288-2DE16-0AA1
EM Digital 16 x Outputs Transistor (EM QT16)	6ES7288-2QT16-0AA1
EM Analog 4 x Inputs (EM AE04)	6ES7288-3AE04-0AA1
EM Analog 4 x Outputs (EM AQ04)	6ES7288-3AQ04-0AA1
EM Analog 8 x Inputs (EM AE08)	6ES7288-3AE08-0AA1
EM Analog 4 x Inputs / Analog 2 x Outputs (EM AM06)	6ES7288-3AM06-0AA1
EM RTD 4 x 16 bit (EM AR04)	6ES7288-3AR04-0AA1
EM TC 8 x 16 bit (EM AT08)	6ES7288-3AT08-0AA1
SB RS485/RS232 (SB CM01)	6ES7288-5CM01-0AA1

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

General accessories	Article number
End Retainer, Steel	8WA1805
Terminal block 24V	6ES7193-8ST00-1AN0
DIN rail 35 mm, tinned steel strip[1]	
• Length: 483 mm	6ES5710-8MA11
• Length: 530 mm	6ES5710-8MA21

General accessories	Article number
• Length: 830 mm	6ES5710-8MA31
• Length: 2000 mm	6ES5710-8MA41
SIMATIC system rail[1]	
• Length: 483 mm	6ES7193-6MR00-0AA0
• Length: 530 mm	6ES7193-6MR00-0BA0
• Length: 830 mm	6ES7193-6MR00-0CA0
• Length: 2000 mm	6ES7193-6MR00-0DA0

[1] DIN rails and SIMATIC system rails can be shortened as needed.

Index

-

-D (STL-Subtract double integer), 381
-I (STL-Subtract integer), 381
-R (STL-Subtract real), 381

*****

*D (STL-Multiply double integer), 381
*I (STL-Multiply integer), 381
*R (STL-Multiply real), 381

.

.smart files, 145

/

/D (STL-Divide double integer), 381
/I (STL-Divide integer), 381
/R (STL-Divide real), 381

+

+D (STL-Add double integer), 381
+I (STL-Add integer), 381
+R (STL-Add real), 381

<

<=B, 312
<=R, 312
<=W, 312
<>B, 312
<>R, 312
<>S, 316
<>W, 312
<B, 312
<R, 312
<W, 312

=

= (STL-output), 263
==B, 312
==R, 312
==S, 316
==W, 312
=I (STL-output immediate), 263

>

>=B, 312
>=R, 312
>=W, 312
>B, 312
>R, 312
>W, 312

A

A (STL-AND), 255
AB< (STL-AND compare byte less than), 312
AB<= (STL-AND compare byte less than or equal), 312
AB<> (STL-AND compare byte not equal), 312
AB= (STL-AND compare byte equal), 312
AB> (STL-AND compare byte greater than), 312
AB>= (STL-AND compare byte greater than or equal), 312
ABS_DI, 395
ABS_I, 395
ABS_R, 395
ABSDI (STL-absolute double integer), 334, 395
ABSI (STL-absolute integer), 334, 395
Absolute positioning mode, 666
ABSR (STL-absolute real), 334, 395
AC
  Grounding guidelines, 98
  Guidelines for isolation, 98
  Prerequisites for wiring and grounding, 97
  Wiring guidelines, 99
AC inductive loads, 103
Access rights
  CPU security, 55
  password privilege levels, 55
Active/Passive communication partners, 490
AD< (STL-AND compare double word less than), 312
AD<= (STL-AND compare double word less than or equal), 312
AD<> (STL-AND compare double word not equal), 312
AD= (STL-AND compare double word equal), 312

AD> (STL-AND compare double word greater than), 312
AD>= (STL-AND compare double word greater than or equal), 312
ADD_DI, 381
ADD_I, 381
ADD_R, 381
Addressing
  Accumulators, 117
  Analog inputs, 120
  Analog outputs, 120
  counter memory, 116
  creating pointers and using indirect address, 123
  DB2 memory, 115
  example of pointer offset to access data, 126
  example of pointer to access data in a table, 125
  Flag memory, 115
  High-speed counter, 117
  local and expansion I/O, 122
  Local memory, 119
  Memory Area, 114
  Process-image output register:, 115
  sequence control relay (SCR) memory, 121
  special memory (SM) bits, 118
  timer memory, 116
  V memory, 115
  variable memory, 115
AENO (STL stack-AND ENO bit)
  instruction, 259
  logic stack overview, 258
AI (STL-AND immediate), 256
air flow, 85
Alarms
  analog input configuration, 222
  from system (SMW100-SMW120), 1215
ALD (STL stack-AND Load), 259
ALT (STL), 267
ALTP
  STL, 267
AN (STL-AND NOT), 255
Analog I/O
  input representation (current), 1166
  input representation (voltage), 1165
  output representation (current), 1167
  Output representation (voltage), 1166
  Status display, 140
  step response times (SM), 1165
Analog inputs
  Analog type configuration, 220
  Rejection, 220
  Smoothing, 220
  system block configuration, 220
Analog outputs
  Analog type configuration, 223
  states at RUN/ STOP transition, 223
ANDB (STL-AND byte), 414
ANDD (STL-AND dword), 414
ANDW (STL-AND word), 414
ANI (STL-AND NOT immediate), 256
Application user reference, 150
AR< (STL-AND compare real less than), 312
AR<= (STL-AND compare real less than or equal), 312
AR<> (STL-AND compare real not equal), 312
AR= (STL-AND compare real equal), 312
AR> (STL-AND compare real greater than), 312
AR>= (STL-AND compare real greater than or equal), 312
ARCCOS(arccosine), 387
ARCSIN(arcsine), 387
ARCTAN(Inverse tangent), 387
AS<> (STL-AND string compare not equal), 316
AS= (STL-AND string compare equal), 316
ASCII array conversion instructions, 320
Assigning
  local variables, 166
  local variables (local), 163
ATCH, 398
ATH, 320
ATT, 448
AVG_B, 394
AVG_DI, 394
AVG_I, 394
AVG_R, 394
AW< (STL-AND compare word less than), 312
AW<= (STL-AND compare word less than or equal), 312
AW<> (STL-AND compare word not equal), 312
AW= (STL-AND compare word equal), 312
AW> (STL-AND compare word greater than), 312
AW>= (STL-AND compare word greater than or equal), 312
Axes group 0 SM addresses (SMB800-SMB848), 1221
Axes group 1 SM addresses (SMB870-SMB928), 1222
Axes group configuration
  buffer mode, 1120
Axes group of motion
  choosing an axis name, 863
  Diagnostics, 864
  path, 1117
  segment, 1117
  State of axes group, 865
Axes group of Motion
  GRPx_2D_MOVELINEAR, 1123
  GRPx_3D_MOVELINEAR, 1123

GRPx_MOVEPATH, 1125
GRPx_RESET, 1126
Programming, 1115
Axis 0 motion control (SMB600-SMB649), 1219
Axis 1 motion control (SMB650-SMB699), 1219
Axis 2 motion control (SMB700-SMB749), 1219
Axis 3 motion control (SMB750-SMB799), 1219
Axis of Motion
ACCEL_TIME, 1044
Assigning the motor speed, 818, 826, 848
Axis state, 819, 833, 841, 862
AXISx_ABSPOS, 1077
AXISx_CACHE, 1075
AXISx_CFG, 1074
AXISx_CTRL, 1064
AXISx_DIS, 1074
AXISx_GOTO, 1067
AXISx_LDOFF, 1071
AXISx_LDPOS, 1072
AXISx_MAN, 1066
AXISx_RDPOS, 1076
AXISx_RSEEK, 1070
AXISx_RUN, 1069
AXISx_SRATE, 1073
Choosing an axis name, 815, 820, 834, 842
choosing directional control and input configuration, 835
Choosing the directional control and output configuration, 821
configuring, 1049
configuring reference point and seek parameters, 1060
configuring the Backlash compensation, 1059
configuring the input pin locations, 1050
Configuring the LMT- input, 824, 846
Configuring the LMT+ input, 824, 846
Configuring the RPS input, 829
Configuring the STP input, 831
Configuring the ZP input, 830
defining the motion profile, 1062
Diagnostics, 819, 833, 841
displaying and controlling the operation of axes, 1092
displaying and modifying the configuration of axes, 1097
displaying the profile configuration for the axes, 1097
eliminating backlash, 1114
entering acceleration time, 1057
entering jerk time, 1058
entering jog parameters, 1057
entering maximum start and stop speed, 1056
error codes, 1098
Homing, 830
mapping the I/O, 1051
Modulo, 816, 824, 838, 845
Origin Point Offset, 831
phasing, 1053
polarity, 1055
programming, 1048
RP Seek modes, 1110
Selecting a measurement system, 816, 823, 837, 844
Setting the acceleration and deceleration times, 818, 828, 848
setting the jerk time, 819, 829, 849
SM locations, 1108
subroutine guidelines, 1064
subroutines, 1063
Touch Probe, 833, 839
AXISx_GOTO
override, 1068

B

B_I, 317
Baud rate
Communication, 52, 209
setting, 528
switch selections:RS232/PPI Multi-Master cable, 542
BCD_I, 317
BCDI (STL-BCD to integer), 317
BGN_ITIME, 467
Biasing and terminating
CM01 signal board, 538
Network cable, 538
Biasing PID loop, 398
BIR (STL-byte immediate read), 422
Bit logic instructions
AENO (STL-AND ENO), 258
ALT, 267
ALTP, 267
contacts, 255
contacts (Immediate), 256
edge detectors, 262
Input examples, 269
NOP (No operation), 266
NOT, 261
output coils, 263
output examples, 270
set and reset bits, 264
set and reset dominant bistable, 265

STL logic stack instructions for inputs, 258
STL logic stack operations, 259
BITIM (STL-Begin interval timer), 467
BIW (STL-byte immediate write), 422
BLKMOV_B, 420
BLKMOV_D, 420
BLKMOV_W, 420
BMB (STL-block move byte), 420
BMD (STL-block move double word), 420
BMW (STL- block move word), 420
Bookmarks in programs, 731
BTI (STL-Byte to integer), 317
Buffer mode
Aborting, 1120
Buffered, 1120
Building status charts, 737
Building your communication network, 535
Byte access
Byte usage table, 733

C

CAL_ITIME, 467
CALL, 468, 474
Cam table, 866
Card (memory), 134, 135
CE approval, 1141
CFND (STL-Find character), 445
Character error received from Freeport (SMB3), 1195
Charts
building, 737
creating, 737
opening, 737
CHR_FIND, 445
CITIM (STL-Calculate interval time), 467
Clear, 69
clearance for airflow and cooling, 85
Cleared memory card, 235
Clearing PLC memory, 233
Clock instructions
READ_RTC, 272
READ_RTCX, 274
SET_RTC, 272
SET_RTCX, 274
Closed loop position control, 770
Closed-loop control, 770
CLR_EVNT, 398
CM01 signal board
Biasing and terminating, 538
connector pin assignments, 537
Coils
output, 263
output (immediate), 263
reset bits, 264
reset bits immediate, 264
set bits, 264
set bits immediate, 264
Cold junction compensation
Thermocouple, 231, 1170
Comment
appearance in configuration page, 811
Communication
dialog, 493, 533
Ethernet hardware connection, 73
Ethernet, RS485, and RS232, 32, 487
IP Address, 493
Network, 72
number of connections (Ethernet), 488
number of connections (RS232), 488
number of connections (RS485), 488
point-to-point interface (PPI) protocol, 527
RS485 network address, 532
Communication ports, 489
connector pin assignments, 537
Freeport mode, 539
system block configuration, 52, 209
Communications
choices, 526
locating MAC address on CPU, 500
restricting writes, 55
RS485 configuration, 526
Communications instructions
Receive, 285
Transfer, 285
Compare instructions
compare character strings, 316
compare number value, 312
Compiling programs
Downloads, 127
PLC information (errors), 172
PLC non-fatal program errors, 1184
Component properties, 65
Configuration
CPU, system block, 205
dynamic IP information, 493
Ethernet, 493
IP Address, 493
MAC address, 500
profile table (Axis of Motion), 1101
RS485 network, 532
RS485 network address, 532
static IP information, 495
Configuration drawings, 141
Configuring trigger conditions, 198

Connection
number of connections (Ethernet), 488
number of connections (RS232), 488
number of connections (RS485), 488
types of communication, 32, 487
Contact
Falling edge detector, 268
Rising edge detector, 268
Contact information, 4
Contacts
ALT, 267
ALTP, 267
negative edge detector, 262
normally closed, 255
normally closed (immediate), 256
normally open, 255
normally open (immediate), 256
NOT, 261
positive edge detector, 262
Convert instructions
ASCII array conversions, 320
ASCII sub-string to number value, 330
encode and decode, 332
number value to string, 326
standard conversion, 317
cooling, 85
COS (cosine), 387
Counter instructions
high-speed counters, 344
standard counters, 337
CPU
accessing data, 113
clearing memory, 233
configuring communication to HMI, 502
DIN rail, 90
Ethernet communications, 490
Ethernet ports, 493
expansion modules supported, 29
fatal errors, 1187
Installation, 90
IP Address, 493
LEDs, 139
MAC address, 500
Memory card, 134
Non-Fatal Error Memory Locations, 1186
number of communication connections, 488
number of PPI connections, 527
process image register, 111
RS485 network address, 532
RS485 network port, 532
setting the type, 79
ST32 specifications, 1146
ST32 wiring diagram, 1152
System block, 205
types of communication, 32, 487
viewing PLC information, 172
CPU hardware/firmware ID (SMB1000-SMB1049), 1224
CPU ID register (SMB6-SMB7), 1197
CPU role
Controller and I-Device, 526
creating
user-defined library, 722
CRET (STL-Conditional return from function block), 474
CRET (STL-Conditional return from subroutine), 468
Cross reference, 731
byte usage, 733
CTD (count down), 337
CTU (count up), 337
CTUD (count up/down), 337
Customer support, 4

D

Data
Receive, 287
Retention, 215
Data block
Password protection, 65
data block (DB), 144
Data log
status (SMB480-SMB515), 1218
DC
Grounding guidelines, 98
Guidelines for isolation, 98
Prerequisites for wiring and grounding, 97
Wiring guidelines, 99
DC inductive loads, 103
Debugging and monitoring
forcing values, 739
program editor status, 733
DEC_B, 389
DEC_DW, 389
DEC_W, 389
DECB (STL-Decrement byte), 389
DECD (STL-Decrement double word), 389
DECO, 332
DECW (STL-Decrement word), 389
Default gateway IP address, 492
Defining
local variables, 166
Device
PROFINET IO device names, 509

Device configuration of CPU and modules, 205
Device view
  Insert module, 520, 808
DI_I, 317
DI_R, 317
DI_S, 326
Diagnostics
  LEDs, 139
Digital input filter, 212
Digital input filter time, 211
Digital inputs, 211
Digital outputs, 214
DIN rail, 90
DISI, 398
Displaying status, 733
DIV, 385
DIV_DI, 381
DIV_I, 381
DIV_R, 381
Downloads
  Example programs, 81
  Program, 127
Drive communication
  calculating time requirement, 636
Drives, 1047, 1077
DSC (Dynamic Servo Control), 770
DTA, 320
DTCH, 398
DTI (STL-Double integer to integer), 317
DTR (STL-Double integer to real), 317
DTS (STL-Double integer to string), 326
Duplicate POU name assignment, 65
dynamic IP information, 493
Dynamic Servo Control (DSC), 770

E

ED (STL-Edge Down), 262
Edge detectors, 262
Editors
  FBD, 155
  LAD, 154
  STL, 155
Electromagnetic compatibility (EMC), 1142
Element usage, 731
EM (expansion module) hardware/firmware ID (SMB1100-SMB1299), 1225
EM DE16 specifications, 1154
ENCO, 332
END, 432
ENI, 398
Environmental
  Conditions of operations, 1143
  industrial environments, 1142
  transport and storage conditions, 1143
Error
  compile and run-time errors (PLC program), 1184
  data retention, 215
  fatal (PLC), 1187
  Fatal error effect on run-time execution, 181
  GET_TABLE and PUT_TABLE instructions, 278
  I/O Module ID and error registers (SMB8-SMB19), 1197
  Memory Locations (PLC Non-Fatal Errors), 1186
  Non-fatal error effect on run-time execution, 180
Error code for axes group, 1138
Error code, Cam and Gear, 1020
Error code, single axis, 1015
Errors
  Axis of Motion, 1098
  character error received from Freeport communication (SMB3), 1195
  I/O error status, 1196
  Modbus RTU master execution, 562
  Modbus RTU slave execution, 568
  Modbus TCP client execution, 577
  Modbus TCP general communication exception codes, 588
  Modbus TCP general exception codes, 587
  Modbus TCP server execution, 579
  Motion instruction, 1099
  PID auto-tune, 758
  PID control, 1217
  PLC information, 172
  PWMx_RUN subroutine, 1042
  signal board ID and error registers (SMB28-SMB29), 1199
  signal board ID and error registers (SMB32-SMB33), 1200
  timestamp mismatch (PC/PLC program difference), 1183
Ethernet
  configuring communication between CPU and HMI device, 502
  GET, 277
  IP Address, 492
  ISO-on-TCP protocol, 503
  MAC address, 500
  number of communication connections, 488
  Ports, 504
  TCP protocol, 503
  TSAPs, 505

types of communication, 32, 487
UDP protocol, 503
Ethernet communications
STEP 7-Micro/WIN SMART V3 settings, 73
Ethernet network
configuring the IP address for a CPU, 493
searching for CPU, 499
EU STL-Edge Up), 262
Events, interrupts, 400
Example
Axes group, 1031
Axes group of motion simple absolute move (pick-and-place application), 1129
bit logic input, 269
GET and PUT instructions, 282
Single Axis, 1028
Subroutine for sampling the value of an analog input, 144
Example S7-200 SMART G2 configuration, 23
Examples
Axis of Motion AXISx_CTRL, AXISx_RUN, AXISx_SEEK, and AXISx_MAN subroutines application, 1087
Axis of Motion simple relative move (cut-to-length application), 1085
bit logic output, 270
Cam, 1033
conversion instructions, 319
count up/down counter instruction, 339
DISCONNECT instruction, 611
high-speed counter initialization sequences, 359
high-speed counter programming, 348
ISO_CONNECT instruction, 597
Modbus RTU slave protocol, programming, 567
Open user communication (OUC) library, 613, 622, 624
scaling analog values, 715
scaling integer to real, 716
scaling real to integer, 718
scaling real to real, 721
Shift register bit (SHRB) instruction, 441
Table Find (TBL_FIND) instruction, 454
tables, 454
TCP_CONNECT instruction, 594
TCP_RECV instruction, 604
UDP_CONNECT instruction, 599
UDP_RECV instruction, 610
UDP_SEND instruction, 607
user-defined library, 715
Using the AXISx_ABSPOS subroutine to read the absolute position from a SINAMICS V90 servo drive, 1079
Executing
program, 111
single or multiple scans, 740
EXP (natural exponential), 387
Expansion and local I/O addressing, 122
Expansion I/O bus - communication errors (SMW98), 1215
Expansion module, 29
EM DE08, 1154
module ID and error registers (SMB8-SMB19), 1197
Expansion module (I/O)
installing and removing, 94
Expansion modules, 1196
module error status (SMB5), 1196
PLC information, 172
Expansion modules (EMs)
Analog input representation (current), 1166
analog input representation (voltage), 1165
Analog output representation (current), 1167
Analog output representation (voltage), 1166
EM AE04 specifications, 1156
EM AE04 wiring diagram, 1158
EM AE08 wiring diagram, 1158
EM AM06 specifications, 1163
EM AM06 wiring diagram, 1164
EM AQ02 specifications, 1159
EM AQ04 wiring diagram, 1160
EM AR0 (RTD) specifications, 1172
EM AR04 (RTD) wiring diagram, 1175
EM AT08 specification, 1168
EM AT08 wiring diagram, 1169
EM DE08 specifications, 1154
EM DE16 wiring diagram, 1154
EM QT16 wiring diagram, 1156
Wiring diagram, 1168

F

Factory defaults memory card, 235
Factory Reset, 69
Fatal error effect on run-time execution, 181
Fatal errors (PLC), 1187
FBD editor, 155
Features
expansion modules supported, 29
Features of the new version, 26
FIFO, 449
File menu
Downloads, 127
Upload, 129
FILL (STL-table fill), 451

FILL_N, 451
Filter times, 211
Filters, digital input configuration, 212
Firmware update, 172
Firmware Update, 132
First scan flag (SMB0), 1193
First scan, executing, 740
Flag memory, 115
FND=, <>, <, > (STL-table find), 452
FOR, 423
Force
  writing and forcing outputs in STOP mode, 740
Forced value indicator (SMB4), 1196
Forgotten password, 55, 235
Freeport mode
  character interrupt control, 295
  Enable, 286
  Example, 540
  Freeport character error (SMB3), 1195
  Freeport configuration (SMB30-Port 0, SMB31-Port2 and SMB130-Port 1), 1201
  Freeport receive character (SMB2), 1195
  Freeport transmitter idle (SMB4), 1196
  interrupts, 404
  SMB405-SMB413 receive message control, 1211
  SMB86-SMB94 and SMB186-SMB194 Receive message control, 1211
Freeze outputs
  analog output configuration, 223
  digital output configuration, 214
Function Block Diagram editor, 155
function block instructions
  CALL, RET, 474
Function block instructions
  Call parameter and return examples, 476

G

General technical specifications, 1141
GERR (STL-Get non-fatal error code), 433
GET, 277
GET_ADDR, 297
GET_ERROR, 433
GIP_ADDR, 298
GRPx_RESET
  ErrorStop, 1127
GSDML, 509
Guidelines
  Earth, 98
  installation procedures, 90
  Isolation, 98
  Prerequisites for wiring and grounding, 97
  Wiring, 99
Guidelines for isolation, 98

H

Hardware configuration
  Adding a module, 520, 808
Hardware Network Topology
  Device view, 796
  Function, 510, 791
Hardware troubleshooting, 742
HDEF (high-speed counter definition), 344
Heat, high voltage, and electrical noise, 84
High-speed counters, 348
High-vibration environment, 84
HMI
  configuring Ethernet communication, 502
  multi-master and multi-slave PPI networks, 531
  single-master PPI networks, 531
Hotline, 4
HSC (High Speed Counter), 235
HSC (high-speed counter), 344
HTA, 320

I

I/O
  Analog input representation (current), 1166
  analog input representation (voltage), 1165
  Analog output representation (current), 1167
  Analog output representation (voltage), 1166
  analog status indicators, 140
  step response times (SM), 1165
I/O addressing, 122
I/O error status
  PLC non-fatal error codes, 1184
  PLC non-fatal error SM flags, 1186
  SMB5, 1196
I/O expansion bus - communication errors (SMW98), 1215
I/O Module ID and error registers (SMB8-SMB19), 1197
I_B, 317
I_BCD, 317
I_DI, 317
I_S, 326
IBCD (STL-Integer to BCD), 317

I-Device, 506
I-Device with lower-level PROFINET IO system, 507
I-Device without lower-level PROFINET IO system, 507
I-Device with lower-level PROFINET IO system, 526
IEC_TOF (Off-delay timer), 464
IEC_TON (On-delay timer), 464
IEC_TONR (Retentive on-delay timer), 464
Immediate I/O read/writes, 111
Immediate instructions
LAD, FBD, and STL, 256
INC_B, 389
INC_DW, 389
INC_W, 389
INCB (STL-Increment byte), 389
INCD (STL-Increment double word), 389
Incremental jog mode, 687
INCW (STL-Increment word), 389
indirect addressing
creating pointers and using indirect address, 123
example of pointer offset to access data, 126
example of pointer to access data in a table, 125
Indirect addressing
Global variable table, 162
Inductive load, 103
Input
Bit logic example, 269
Input filter time, 211
Input process image register, 110
inputs
ALTP contact, 267
Inputs
ALT contact, 267
edge detectors, 262
NOT contact, 261
physical and in program, 109
pulse catch bits (system block), 212
reading, 110
standard contacts, 255, 256
STL logic stack, 258
Installation
airflow, clearance, cooling, and thermal zones, 89
clearance for airflow and cooling, 85
Dimensions, 89
DIN rail, 90
Expansion module (I/O), 94
Grounding guidelines, 98
Guidelines, 83
Guidelines for isolation, 98
High-vibration environment, 84
Inductive load, 103
Lamp load, 102
Mounting dimensions, 89
Overview, 83, 90
Prerequisites for wiring and grounding, 97
Separate the devices from heat, high voltage, and electrical noise, 84
Signal board (SB), 93
thermal zone, 89
Wiring guidelines, 99
Instruction
quick reference guide, 1238
Instruction execution status bits (SMB1), 1194
Instructions
GET, 277
GET_ADDR, 297
GIP_ADDR, 298
loop control (PID), 397
MBUS_CLIENT, 574
MBUS_SERVER, 578
PUT, 277
SET_ADDR, 297
SIP_ADDR, 298
Insulation, 1144
Interrupt routine
Elements of the user program, 144
Interrupt routines, 111
Interrupts
attach/detach, enable/disable, conditional return, and clear event instructions, 398
event support by CPU model, 400
Example programs, 405
global interrupt enable state (SMB4), 1196
interrupt queue overflow (SMB4), 1196
nesting subroutines/FBs, 468, 475
Overview, 400
priority and queuing, 405
programming guidelines, 402
time interval values for timed interrupts (SMB34-SMB35), 1202
types of interrupt events, 404
INV_B, 413
INV_DW, 413
INV_W, 413
INVB (STL-invert byte), 413
INVD (STL-invert double word), 413
INVW (STL-invert word), 413
IP address
assigning, 491
MAC address, 500
IP Address, 492
Assignment, 498
Configuration, 493
IP router, 493

ITA, 320
ITB (STL-Integer to byte), 317
ITD (STL-Integer to double integer), 317
ITS (STL-Integer to string), 326

J

JMP, 424
Jog mode, 684

L

L memory, 163
LAD editor
  features, 154
Lamp load, 102
LBL, 424
LD (STL stack-Load NOT immediate), 258
LD (STL stack-Load), 258
LD (STL-Load), 255
LDB< (STL-Load compare byte less than), 312
LDB<= (STL-Load compare byte less than or equal), 312
LDB<> (STL-Load compare byte not equal), 312
LDB= (STL-Load compare byte equal), 312
LDB> (STL-Load compare byte greater than), 312
LDB>= (STL-Load compare byte greater than or equal), 312
LDD< (STL-Load compare double word less than), 312
LDD<= (STL-Load compare double word less than or equal), 312
LDD<> (STL-Load compare double word not equal), 312
LDD= (STL-Load compare double word equal), 312
LDD> (STL-Load compare double word greater than), 312
LDD>= (STL-Load compare double word greater than or equal), 312
LDI (STL stack-Load immediate), 258
LDI (STL-Load NOT immediate), 256
LDN (STL stack-Load NOT), 258
LDN (STL-Load NOT), 255
LDNI (STL-Load NOT immediate), 256
LDR< (STL-Load compare real less than), 312
LDR<= (STL-Load compare real less than or equal), 312
LDR<> (STL-Load compare real not equal), 312
LDR= (STL-Load compare real equal), 312
LDR> (STL-Load compare real greater than), 312
LDR>= (STL-Load compare real greater than or equal), 312
LDS (STL stack-Load), 259
LDS<> (STL-Load string compare not equal), 316
LDS= (STL-Load string compare equal), 316
LDW< (STL-Load compare word less than), 312
LDW<= (STL-Load compare word less than or equal), 312
LDW<> (STL-Load compare word not equal), 312
LDW= (STL-Load compare word equal), 312
LDW> (STL-Load compare word greater than), 312
LDW>= (STL-Load compare word greater than or equal), 312
LEDs
  CPU status, 139
Library
  adding from third party, 712
  creating user-defined library, 722
  Delete, 712
  including in project, 712
  types, 547
  User-defined, 707
LIFO, 449
LN (natural logarithm), 387
Local and expansion I/O addressing, 122
local station (this CPU)
  creating pointers and using indirect address, 123
  example of pointer offset to access data, 126
  example of pointer to access data in a table, 125
local variable table, 163
Local variables, 163
Local/Partner connection, 489
Locks
  Project tree, 65
Logic stack
  STL inputs, 258
  STL stack operations, 259
Logic, control, 109
Logical operation instructions
  A, 334
  Absolute, 395
  AND, OR, XOR (byte, word, and dword), 414
  invert, 413
  Low-pass filter, 396
  NAND, NOR, NXOR (byte, word, and dword), 414
Loop control (PID)
  loop definition table, 749
Lost password, 235
LPF, 396
LPF (STL-low pass filter), 396
LPP (STL stack-Logic Pop), 259
LPS (STL stack-Logic Push), 259
LRD (STL stack-Logic Read), 259
LSCR (STL-Load SCR), 425

M

MAC address, 500
Main program, 143
Math instructions
  add, subtract, multiply and divide, 381
  divide integer with remainder, 385
  increment and decrement, 389
  multiply integers to double integer and divide integer with remainder, 385
  numeric functions, 387, 390
MBUS_CLIENT, 574
MBUS_CTRL (initialize Modbus master communication), 558
MBUS_INIT (initialize slave communication), 565
MBUS_MSG / MB_MSG2 (send message from Modbus master), 559
MBUS_SERVER, 578
MBUS_SLAVE (slave response to master message), 567
MC_AbortTrigger, 912
MC_CamIn, 970
MC_CamOut, 985
MC_GearIn, 989
MC_GearOut, 993
MC_GroupDisable, 1006
MC_GroupEnable, 1005
MC_GroupReset, 1007
MC_GroupStop, 1008
MC_Halt, 922
MC_Home, 915
MC_ImmediateStop, 944
MC_MoveAbsolute, 928
MC_MoveJog, 947
MC_MoveLinearAbsolute, 1009
MC_MoveLinearRelative, 1011
MC_MoveRelative, 933
MC_MoveVelocity, 940
MC_PhasingRelative, 997
MC_Power, 906
MC_Reset, 913
MC_Stop, 920
MC_TorqueControl, 952
MC_TorqueLimiting, 967
MC_TorqueRange, 960
MC_TouchProbe, 908
Memory, 1186
  Addresses for Non-Fatal Error Indicators, 1186
  clearing PLC, 233
  retentive range configuration, 215
Memory card
  program transfer card, 134, 135
  reset to factory defaults, 235
  Type, 130
  Use, 131
MIN_MAX_B, 390
MIN_MAX_DI, 390
MIN_MAX_I, 390
MIN_MAX_R, 390
Modbus general
  addressing, 550
  advanced user information, 570, 585
  initialization and execution time for Modbus protocol, 555
  library features, 552
  requirements for using Modbus instructions, 553
Modbus RTU library overview, 547
Modbus RTU master
  example program, 568
  execution error codes, 562
  MBUS_CTRL (initialize master communication), 558
  MBUS_MSG / MB_MSG2 (send message from master), 559
  using the instructions, 556
Modbus RTU slave
  execution error codes, 568
  MBUS_INIT (initialize slave communication), 565
  MBUS_SLAVE (slave response to master message), 567
  using the instructions, 564
Modbus TCP
  general exception codes, 587, 588
MODBUS TCP
  MBUS_CLIENT, 574
  MBUS_SERVER, 578
Modbus TCP client
  execution error codes, 577
Modbus TCP library
  example, 580
Modbus TCP library features, 572
Modbus TCP library overview, 547
Modbus TCP server
  execution error codes, 579
Module
  CPU ST32, 1146
  EM AE04, 1156
  EM AM06, 1163
  EM AQ02, 1159
  EM AR04 (RTD), 1172
  EM DE08, 1154
  EM DE16 specifications, 1154

EM QT16, 1155
SB CM01, 1176
thermal zone, 89
Motion control
configuring reference point and seek parameters, 1060
configuring the Backlash compensation, 1059
configuring the input pin locations, 1050
defining the motion profile, 1062
entering acceleration time, 1057
entering jerk time, 1058
entering jog parameters, 1057
entering maximum start and stop speed, 1056
mapping the I/O, 1051
Motion control resources, 218
Motion features, 1046
phasing, 1053
polarity, 1055
Motion control panel
displaying and controlling the operation of axes, 1092
displaying and modifying the configuration of axes, 1097
displaying the profile configuration for the axes, 1097
Motion inputs and outputs
CPU, 1047
Motion instruction, error codes, 1099
Motion profile
configuring, 1044
creating steps, 1046
defining, 1045
mode of operation, 1045
Motion wizard
configuration/profile table, 1101
maximum and start/stop speeds, 1043
open loop motion control, 1039
SM locations, 1108
MOV_B, 419
MOV_BIR, 422
MOV_BIW, 422
MOV_DW, 419
MOV_R, 419
MOV_W, 419
MOVB (STL-move byte), 419
MOVD (STL-move double word), 419
Move instructions
block move (byte, word, dword), 420
move (byte, word, dword, real), 419
move byte immediate (read and write), 422
SWAP (exchange byte data in a word), 421
MOVR (STL-move real), 419
MOVW (STL-move word), 419
MUL, 385
MUL_DI, 381
MUL_I, 381
MUL_R, 381
Multiple scans, 740

N

NAND_B, 414
NAND_DW, 414
NAND_W, 414
NANDB (STL-NAND byte), 414
NANDD (STL-NAND dword), 414
NANDW (STL-NAND word), 414
Nested subroutines/FBs, 468, 475
Network communication, 72
Networks (communication)
addresses, 528
biasing and terminating the network cable, 538
calculating network distances, 535
general guidelines for building, 535
network configurations, 526
safety concerns, 535
Sample RS485 network configurations, 531
selecting the network cable, 537
single-master PPI, 531
types of communication, 32, 487
New features, 26
NEXT, 423
Non-fatal error effect on run-time execution, 180
Non-fatal PLC errors
compile and run-time, 1184
Special Memory locations, 1186
Non-volatile memory, 130
NOP, 266
NOR_B, 414
NOR_DW, 414
NOR_W, 414
NORD (STL-NOR dword), 414
Normally closed contact
immediate, 256
standard, 255
Normally open contact
immediate, 256
standard, 255
NORW (STL-NOR word), 414
NOT (STL), 261
Number value to string conversion instructions, 326
NXOR_B, 414
NXOR_DW, 414
NXOR_W, 414

NXORB (STL-NXOR byte), 414
NXORD (STL-NXOR dword), 414
NXORW (STL-NXOR word), 414

O

O (STL-OR), 255
OB< (STL-OR compare byte less than), 312
OB<= (STL-OR compare byte less than or equal), 312
OB<> (STL-OR compare byte not equal), 312
OB= (STL-OR compare byte equal), 312
OB> (STL-OR compare byte greater than), 312
OB>= (STL-OR compare byte greater than or equal), 312
OB1, 143
OD< (STL-OR compare double word less than), 312
OD<= (STL-OR compare double word less than or equal), 312
OD<> (STL-OR compare double word not equal), 312
OD= (STL-OR compare double word equal), 312
OD> (STL-OR compare double word greater than), 312
OD>= (STL-OR compare double word greater than or equal), 312
OI (STL-OR immediate), 256
OLD (STL stack-OR Load), 259
ON (STL-OR NOT), 255
ONI (STL-OR NOT immediate), 256
Open loop control, 1039
Open Source Software, 38
  Use, 39
Open user communication (OUC)
  connection instructions, 504
  connection types, 504
Open user communication (OUC) library
  common library instruction parameters, 590
  DISCONNECT instruction, 610
  instruction error codes, 612
  ISO_CONNECT instruction, 595
  overview, 547
  TCP_CONNECT instruction, 592
  TCP_RECV instruction, 602
  TCP_SEND instruction, 600
  UDP_CONNECT instruction, 598
  UDP_RECV instruction, 608
  UDP_SEND instruction, 605
Opening earlier STEP 7-Micro/WIN projects, 145
Operating mode
  changing to STOP, 82, 138
  read PLC information, 172
  startup options, 217
Operator stations, 141
Options
  STL status, 736
OR< (STL-OR compare real less than), 312
OR<= (STL-OR compare real less than or equal), 312
OR<> (STL-OR compare real not equal), 312
OR= (STL-OR compare real equal), 312
OR> (STL-OR compare real greater than), 312
OR>= (STL-OR compare real greater than or equal), 312
ORB (STL-NOR byte), 414
ORB (STL-OR byte), 414
ORD (STL-OR dword), 414
Ordering information, 1249
ORW (STL-OR word), 414
OS<> (STL-OR string compare not equal), 316
OS= (STL-OR string compare equal), 316
OUC library example
  Active partner (client) program, 613
  CheckErrors subroutine program, 622
  Passive partner (server) program, 624
OUC library example Active partner (client) program
  variable table, 623
OUC library example Passive partner (server) program
  Variable table, 631
Output image register, 110
Outputs
  coils, 263
  example bit logic, 270
  physical and in program, 109
  set and reset bits, 264
  set and reset dominant bistable, 265
  writing, 111
Overlapping
  bytes, 733
Overvoltage, 1144
OW< (STL-OR compare word less than), 312
OW<= (STL-OR compare word less than or equal), 312
OW<> (STL-OR compare word not equal), 312
OW= (STL-OR compare word equal), 312
OW> (STL-OR compare word greater than), 312
OW>= (STL-OR compare word greater than or equal), 312

P

Part number, 1249
Password
  lost or forgotten, 235
  privilege levels, 55
Password protection, 55
  POUs and data blocks, 65
  projects, 64

PID auto-tune
exception conditions, 758
prerequisites, 756
PV out-of-range, 759
sequence, 756
PID loop control
loop definition table, 749
PID Tune control panel, 760
PID Tune control panel, 760
Pin assignments for network connector, 537
Pipelining
PTO pulses, 372
Placing the CPU in RUN mode RUN mode,
Plant
Insert, 802
Move, 801
Swap, 802
PLC
clearing memory, 233
compile and run-time errors, 1184
fatal errors, 1187
Installation, 90
Memory card, 134
Non-Fatal Error Memory Locations, 1186
System block, 205
viewing PLC information, 172
PLC menu
Downloads, 127
Upload, 129
PLS
instruction, 370
Override, 371
Special Memory to monitor and control PTO and PWM outputs, 1205
PN IRT axis
Diagnostics, 862
PN Read Write library
PN_RD_REC instruction, 632
PN_WR_REC instruction, 632
PN Read Write Record library
overview, 547
Pollution degree, 1144
Position loop gain (Kv factor), 771
POU
Password protection, 65
viewing properties, 65
Power interruption (PLC), 215
Power requirements
Calculating, 1181
CPU, 87, 1179
Power supply, 87, 1179
PPI communication
changing to Freeport mode, 286
multi-master and multi-slave PPI networks, 531
port configuration in system block, 52, 209
single-master networks, 531
Prerequisites
Using SINA_PARA_S instruction, 698
Using SINA_POS instruction, 653
Using SINA_SPEED instruction, 692
Previous STEP 7-Micro/WIN projects, 145
Profile table values
PTO generators, 377
PROFINET device status
SMB1800-SMB1999, 1232
PROFINET instruction
BLKMOV_BIR, 485
BLKMOV_BIW, 485
BMIR, 485
BMIW, 485
RDREC, 481
WRREC, 481
PROFINET IO
Assigning device names, 509
device names, 509
Program
Element, 143
Interrupt routine, 144
Memory card, 134, 135
Subroutines, 143
Program Blocks, 143
Program control instructions
END, STOP, and WDR, 432
FOR-NEXT loop, 423
GET_ERROR, 433
JMP-LBL, 424
SCR (Sequence control relay), 425
Program editor
bookmarks, 731
debugging and monitoring, 733
STL status options, 736
Type, 153
using, 77
Program instruction
bit logic, 267
Program instructions
bit logic, 255, 256, 258, 259, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270
clock, 272, 274
compare, 312, 316
convert, 317, 320, 326, 330, 332
counters, 337, 344, 348, 359
frequency mode, 366

function block, 474
Function block, 476
interrupts, 398
logical operations, 334, 395, 396, 413, 414
math, 381, 385, 387, 389, 390
move, 419, 420, 421, 422
program control, 423, 424, 425, 432, 433
shift and rotate, 435, 440
string, 444, 445
subroutine, 468, 470
table, 448, 449, 451
table find, 452
timer, 456, 467
Timer, 464
Transfer, 418
Program status
building a status chart, 737
displaying in program editor, 733
executing a limited number of scans, 740
program transfer card, 131
Programs
executing limited scans, 740
status charts, 737
Project
Migrate, 145
Projects
password protection, 64
Properties
POUs and DBs, 65
Protection class, 1144
PTO
overriding, 372
PTO0, PWM0, PTO1, PWM1, PTO2, PWM2, PTO3 and PWM3 high-speed outputs (SMB66-SMB85, SMB166-SMB169, SMB176-SMB179, SMB566-SMB579 and SMB580-SMB593), 1205
Pulse catch bits, digital input configuration in system block, 212
Pulse train output (PTO)
cycle time, 371
Instructions, 236
pulse output instruction (PLS), 370
pulse width modulation (PWM)
Instructions, 236
Pulse width modulation (PWM)
cycle time, 373
output, 1040, 1041
pulse output instruction (PLS), 370
pulse outputs, 1042
PUT, 277
PWM wizard
PWMx_RUN subroutine, 1042
PWMx_RUN, 1042

Q

Queue interrupt overflow (SMB4), 1196
Quick access toolbar, 33

R

R (STL-Reset), 264
R_S, 326
RAND_B, 392
RAND_DI, 392
RAND_I, 392
RAND_R, 392
Rated voltage, 1144
RCV (receive message control SMB86-SMB94, SMB405-SMB413 and SMB186-SMB194), 1211
READ_RTC, 272
READ_RTCX, 274
Real number values, 121
Receive instruction
Break detection, 292
end character detection, 293
end conditions, 290
Idle line detection, 290
intercharacter timer, 293
maximum character count, 294
message timer, 294
Parity Error, 294
Start character detection, 291
user termination, 294
Recursion in subroutines/FBs, 468, 475
Recycling and disposal, 69
Referencing (active referencing) mode, 675
Referencing (set reference point) mode), 678
Relative positioning mode, 663
Relay electrical service life, 1145
Repeaters, 536
Reset-to-factory-defaults memory card, 235
Restoring data after power-on, 138
RET, 468, 474
Retentive memory, 130
Retentive ranges, system block configuration, 215
RETI, 398
RI (STL-Reset immediate), 264
ROUND, 317
RS (LAD/FBD Reset dominant bistable), 265
RS232, 545
Freeport mode, 541

number of communication connections, 488
types of communication, 32, 487
RS232/PPI cable, 541
RS485
communication overview, 526
communication ports configuration, 52, 209
number of communication connections, 488
Sample Network Configurations, 531
setting baud rate and port network address, 529
types of communication, 32, 487
RS485 address
Assignment, 534
RS485 network
Address, 532
configuring the RS485 network address for a CPU, 532
RS485 network address
Configuration, 532
RTA, 320
RTD analog inputs
Coefficient, 225
Rejection, 225
Resistor, 225
RTD types, 225
Scale, 225
Smoothing, 225
system block configuration, 225
RTS (STL-Real to string), 326
RUN mode, 81, 138
RUN to STOP transition
Analog output states, 223
digital output states, 214
Run-time and PLC compile errors, 1184

S

S (STL-Set), 264
S_DI, 330
S_I, 330
S_ITR, 334
S_R, 330
S_RTI, 334
S_RTR, 334
S7-200 SMART G2
Configuration, 23
S7-200 SMART motion control
Closed-loop control, 770
S7-200 SMART security
Network security, 47
Safety circuits, 141
Safety information, 86
Sample control program, 76
Sample network configurations, RS485 devices, 531
Saving project, 80
SB (signal board) hardware/firmware ID (SMB1050-SMB1099), 1224
Scaling
integer to real, 716
real to integer, 718
real to real, 721
Scan cycle
executing a single scan, 740
executing multiple scans, 740
Scan rates (PLC information), 172
scan times (SMW22-SMW26), 1199
SCR, 425
SCRE, 425
SCRT, 425
Security, 55
SEG, 317
Selecting the network cable, 537
Service&Support, 4
Set and reset dominant bistable instructions, 265
Set and reset immediate instructions, 264
SET_ADDR, 297
SET_RTC, 272
SET_RTCX, 274
Setting the CPU type, 79
Setup mode, 671
SFND (STL-Find string), 445
Shielding connection and grounding guidelines, 98
Shift and rotate instructions
bit (SHRB), 440
byte, word, dword, 435
SHL_B, 435
SHL_DW, 435
SHL_W, 435
SHR_B, 435
SHR_DW, 435
SHR_W, 435
SHRB, 440
SI (STL-set immediate), 264
Siemens technical support, 4
Siemens-supplied libraries, 547
Signal board (SB)
Analog output representation (current), 1167
Analog output representation (voltage), 1166
input representation (current), 1166
input representation (voltage), 1165
installing and removing, 93
SB CM01 specifications, 1176
SB CM01 wiring diagram, 1177
Signal board ID and error registers (SMB28-SMB29), 1199

Signal board ID and error registers (SMB32-SMB33), 1200
signal modules (SM)
  step response times, 1165
SIN (sine), 387
SINA_PARA_S instruction
  Example, 704
SINA_POS instruction
  Absolute positioning mode example, 668
  Incremental jog mode example, 688
  Jog mode example, 685
  operating mode selection, 662
  Referencing (active referencing) mode example, 676
  Referencing (set reference point) mode example, 679
  Relative positioning mode example, 664
  Setup mode example, 673
  Traversing blocks mode example, 682
SINA_SPEED instruction
  example, 696
SINAMICS library
  overview, 547
  SINA_PARA_S instruction, 700
  SINA_POS instruction, 655
  SINA_SPEED instruction, 694
  SINAMICS_Control, 651
  SINAMICS_Parameter, 652
SINAMICS library features, 651
SIP_ADDR, 298
SLB (STL-Shift left byte), 435
SLD (STL-Shift left dword), 435
SLW (STL-Shift left word), 435
SM (Special memory) assignments and functions, 1191
SM locations (Axis of Motion), 1108
SM memory, PTO/PWM operation, 374
SMB0 system status bits, 1193
SMB1 instruction execution status bits, 1194
SMB1000-SMB1049 CPU hardware/firmware ID, 1224
SMB1050-SMB1099 SB (signal board) hardware/firmware ID, 1224
SMB1100-SMB1299 EM (expansion module) hardware/firmware ID, 1225
SMB130 Port 1 configuration, 1201
SMB136-SMB145 (HSC3) high-speed counter 3, 1203
SMB146-SMB155 (HSC4) high-speed counter 4, 1203
SMB156-SMB165 (HSC5) high-speed counter 5, 1203
SMB1800-SMB1999 PROFINET device status, 1232
SMB186-SMB194 receive message control, 1211
SMB195 to SMB210
  HSC, 1217
SMB2 Freeport receive character, 1195
SMB2000-SMB2014: PTO4/PWM4 high-speed outputs, 1205
SMB28-SMB29 signal board ID and error registers, 1199
SMB3 Freeport character error, 1195
SMB30 (Port 0), SMB31(Port2) and SMB130 (Port 1) configuration, 1201
SMB32-SMB33 signal board ID and error registers, 1200
SMB34-SMB35 time interval values for timed interrupts, 1202
SMB360-SMB369 (HSC6) high-speed counter 6, 1203
SMB36-SMB45 (HSC0) high-speed counter 0, 1203
SMB370-SMB379 (HSC7) high-speed counter 7, 1203
SMB380 to SMB404
  HSC, 1218
SMB4 interrupt queue overflow, run-time program error, interrupts enabled, Freeport transmitter idle, value forced, 1196
SMB46-SMB55 (HSC1) high-speed counter 1, 1203
SMB480-SMB515 Data log status, 1218
SMB5 I/O error status bit, 1196
SMB566-SMB579: PTO2 and PWM2 high-speed outputs, 1205
SMB56-SMB65 (HSC2) high-speed counter 2, 1203
SMB580-SMB593: PTO3 and PWM3 high-speed outputs, 1205
SMB580-SMB593: PTO3/PWM3 high-speed outputs, 1205
SMB600-SMB799 Axis 0,1,2 and 3 motion control, 1219
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, 1205
SMB6-SMB7 CPU ID register, 1197
SMB86-SMB94 and SMB186-SMB194 Receive message control, 1211
SMB8-SMB19 I/O module ID and error registers, 1197
SMB929, SMB931, SMB933, SMB935 trace control, 1223
SMB930, SMB932, SMB934, SMB936 trace status, 1223
SMB950-SMB998 SB (signal board) hardware/firmware ID, 1224
SMW100-SMW120 System alarms, 1215
SMW22-SMW26 scan times, 1199

SMW98 Expansion I/O bus - communication errors, 1215
Software debugging, 731
SORT_B, 393
SORT_DI, 393
SORT_I, 393
SORT_R, 393
Special memory assignments and functions, 1191
Specifications
  Ambient conditions, 1143
  Analog input representation (current), 1166
  analog input representation (voltage), 1165
  Analog output representation (current), 1167
  Analog output representation (voltage), 1166
  CE approval, 1141
  CPU ST32, 1146
  Electromagnetic compatibility (EMC), 1142
  EM AE04, 1156
  EM AM06, 1163
  EM AQ02, 1159
  EM AR04 (RTD), 1172
  EM AT08, 1168
  EM QT16, 1155
  General technical specifications, 1141
  industrial environments, 1142
  Insulation, 1144
  Overvoltage, 1144
  Pollution degree, 1144
  Protection, 1144
  Rated voltage, 1144
  Relay electrical service life, 1145
  SB CM01, 1176
  step response times (SM), 1165
  UK Conformity Assessed marking, 1141
Speed loop response time, 771
Speed precontrol, 771
Speedy Splitter, 511, 792, 796
SQRT (square root), 387
SR (LAD/FBD Set dominant bistable), 265
SRB (STL-Shift right byte), 435
SRD (STL-Shift right dword), 435
SRW (STL-Shift right word), 435
SSCPY (STL-Copy substring), 444
SSTR_CPY, 444
Starting
  startup options, 217
Starting Ethernet communications
  STEP 7-Micro/WIN SMART, 73
Statement List editor (STL), 155
static IP information, 493
Status
  building a status chart, 737
  displaying in program editor, 733
  executing a limited number of scans, 740
Status error (timestamp mismatch), 1183
Status LEDs
  CPU, 139
  Expansion modules (EMs), 139
  PROFINET network, 509
STD (STL-Sub-string to double integer), 330
STEP 7-Micro/WIN (earlier version), 145
STEP 7-Micro/WIN SMART
  connecting with the CPU, 75, 494
  Ethernet port configuration, 493
  Requirements for the equipment, 33
  RS485 network port configuration, 532
  RUN and STOP mode, 81, 138
Steppers in motion control, 1042
STI (STL-Sub-string to integer), 330
STL
  logic stack operations, 259
  status options, 736
STL editor, 155
STOP (instruction), 432
STOP mode, 82, 138
  Analog output states, 223
  digital output states, 214
  writing and forcing outputs, 740
STR (STL-Sub-string to real), 330
STR_FIND, 445
String
  Format, 121
  Representation, 122
String instructions
  copy substring, 444
  Find string / character, 445
SUB_DI, 381
SUB_I, 381
SUB_R, 381
Subroutine instructions
  Call parameter and return examples, 470
  CALL, RET, 468
Subroutines
  Axes group of Motion, 1122
  Axis of Motion, 1063
  element of user program, 143
  guidelines, 1064
  PWMx_RUN, 1042
Subroutines/FBs
  nesting depth, 468, 475
  total number, 468, 475
Support, 4
Suppression circuits, 103
SWAP, 421

Synchronous updates (PWM instruction), 374
System alarms (SMW100-SMW120), 1215
System block, 144
  CPU configuration, 205
  Digital input filter, 212
  IP address of CPU, 495
  password and security, 55
  RS485 network address of CPU, 533
  RS485/RS232 CM01 communications signal board, 232
  RTD analog input module, 225
  startup options, 217
  TC analog input module, 229
System status bits (SMB0), 1193

T

Table instructions
  ATT, 448
  FIFO/LIFO, 449
  FILL_N, 451
  TBL_FIND, 452
TAN (tangent), 387
TBL_FIND, 452
TC analog input module, 229
TC analog inputs
  Rejection, 229
  Scale, 229
  Smoothing, 229
  system block configuration, 229
  TC type, 229
Technical specifications, 1141
Technical Support, 4
thermal zone, 85, 89
Thermocouple
  Basic operations, 231, 1170
  Cold junction compensation, 231, 1170
  EM AT04 Thermocouple filter selection table, 1170
  EM AT04 Thermocouple selection table, 1170
Timed interrupt configuration (SMB34-SMB35), 1202
Time-of-day
  clock instructions, 272
  extended clock instructions, 274
  protection for reads and writes, 55
Timer instructions
  BITIM, CITIM, 467
  IEC_TON, IEC_TONR, IEC_TOF, 464
  interrupts, 404
  programming tips and examples, 458, 465
  TON, TONR, TOF, 456
Timestamp mismatch (PC/PLC program difference), 1183
TODR (STL-Read time-of-day clock), 272
TODRX (STL-Read time-of-day clock extended), 274
TODW (STL-Write time-of-day clock), 272
TODWX (STL-Write time-of-day clock extended), 274
TOF (Off-delay timer), 456
TON (On-delay timer), 456
TONR (On-delay timer retentive), 456
Tools (options)
  execution status coloring, 733
  STL status, 736
Tools menu
  PID Tune control panel, 760
Transfer card, 131
Transfer instructions
  move (byte, word, dword, real, UDT, array, string, FB instance), 418
Transmission rate, 535
Transmit instruction
  Example, 296
  Transmit Data, 287
Traversing blocks mode, 681
Troubleshooting S7-200 SMART G2 hardware, 742
Trouobleshooting
  LEDs, 139
TRUNC, 317

U

UK Conformity Assessed marking, 1141
Updating firmware, 172
Uploading programs, 129
User reference, 150
user-defined library
  programming tips, 707
User-defined library
  example, 715
USS protocol instructions
  example program, 648
  using, 637
  USS_CTRL, 640
  USS_INIT, 638
  USS_RPM_x, 643
  USS_WPM_x, 646
USS protocol library
  calculating time for communications, 636
  execution errors, 650
  overview, 547, 635
  requirements, 636
  using the USS protocol instructions, 637

V

V90 drives, 1047, 1077
variables (variable addressing)
  Indirect addressing, 162
Version (firmware), 172
Vibrations, 84

W

WAND_B, 414
WAND_DW, 414
WAND_W, 414
WDR (watchdog timer reset), 432
Web API, 252
Windows
  floating, docking, and combining, 150
Wiring diagram
  CPU ST32, 1152
  EM AE04, 1158
  EM AE08, 1158
  EM AM06, 1164
  EM AQ04, 1160
  EM AR04 (RTD), 1175
  EM AT08, 1168, 1169
  EM DE16, 1154
  SB CM01, 1177
Wiring guidelines, 99
  clearance for airflow and cooling, 85
  DIN rail, 90
  Earth, 98
  Inductive load, 103
  Installation, 83
  Lamp load, 102
  Prerequisites, 97
  Separate the devices from heat, high voltage, and electrical noise, 84
Wizards
  high-speed counter (HSC), 348
WOR_B, 414
WOR_DW, 414
WOR_W, 414
Word access, 114
Writing values
  outputs, 111
  writing and forcing outputs in STOP mode, 740
WXOR_B, 414
WXOR_DW, 414
WXOR_W, 414

X

XORB (STL-XOR byte), 414
XORD (STL-XOR dword), 414
XORW (STL-XOR word), 414

Z

Zoom
  Adjusting the zoom setting, 511, 792