SIEMENS

SIMATIC

S7
S7-1200 Programmable controller

System Manual

01/2015

A5E02486680-AH

Legal information

Warning notice system

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

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

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

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

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

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

Qualified Personnel

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

Proper use of Siemens products

Note the following:

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

Trademarks

All names identified by ® are registered trademarks of Siemens AG. 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 AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E02486680-AH
℗ 11/2014 Subject to change

Preface

Purpose of the manual

The S7-1200 series is a line of programmable logic controllers (PLCs) that can control a variety of automation applications. Compact design, low cost, and a powerful instruction set make the S7-1200 a perfect solution for controlling a wide variety of applications. The S7-1200 models and the Windows-based STEP 7 programming tool (Page 35) give you the flexibility you need to solve your automation problems.

This manual provides information about installing and programming the S7-1200 PLCs and is designed for engineers, programmers, installers, and electricians who have a general knowledge of programmable logic controllers.

Required basic knowledge

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

Scope of the manual

This manual describes the following products:

- STEP 7 V13 SP1 Basic and Professional (Page 35)
- S7-1200 CPU firmware release V4.1

For a complete list of the S7-1200 products described in this manual, refer to the technical specifications (Page 1099).

Certification, CE label, C-Tick, and other standards

Refer to the technical specifications (Page 1099) for more information.

Service and support

In addition to our documentation, Siemens offers technical expertise on the Internet and on the customer support web site (http://www.siemens.com/tiaportal).

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.

Documentation and information

S7-1200 and STEP 7 provide a variety of documentation and other resources for finding the technical information that you require.

- The S7-1200 Programmable Controller System Manual provides specific information about the operation, programming, and the specifications for the complete S7-1200 product family. In addition to the system manual, the S7-1200 Easy Book provides a more general overview to the capabilities of the S7-1200 family.

  Both the system manual and the Easy Book are available as electronic (PDF) manuals. The electronic manuals can be downloaded from the customer support web site and can also be found on the documentation disk that ships with every S7-1200 CPU.
- The online STEP 7 information system provides immediate access to the conceptual information and specific instructions that describe the operation and functionality of the programming package and basic operation of SIMATIC CPUs.
- My Documentation Manager accesses the electronic (PDF) versions of the SIMATIC documentation set, including the system manual, the Easy Book, and the STEP 7 information system. With My Documentation Manager, you can drag and drop topics from various documents to create your own custom manual.

  The customer support entry portal (http://support.automation.siemens.com) provides a link to My Documentation Manager under mySupport.
- The customer support web site also provides podcasts, FAQs, and other helpful documents for S7-1200 and STEP 7. The podcasts utilize short educational video presentations that focus on specific features or scenarios in order to demonstrate the interactions, convenience, and efficiency provided by STEP 7. Visit the following web sites to access the collection of podcasts:
  - STEP 7 Basic web page (http://www.automation.siemens.com/mcms/simatic-controller-software/en/step7/step7-basic/Pages/Default.aspx)
  - STEP 7 Professional web page (http://www.automation.siemens.com/mcms/simatic-controller-software/en/step7/step7-professional/Pages/Default.aspx)
- You can also follow or join product discussions on the Service & Support technical forum (https://www.automation.siemens.com/WW/forum/guests/Conferences.aspx?Language=en&siteid=csius&treeLang=en&groupid=4000002&extranet=standard&viewreg=WW&nodeid0=34612486). These forums allow you to interact with various product experts.
  - Forum for S7-1200 (https://www.automation.siemens.com/WW/forum/guests/Conference.aspx?SortField=LastPostDate&SortOrder=Descending&ForumID=258&Language=en&onlyInternet=False)
  - Forum for STEP 7 Basic (https://www.automation.siemens.com/WW/forum/guests/Conference.aspx?SortField=LastPostDate&SortOrder=Descending&ForumID=265&Language=en&onlyInternet=False)

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. You can find more information about industrial security on the Internet (http://www.siemens.com/industrialsecurity).

To stay informed about product updates as they occur, sign up for a product-specific newsletter. You can find more information on the Internet (http://support.automation.siemens.com).

Table of contents

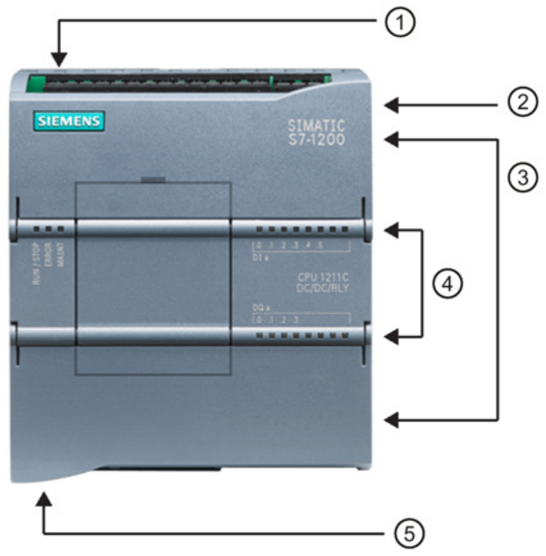
1 Product overview

1.1 Introducing the S7-1200 PLC

The S7-1200 controller provides the flexibility and power to control a wide variety of devices in support of your automation needs. The compact design, flexible configuration, and powerful instruction set combine to make the S7-1200 a perfect solution for controlling a wide variety of applications.

The CPU combines a microprocessor, an integrated power supply, input and output circuits, built-in PROFINET, high-speed motion control I/O, and on-board analog inputs in a compact housing to create a powerful controller. After you download your program, the CPU contains the logic required to monitor and control the devices in your application. The CPU monitors the inputs and changes the outputs according to the logic of your user program, which can include Boolean logic, counting, timing, complex math operations, and communications with other intelligent devices.

The CPU provides a PROFINET port for communication over a PROFINET network. Additional modules are available for communicating over PROFIBUS, GPRS, RS485, RS232, IEC, DNP3, and WDC networks.



①	Power connector
②	Memory card slot under top door
③	Removable user wiring connectors (behind the doors)
④	Status LEDs for the on-board I/O
⑤	PROFINET connector (on the bottom of the CPU)

Several security features help protect access to both the CPU and the control program:

- Every CPU provides password protection (Page 197) that allows you to configure access to the CPU functions.
- You can use "know-how protection" (Page 200) to hide the code within a specific block.
- You can use copy protection (Page 201) to bind your program to a specific memory card or CPU.

Table 1- 1 Comparing the CPU models

Feature		CPU 1211C	CPU 1212C	CPU 1214C	CPU 1215C	CPU 1217C
Physical size (mm)		90 x 100 x 75		110 x 100 x 75	130 x 100 x 75	150 x 100 x 75
User memory	Work	50 Kbytes	75 Kbytes	100 Kbytes	125 Kbytes	150 Kbytes
	Load	1 Mbyte		4 Mbytes		
	Retentive	10 Kbytes				
Local on-board I/O	Digital	6 inputs/4 outputs	8 inputs/6 outputs	14 inputs/10 output		
	Analog	2 inputs			2 inputs/2 output	
Process image size	Inputs (I)	1024 bytes				
	Outputs (Q)	1024 bytes				
Bit memory (M)		4096 bytes		8192 bytes		
Signal module (SM) expansion		None	2	8		
Signal board (SB), Battery board (BB), or communication board (CB)		1				
Communication module (CM) (left-side expansion)		3				
High-speed counters	Total	Up to 6 configured to use any built-in or SB inputs				
	1 MHz	-				Ib.2 to Ib.5
	100/[1]80 kHz	Ia.0 to Ia.5				
	30/[1]20 kHz	--	Ia.6 to Ia.7	Ia.6 to Ib.5		Ia.6 to Ib.1
	200 kHz[3]					
Pulse outputs[2]	Total	Up to 4 configured to use any built-in or SB outputs				
	1 MHz	--				Qa.0 to Qa.3
	100 kHz	Qa.0 to Qa.3				Qa.4 to Qb.1
	20 kHz	--	Qa.4 to Qa.5	Qa.4 to Qb.		--
Memory card		SIMATIC Memory card (optional)				
Real time clock retention time		20 days, typ./12 day min. at 40 degrees C (maintenance-free Super Capacitor)				
PROFINET Ethernet communication port		1			2	
Real math execution speed		2.3 μs/instruction				
Boolean execution speed		0.08 μs/instruction				

1 The slower speed is applicable when the HSC is configured for quadrature mode of operation.

2 For CPU models with relay outputs, you must install a digital signal (SB) to use the pulse outputs.

3 Up to 200 kHz are available with the SB 1221 DI x 24 VDC 200 kHz and SB 1221 DI 4 x 5 VDC 200 kHz.

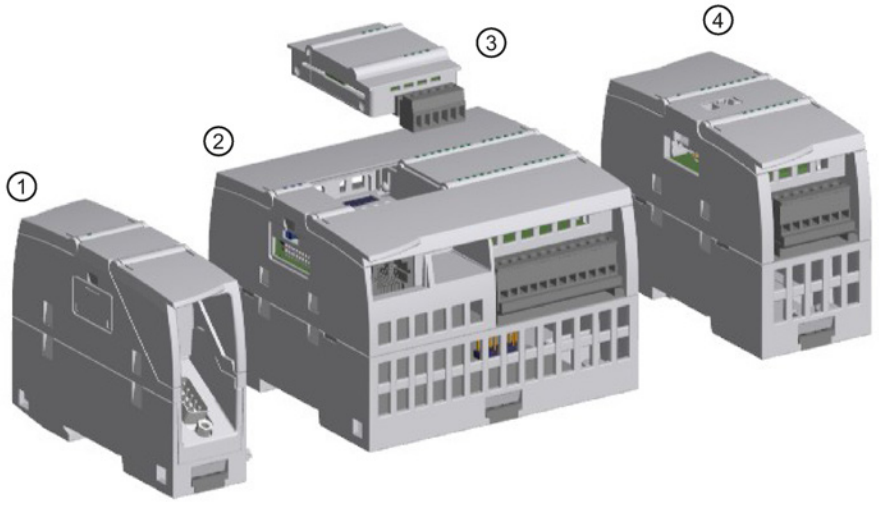
The different CPU models provide a diversity of features and capabilities that help you create effective solutions for your varied applications. For detailed information about a specific CPU, see the technical specifications (Page 1099).

Table 1- 2 Blocks, timers, and counters supported by S7-1200

Element		Description
Blocks	Type	OB, FB, FC, DB
	Size	50 Kbytes (CPU 1211C) 75 Kbytes (CPU 1212C) 100 Kbytes (CPU 1214C) 125 Kbytes (CPU 1215C) 150 Kbytes (CPU 1217C)
	Quantity	Up to 1024 blocks total (OBs + FBs + FCs + DBs)
	Nesting depth	16 from the program cycle or startup OB; 6 from any interrupt event OB
	Monitoring	Status of 2 code blocks can be monitored simultaneously
OBs	Program cycle	Multiple
	Startup	Multiple
	Time-delay interrupts	4 (1 per event)
	Cyclic interrupts	4 (1 per event)
	Hardware interrupts	50 (1 per event)
	Time error interrupts	1
	Diagnostic error interrupts	1
	Pull or plug of modules	1
	Rack or station failure	1
	Time of day	Multiple
	Status	1
	Update	1
	Profile	1
Timers	Type	IEC
	Quantity	Limited only by memory size
	Storage	Structure in DB, 16 bytes per timer
Counters	Type	IEC
	Quantity	Limited only by memory size
	Storage	Structure in DB, size dependent upon count type • SInt, USInt: 3 bytes • Int, UInt: 6 bytes • DInt, UDInt: 12 bytes

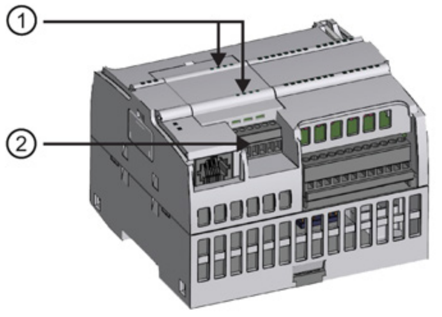
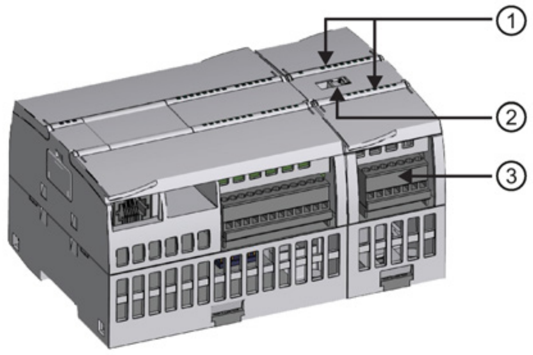
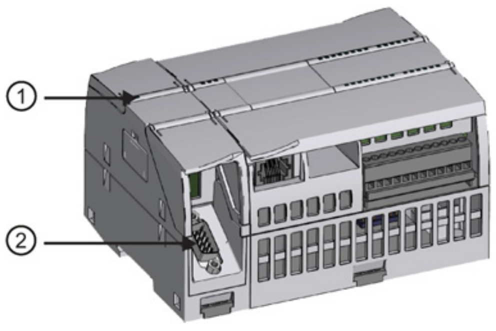
1.2 Expansion capability of the CPU

The S7-1200 family provides a variety of modules and plug-in boards for expanding the capabilities of the CPU with additional I/O or other communication protocols. For detailed information about a specific module, see the technical specifications (Page 1099).



①	Communication module (CM) or communication processor (CP) (Page 1250)
②	CPU (CPU 1211C (Page 1109), CPU 1212C (Page 1119), CPU 1214C (Page 1129), CPU 1215C (Page 1141), CPU 1217C (Page 1155))
③	Signal board (SB) (digital SB (Page 1221), analog SB (Page 1232)), communication board (CB) (Page 1260), or Battery Board (BB) CPU (CPU 1211C, CPU 1212C, CPU 1214C, CPU 1215C, CPU 1217C) (Page 1248)
④	Signal module (SM) (digital SM (Page 1169), analog SM (Page 1185), thermocouple SM (Page 1197), RTD SM (Page 1202), technology SM) (Page 1208)

Table 1- 3 S7-1200 expansion modules

Type of module	Description
The CPU supports one plug-in expansion board: • A signal board (SB) provides additional I/O for your CPU. The SB connects on the front of the CPU. • A communication board (CB) allows you to add another communication port to your CPU. • A battery board (BB) allows you to provide long term backup of the realtime clock.	
	① Status LEDs on the SB
	② Removable user wiring connector
Signal modules (SMs) add additional functionality to the CPU. SMs connect to the right side of the CPU. • Digital I/O • Analog I/O • RTD and thermocouple • SM 1278 IO-Link Master	
	① Status LEDs
	② Bus connector slide tab
	③ Removable user wiring connector
Communication modules (CMs) and communications processors (CPs) add communication options to the CPU, such as for PROFIBUS or RS232/RS485 connectivity (for PtP, Modbus or USS), or the AS-i master. A CP provides capabilities for other types of communication, such as connecting to the CPU over a GPRS, IEC, DNP3, or WDC network. • The CPU supports up to three CMs or CPs • Each CM or CP connects to the left side of the CPU (or to the left side of another CM or CP)	
	① Status LEDs
	② Communication connector

1.3 Basic HMI panels

The SIMATIC HMI Basic Panels provide touch-screen devices for basic operator control and monitoring tasks. All panels have a protection rating for IP65 and have CE, UL, cULus, and NEMA 4x certification.

The available Basic HMI panels (Page 1282)are described below:

- KTP400 Basic: 4" Touch screen with 4 configurable keys, a resolution of 480 x 272 and 800 tags
- KTP700 Basic: 7" Touch screen with 8 configurable keys, a resolution of 800 x 480 and 800 tags
- KTP700 Basic DP: 7" Touch screen with 8 configurable keys, a resolution of 800 x 480 and 800 tags
- KTP900 Basic: 9" Touch screen with 8 configurable keys, a resolution of 800 x 480 and 800 tags
- KTP1200 Basic: 12" Touch screen with 10 configurable keys, a resolution of 800 x 480 and 800 tags
- KTP 1200 Basic DP: 12 Touch screen with 10 configurable keys, a resolution of 800 x 400 and 800 tags

See also

Customer support (http://www.siemens.com/automation/)

2 New features

The following features are new in this release:

- You can now implement functional safety, using the hardware and firmware of the S7-1200 fail-safe CPUs and signal modules (SM) in conjunction with the safety program downloaded by the software (ES). Refer to the S7-1200 Functional Safety Manual (http://support.automation.siemens.com/WW/view/en/104547552)for further information.
- Simulation of S7-1200 CPUs with firmware version V4.0 and higher: S7-PLCSIM V13 SP1 enables you to test your PLC programs on a simulated PLC without requiring actual hardware. S7-PLCSIM is a separately installed application that operates in conjunction with STEP 7 in the TIA Portal. You can configure your PLC and any associated modules in STEP 7, program your application logic, and then download the hardware configuration and program to S7-PLCSIM. You can then use the tools of S7-PLCSIM to simulate and test your program. Refer to the online help for S7-PLCSIM for complete documentation. Note that you cannot simulate fail-safe CPUs.
- Configuration control (option handling) (Page 151): You can configure the hardware for a maximum machine configuration including modules that you might not actually use during operation. The configuration and designation of these flexible modules is new with this release of STEP 7 and the S7-1200. Modules that you so designate will not cause error conditions if they are absent.
- The Web server (Page 785) now supports access through the IP address of selected (communications processor) modules in the local rack as well as through the IP address of the S7-1200 CPU.
- Enhanced motion functionality:
  - Analog and PROFIdrive connections
  - Modulo and control loop extended parameters
- Period measurement using High-speed counters (HSC) (Page 457)
- Performance improvements to the SCL compiler
- Dynamic copy protection (Page 201) binding of program blocks with a mandatory password
- Enhanced PROFINET functionality, including support for shared devices (Page 734).

- New programming instructions:
  - EQ_Type, NE_Type, EQ_ElemType, NE_ElemType (Page 235)
  - IS_NULL, NOT_NULL (Page 236)
  - IS_ARRAY (Page 236)
  - Deserialize (Page 251), Serialize (Page 255)
  - VariantGet (Page 264), VariantPut (Page 265), CountOfElements (Page 266)
  - Variant_to_DB_Any (Page 278), DB_Any_To_Variant (Page 279)
  - GET_IM_DATA (Page 404)
  - RUNTIME (Page 293)
  - GEO2LOG (Page 447), IO2MOD (Page 451)
  - ReadLittle, WriteLittle, ReadBig, WriteBig (SCL only) (Page 262)
  - T_RESET (Page 681), T_DIAG (Page 683), and TMAIL_C (Page 688)
  - PID_Temp (Page 493)
  - New Modbus instructions (Page 931)
  - New Point-to-point (PtP) instructions (Page 864)
  - New USS instructions (Page 911)

New modules for the S7-1200

New modules expand the power of the S7-1200 CPU and provide the flexibility to meet your automation needs:

- Industrial remote control communication modules (Page 1280): You can use these CPs as communication modules with the S7-1200 V4.1 CPU.
- Fail-safe CPUs and I/O: There are four fail-safe CPUs and three fail-safe signal modules (SM) in conjunction with the S7-1200 V4.1 or later release:
  - CPU 1214FC DC/DC/DC (6ES7 214-1AF40-0XB0)
  - CPU 1214FC DC/DC/RLY (6ES7 214-1HF40-0XB0)
  - CPU 1215FC DC/DC/DC (6ES7 215-1AF40-0XB0)
  - CPU 1215FC DC/DC/RLY (6ES7 215-1HF40-0XB0)
  - SM 1226 F-DI 16 x 24 VDC (6ES7 226-6BA32-0XB0)
  - SM 1226 F-DQ 4 x 24 VDC (6ES7 226-6DA32-0XB0)
  - SM 1226 F-DQ 2 x Relay (6ES7 226-6RA32-0XB0)

  You can use the S7-1200 standard signal modules (SM), communication modules (CM), and signal boards (SB) in the same system with fail-safe SMs to complete your application control functions that do not require a functional safety rating. Standard SMs that are supported for use with fail-safe SMs have the article numbers (6ES7 --- ---32 0XB0) or later.

Exchanging your V3.0 CPU for a V4.1 CPU

If you are replacing an S7-1200 V3.0 CPU with an S7-1200 V4.1 CPU, take note of the documented differences (Page 1287) in the versions and the required user actions.

STEP 7 programming software 3

STEP 7 provides a user-friendly environment to develop, edit, and monitor the logic needed to control your application, including the tools for managing and configuring all of the devices in your project, such as controllers and HMI devices. To help you find the information you need, STEP 7 provides an extensive online help system.

STEP 7 provides standard programming languages for convenience and efficiency in developing the control program for your application.

- LAD (ladder logic) (Page 186) is a graphical programming language. The representation is based on circuit diagrams.
- FBD (Function Block Diagram) (Page 187) is a programming language that is based on the graphical logic symbols used in Boolean algebra.
- SCL (structured control language) (Page 188) is a text-based, high-level programming language.

When you create a code block, you select the programming language to be used by that block. Your user program can utilize code blocks created in any or all of the programming languages.

Note

STEP 7 is the programming and configuration software component of the TIA Portal. The TIA Portal, in addition to STEP 7, also includes WinCC for designing and executing runtime process visualization, and includes online help for WinCC as well as STEP 7.

3.1 System requirements

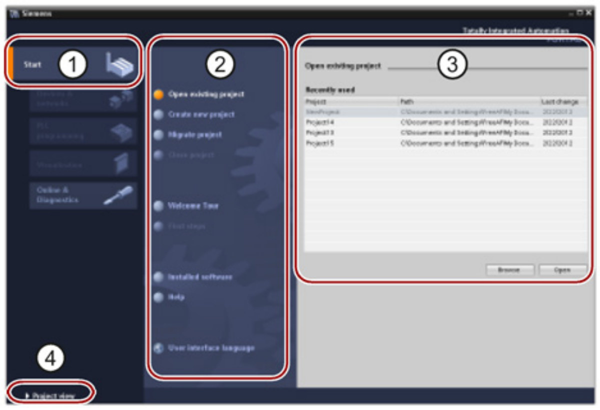
You must install STEP 7 with Administrator privileges.

Table 3- 1 System requirements

Hardware/software	**Requirements**
Processor type	Intel® Core™ i5-3320M 3.3 GHz or better
RAM	8 GB
Available hard disk space	2 GB on system drive C:\
Operating systems	You can use STEP 7 with the following operating systems (64-bit, Windows 7 also 32-bit) • Microsoft Windows 7 Home Premium SP1 or higher (STEP 7 Basic only, not supported for STEP 7 Professional) • Microsoft Windows 7 or higher (Professional SP1, Enterprise SP1, Ultimate SP1) • Microsoft Windows 8.1 (STEP 7 Basic only, not supported for STEP 7 Professional) • Microsoft Windows 8.1 (Professional, Enterprise) • Microsoft Server 2008 R2 StdE SP1 (STEP 7 Professional only) • Microsoft Server 2012 R2 StdE
Graphics card	32 MB RAM 24-bit color depth
Screen resolution	1920 x 1080 (recommended)
Network	20 Mbit/s Ethernet or faster
Optical drive	DVD-ROM

3.2 Different views to make the work easier

STEP 7 provides a user-friendly environment to develop controller logic, configure HMI visualization, and setup network communication. To help increase your productivity, STEP 7 provides two different views of the project: a task-oriented set of portals that are organized on the functionality of the tools (Portal view), or a project-oriented view of the elements within the project (Project view). Choose which view helps you work most efficiently. With a single click, you can toggle between the Portal view and the Project view.



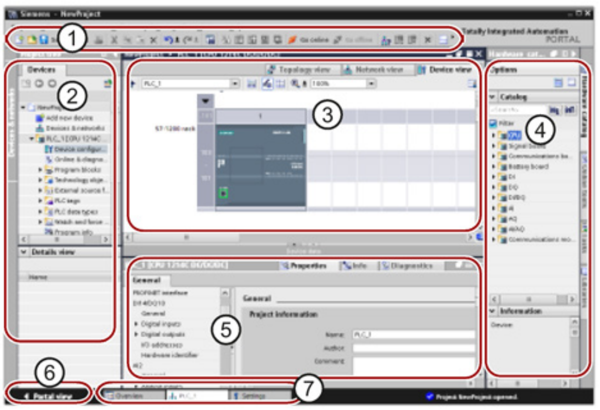
Portal view

① Portals for the different tasks

② Tasks for the selected portal

③ Selection panel for the selected action

④ Changes to the Project view



Project view

① Menus and toolbar

② Project navigator

③ Work area

④ Task cards

⑤ Inspector window

⑥ Changes to the Portal view

⑦ Editor bar

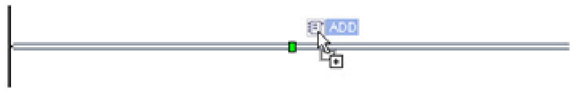
With all of these components in one place, you have easy access to every aspect of your project. For example, the inspector window shows the properties and information for the object that you have selected in the work area. As you select different objects, the inspector window displays the properties that you can configure. The inspector window includes tabs that allow you to see diagnostic information and other messages.

By showing all of the editors that are open, the editor bar helps you work more quickly and efficiently. To toggle between the open editors, simply click the different editor. You can also arrange two editors to appear together, arranged either vertically or horizontally. This feature allows you to drag and drop between editors.

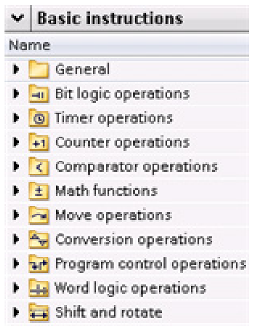
3.3 Easy-to-use tools

3.3.1 Inserting instructions into your user program

STEP 7 provides task cards that contain the instructions for your program. The instructions are grouped according to function.

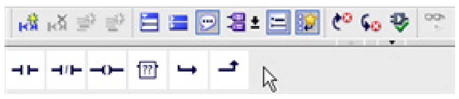


To create your program, you drag instructions from the task card onto a network.

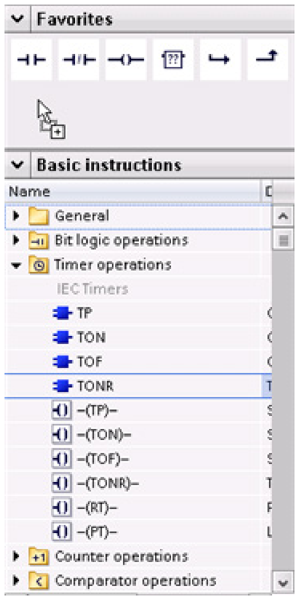


3.3.2 Accessing instructions from the "Favorites" toolbar

STEP 7 provides a "Favorites" toolbar to give you quick access to the instructions that you frequently use. Simply click the icon for the instruction to insert it into your network!



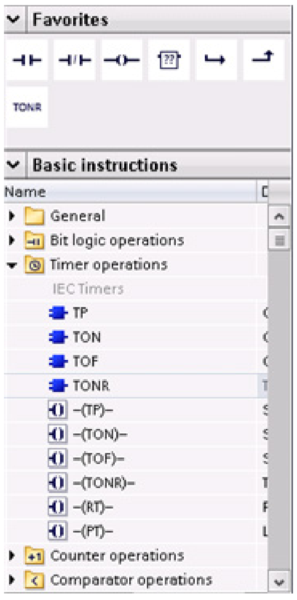
(For the "Favorites" in the instruction tree, double-click the icon.)



You can easily customize the "Favorites" by adding new instructions.

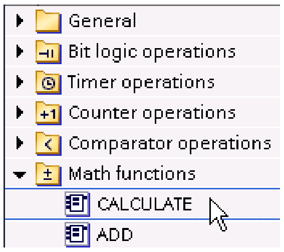
Simply drag and drop an instruction to the "Favorites".

The instruction is now just a click away!

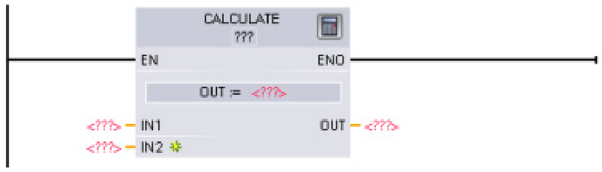


3.3.3 Creating a complex equation with a simple instruction

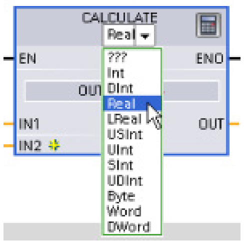
The Calculate instruction (Page 237) lets you create a math function that operates on multiple input parameters to produce the result, according to the equation that you define.



In the Basic instruction tree, expand the Math functions folder. Double-click the Calculate instruction to insert the instruction into your user program.



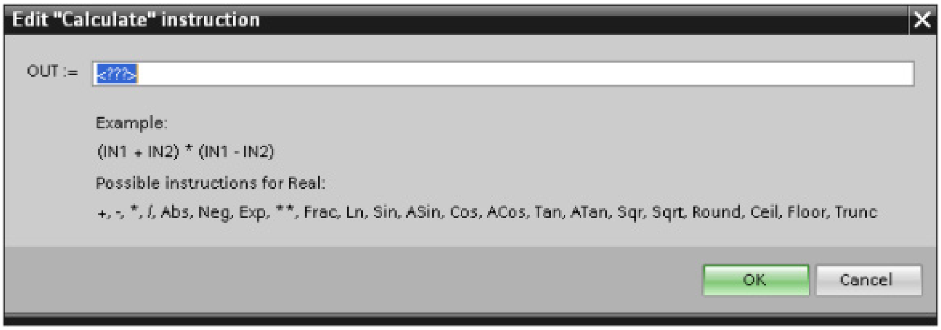
The unconfigured Calculate instruction provides two input parameters and an output parameter.



Click the "???" and select the data types for the input and output parameters. (The input and output parameters must all be the same data type.)

For this example, select the "Real" data type.

Click the "Edit equation" icon to enter the equation.



Example:
(IN1 + IN2) * (IN1 - IN2)
Possible instructions for Real:
+, -, *, /, Abs, Neg, Exp, **, Frac, Ln, Sin, ASin, Cos, ACos, Tan, ATan, Sqr, Sqrt, Round, Ceil, Floor, Trunc

For this example, enter the following equation for scaling a raw analog value. (The "In" and "Out" designations correspond to the parameters of the Calculate instruction.)

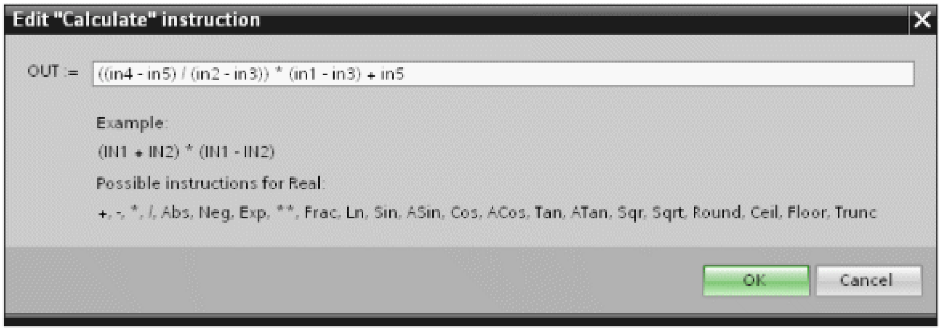
Out_{value} = ((Out_{high} - Out_{low}) / (In_{high} - In_{low})) * (In_{value} - In_{low}) + Out_{low}

Out = ((in4 - in5) / (in2 - in3)) * (in1 - in3) + in5

Where:			
	Out_{value}	(Out)	Scaled output value
	In_{value}	(in1)	Analog input value
	In_{high}	(in2)	Upper limit for the scaled input value
	In_{low}	(in3)	Lower limit for the scaled input value
	Out_{high}	(in4)	Upper limit for the scaled output value
	Out_{low}	(in5)	Lower limit for the scaled output value

In the "Edit Calculate" box, enter the equation with the parameter names:

OUT = ((in4 - in5) / (in2 - in3)) * (in1 - in3) + in5

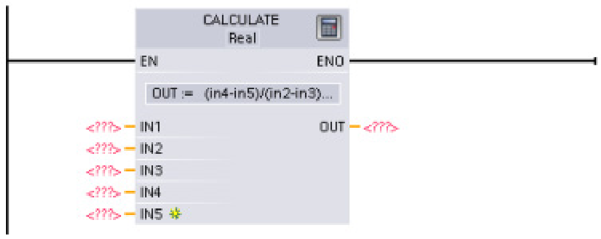


Edit "Calculate" instruction

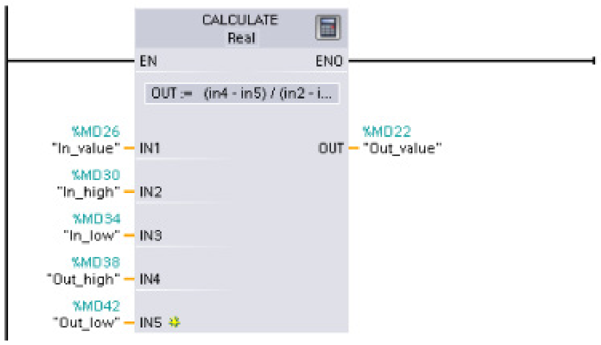
OUT := ((in4 - in5) / (in2 - in3)) * (in1 - in3) + in5

Example:
(IN1 + IN2) * (IN1 - IN2)
Possible instructions for Real:
+, -, *, /, Abs, Neg, Exp, **, Frac, Ln, Sin, ASin, Cos, ACos, Tan, ATan, Sqr, Sqrt, Round, Ceil, Floor, Trunc

When you click "OK", the Calculate instruction creates the inputs required for the instruction.



Enter the tag names for the values that correspond to the parameters.



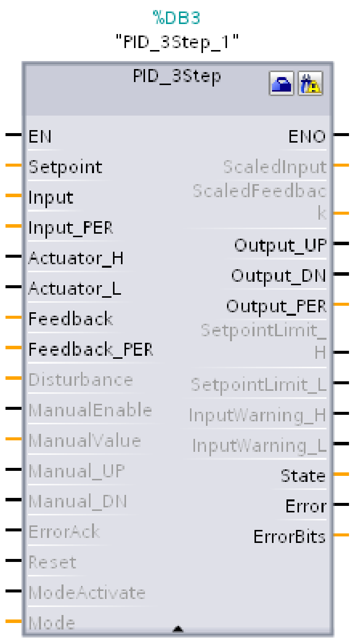
3.3.4 Adding inputs or outputs to a LAD or FBD instruction

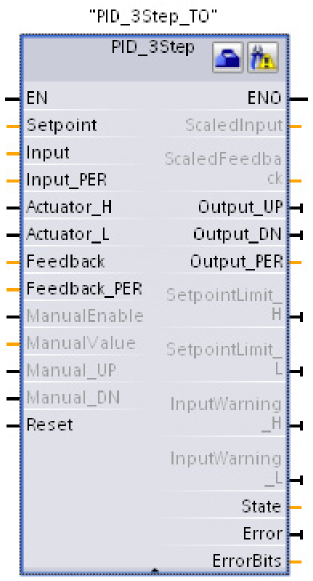
Some of the instructions allow you to create additional inputs or outputs.

- To add an input or output, click the "Create" icon or right-click on an input stub for one of the existing IN or OUT parameters and select the "Insert input" command.
- To remove an input or output, right-click on the stub for one of the existing IN or OUT parameters (when there are more than the original two inputs) and select the "Delete" command.

3.3.5 Expandable instructions

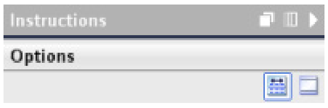
Some of the more complex instructions are expandable, displaying only the key inputs and outputs. To display all the inputs and outputs, click the arrow at the bottom of the instruction.



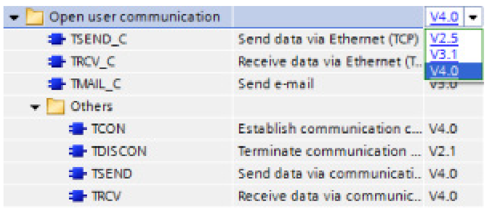


3.3.6 Selecting a version for an instruction

The development and release cycles for certain sets of instructions (such as Modbus, PID and motion) have created multiple released versions for these instructions. To help ensure compatibility and migration with older projects, STEP 7 allows you to choose which version of instruction to insert into your user program.



Click the icon on the instruction tree task card to enable the headers and columns of the instruction tree.



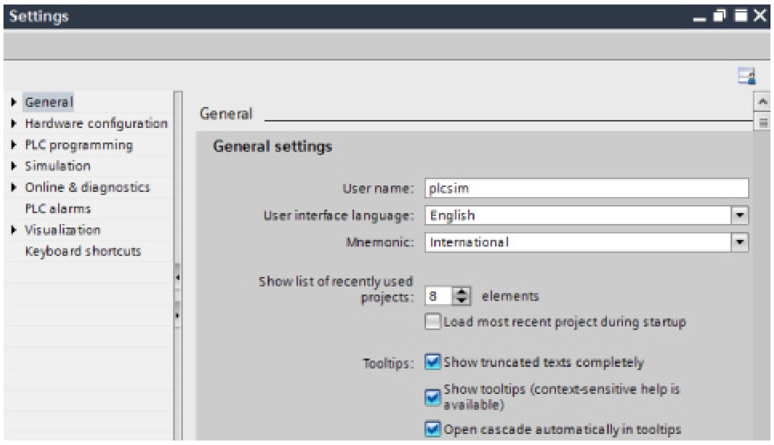
Open user communication		V4.0
TSEND_C	Send data via Ethernet (TCP)	V2.5
TRCV_C	Receive data via Ethernet (T...	V3.1
TMAIL_C	Send e-mail	V4.0
Others		
TCON	Establish communication c...	V4.0
TDISCON	Terminate communication ...	V2.1
TSEND	Send data via communicati...	V4.0
TRCV	Receive data via communic...	V4.0

To change the version of the instruction, select the appropriate version from the drop-down list.

3.3.7 Modifying the appearance and configuration of STEP 7

You can select a variety of settings, such as the appearance of the interface, language, or the folder for saving your work.

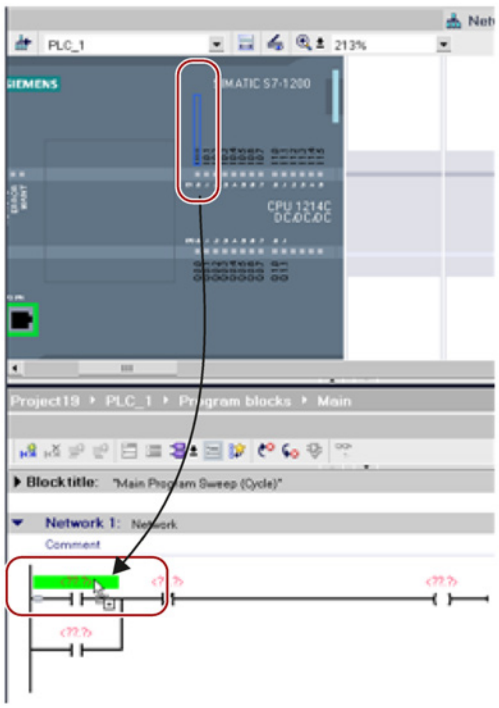
Select the "Settings" command from the "Options" menu to change these settings.



General settings

User name: plcsim
User interface language: English
Mnemonic: International
Show list of recently used projects: 8 elements
☐ Load most recent project during startup
Tooltips: ✓ Show truncated texts completely
✓ Show tooltips (context-sensitive help is available)
✓ Open cascade automatically in tooltips

3.3.8 Dragging and dropping between editors

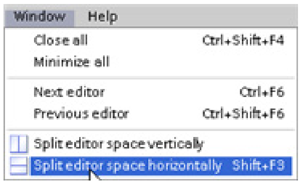


To help you perform tasks quickly and easily, STEP 7 allows you to drag and drop elements from one editor to another. For example, you can drag an input from the CPU to the address of an instruction in your user program.

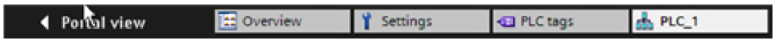
You must zoom in at least 200% to select the inputs or outputs of the CPU.

Notice that the tag names are displayed not only in the PLC tag table, but also are displayed on the CPU.

To display two editors at one time, use the "Split editor" menu commands or buttons in the toolbar.


Window
Help
Close all Ctrl+Shift+F4
Minimize all
Next editor Ctrl+F6
Previous editor Ctrl+Shift+F6
Split editor space vertically
Split editor space horizontally Shift+F8

To toggle between the editors that have been opened, click the icons in the editor bar.



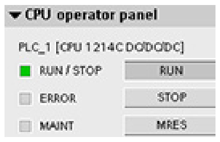
3.3.9 Changing the operating mode of the CPU

The CPU does not have a physical switch for changing the operating mode (STOP or RUN).

Use the "Start CPU" and "Stop CPU" toolbar buttons to change the operating mode of the CPU.

When you configure the CPU in the device configuration, you configure the start-up behavior in the properties of the CPU (Page 162).

The "Online and diagnostics" portal also provides an operator panel for changing the operating mode of the online CPU. To use the CPU operator panel, you must be connected online to the CPU. The "Online tools" task card displays an operator panel that shows the operating mode of the online CPU. The operator panel also allows you to change the operating mode of the online CPU.

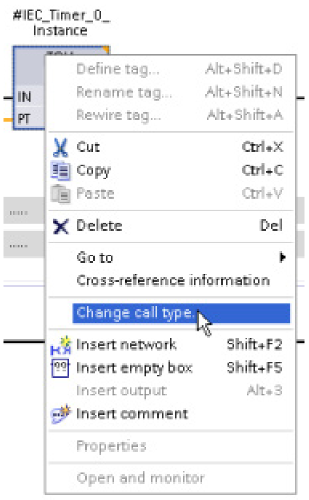


Use the button on the operator panel to change the operating mode (STOP or RUN). The operator panel also provides an MRES button for resetting the memory.

The color of the RUN/STOP indicator shows the current operating mode of the CPU. Yellow indicates STOP mode, and green indicates RUN mode.

From the device configuration in STEP 7 you can also configure the default operating mode on power up of the CPU (Page 83).

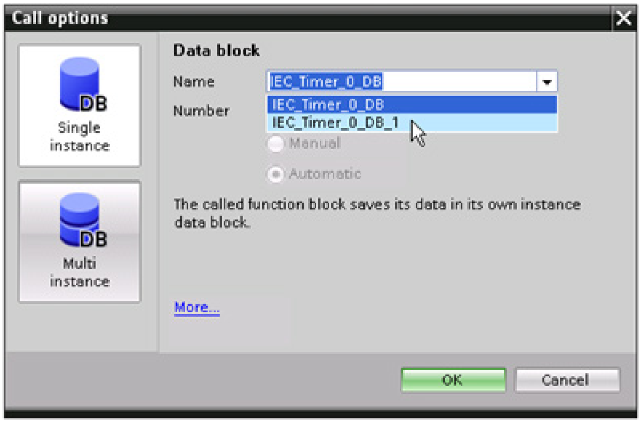
3.3.10 Changing the call type for a DB



STEP 7 allows you to easily create or change the association of a DB for an instruction or an FB that is in an FB.

- You can switch the association between different DBs.
- You can switch the association between a single-instance DB and a multi-instance DB.
- You can create an instance DB (if an instance DB is missing or not available).

You can access the "Change call type" command either by right-clicking the instruction or FB in the program editor or by selecting the "Block call" command from the "Options" menu.

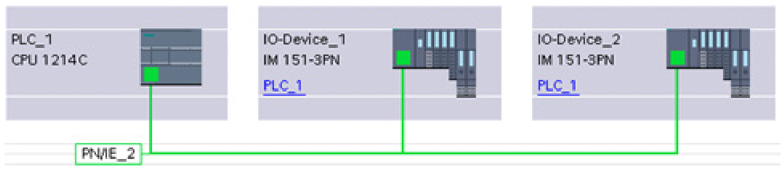


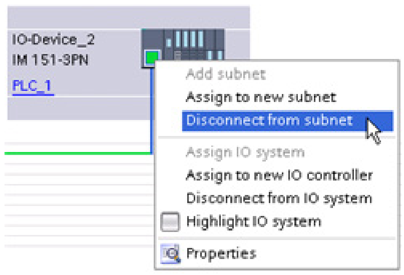
The called function block saves its data in its own instance data block.

The "Call options" dialog allows you to select a single-instance or multi-instance DB. You can also select specific DBs from a drop-down list of available DBs.

3.3.11 Temporarily disconnecting devices from a network

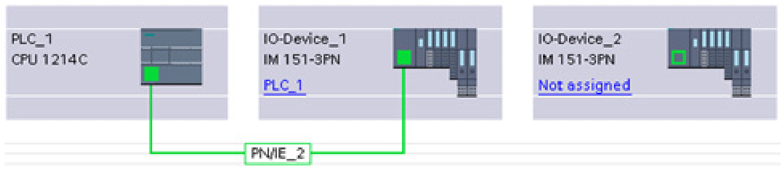
You can disconnect individual network devices from the subnet. Because the configuration of the device is not removed from the project, you can easily restore the connection to the device.





Right-click the interface port of the network device and select the "Disconnect from subnet" command from the context menu.

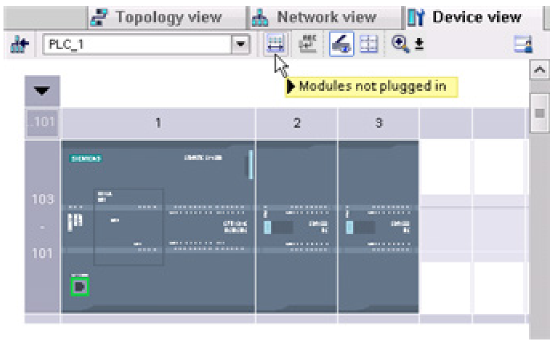
STEP 7 reconfigures the network connections, but does not remove the disconnected device from the project. While the network connection is deleted, the interface addresses are not changed.

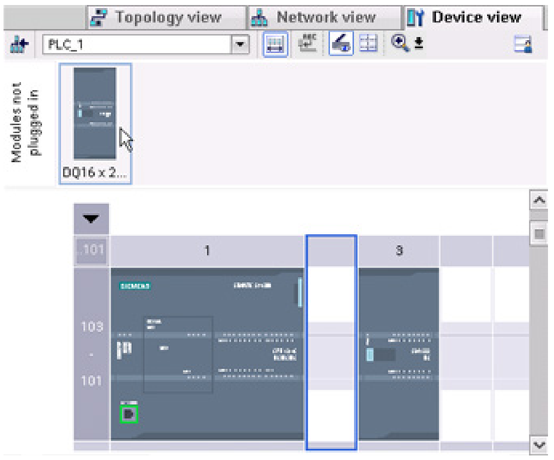


When you download the new network connections, the CPU must be set to STOP mode.

To reconnect the device, simply create a new network connection to the port of the device.

3.3.12 Virtual unplugging of devices from the configuration





STEP 7 provides a storage area for "un-plugged" modules. You can drag a module from the rack to save the configuration of that module. These unplugged modules are saved with your project, allowing you to reinsert the module in the future without having to reconfigure the parameters.

One use of this feature is for temporary maintenance. Consider a scenario where you might be waiting for a replacement module and plan to temporarily use a different module as a short-term replacement. You could drag the configured module from the rack to the "Unplugged modules" and then insert the temporary module.

3.4 Backward compatibility

STEP 7 V13 SP1 supports configuration and programming of the S7-1200 V4.1 CPU.

You can, however, download configurations and programs for S7-1200 V4.0 from STEP 7 V13 to an S7-1200 V4.1 CPU. Your configuration and program will be limited to the set of features and instructions that STEP 7 V13 and the S7-1200 V4.0 supported.

This backwards compatibility makes it possible for you to run programs on the new S7-1200 V4.1 CPU versions that you have previously designed and programmed for older versions.

Installation 4

4.1 Guidelines for installing S7-1200 devices

The S7-1200 equipment is designed to be easy to install. You can install an S7-1200 either on a panel or on a standard rail, and you can orient the S7-1200 either horizontally or vertically. The small size of the S7-1200 allows you to make efficient use of space.

The S7-1200 fail-safe CPUs do not support PROFIBUS or PROFINET distributed fail-safe I/O.

Electrical equipment standards classify the SIMATIC S7-1200 system as Open Equipment. You must install the S7-1200 in a housing, cabinet, or electric control room. You should limit entry to the housing, cabinet, or electric control room to authorized personnel.

The installation should provide a dry environment for the S7-1200. SELV/PELV circuits are considered to provide protection against electric shock in dry locations.

The installation should provide mechanical and environmental 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 S7-1200 can result in electrical faults or unexpected operation of machinery.

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

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

Separate the S7-1200 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 S7-1200.

When configuring the layout of the S7-1200 inside your panel, consider the heat-generating devices and locate the electronic-type devices in the cooler areas of your cabinet. Reducing the exposure to a high-temperature environment will extend the operating life of any electronic device.

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

Provide adequate clearance for cooling and wiring

S7-1200 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.

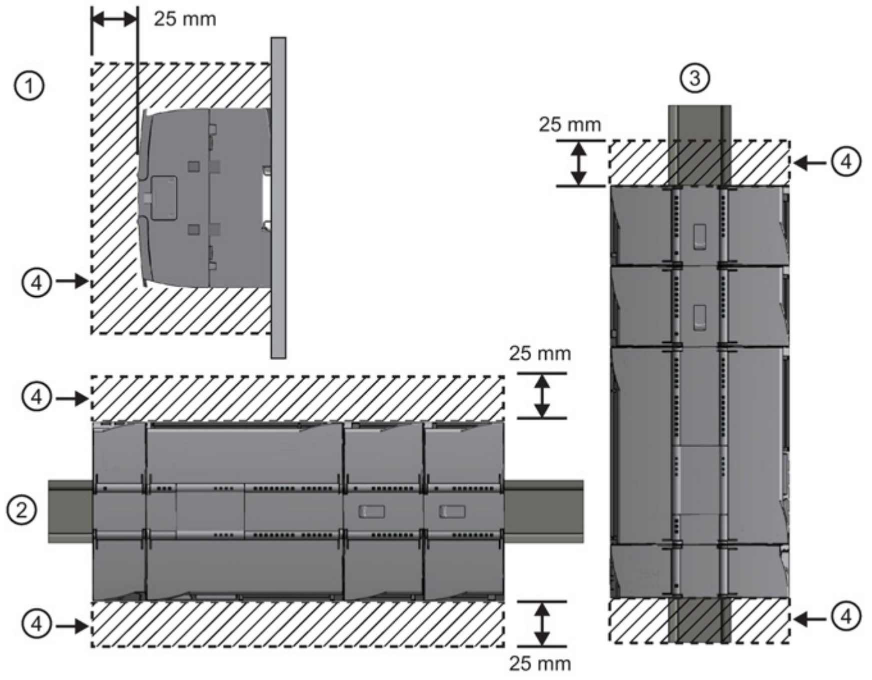
⚠ CAUTION

For vertical mounting, the maximum allowable ambient temperature is reduced by 10 degrees C.

Orient a vertically mounted S7-1200 system as shown in the following figure.

Ensure that the S7-1200 system is mounted correctly.

When planning your layout for the S7-1200 system, allow enough clearance for the wiring and communications cable connections.



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

4.2 Power budget

Your CPU has an internal power supply that provides power for the CPU, the signal modules, signal board and communication modules and for other 24 VDC user power requirements.

Refer to the technical specifications (Page 1099) for information about the 5 VDC logic budget supplied by your CPU and the 5 VDC power requirements of the signal modules, signal boards, and communication modules. Refer to the "Calculating a power budget" (Page 1273) to determine how much power (or current) the CPU can provide for your configuration.

The CPU provides a 24 VDC sensor supply that can supply 24 VDC for input points, for relay coil power on the signal modules, or for other requirements. If your 24 VDC power requirements exceed the budget of the sensor supply, then you must add an external 24 VDC power supply to your system. Refer to the technical specifications (Page 1099) for the 24 VDC sensor supply power budget for your particular CPU.

Note

The CM 1243-5 (PROFIBUS master module) requires power from the 24 VDC sensor supply of the CPU.

If you require an external 24 VDC 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 an external 24 VDC power supply in parallel with the 24 VDC sensor supply 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, severe personal injury and/or property damage.

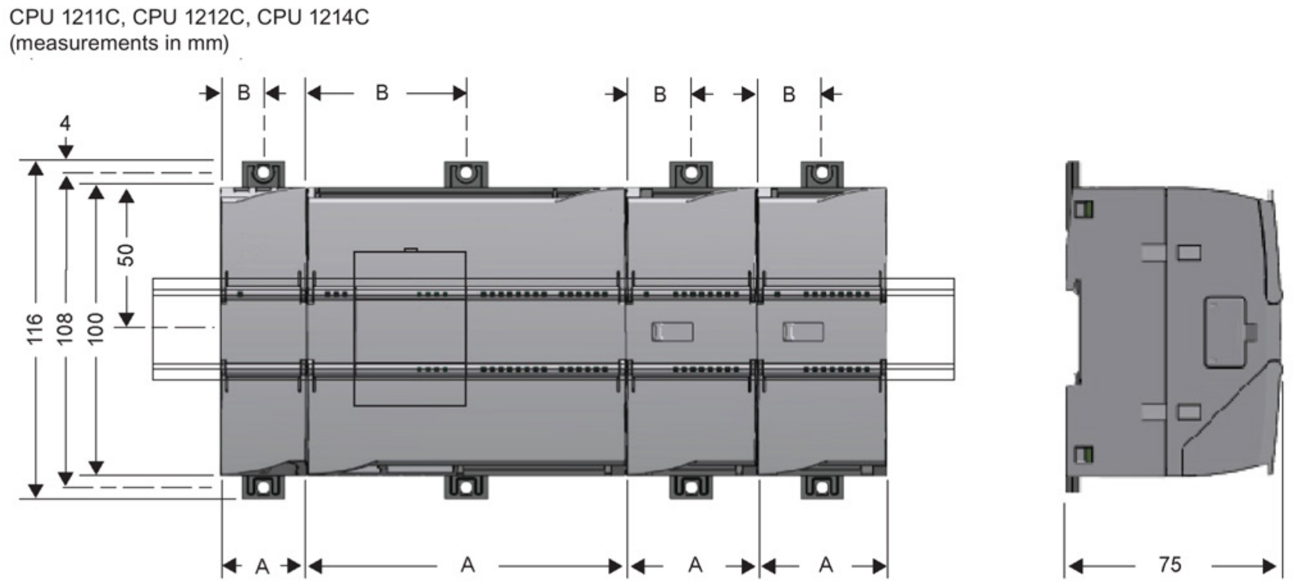
The DC sensor supply and any external power supply should provide power to different points.

Some of the 24 VDC power input ports in the S7-1200 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 VDC power supply of the CPU, the power input for the relay coil of an SM, or the power supply for a non-isolated analog input. All non-isolated M terminals must connect to the same external reference potential.

⚠WARNING
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 S7-1200 system are connected to the same reference potential.

4.3 Installation and removal procedures

4.3.1 Mounting dimensions for the S7-1200 devices



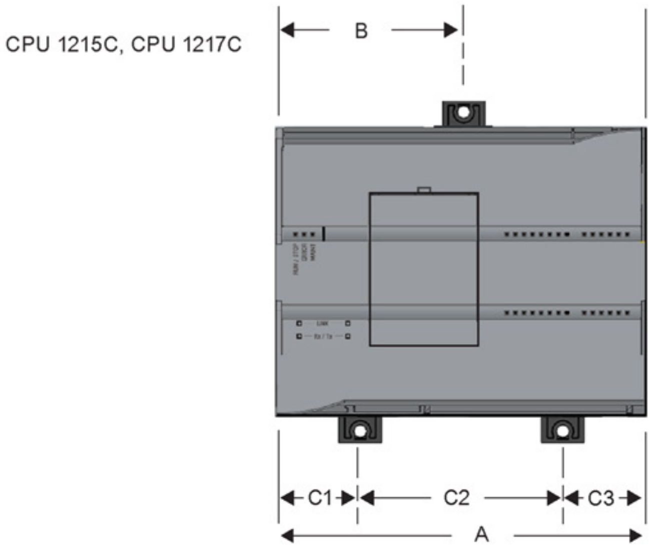


Table 4- 1 Mounting dimensions (mm)

S7-1200 Devices		Width A (mm)	Width B (mm)	Width C (mm)
CPU	CPU 1211C and CPU 1212C	90	45	--
	CPU 1214C	110	55	--
	CPU 1215C	130	65 (top)	Bottom: C1: 32.5 C2: 65 C3: 32.5
	CPU 1217C	150	75	Bottom: C1: 37.5 C2: 75 C3: 37.5
Signal modules	Digital 8 and 16 point Analog 2, 4, and 8 point Thermocouple 4 and 8 point RTD 4 point SM 1278 IO Link-Master	45	22.5	--
	Digital DQ 8 x Relay (Changeover)	70	35	--
	Analog 16 point RTD 8 point	70	35	--
Communication interfaces	CM 1241 RS232, and CM 1241 RS422/485 CM 1243-5 PROFIBUS master and CM 1242-5 PROFIBUS slave CM 1242-2 AS-i Master CP 1242-7 GPRS V2 CP 1243-7 LTE-EU CP 1243-1 DNP3 CP 1243-1 IEC CP 1243-1 CP1243-1 PCC CP 1243-8 ST7 RF120C	30	15	--
	TS (TeleService) Adapter IE Advanced [1] TS (Teleservice) Adapter IE Basic [1] TS Adapter TS Module	 30 30	 15 15	 -- --

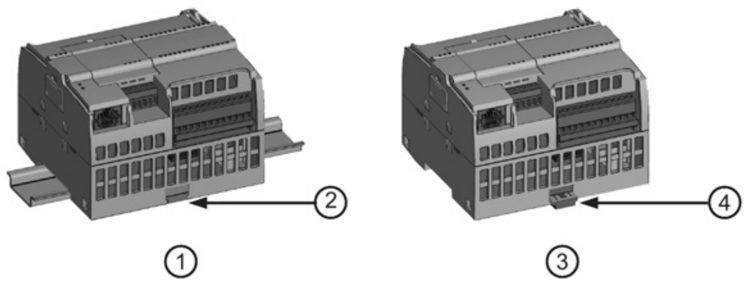
[1] Before installing the TS (TeleService) Adapter IE Advanced or IE Basic, you must first connect the TS Adapter and a TS module. The total width ("width A") is 60 mm.

Each CPU, SM, CM, and CP supports mounting on either a DIN rail or on a panel. 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. The interior dimension of the hole for the DIN clips on the device is 4.3 mm.

A 25 mm thermal zone must be provided above and below the unit for free air circulation.

Installing and removing the S7-1200 devices

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



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

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

Installation or removal of S7-1200 or related equipment with the power applied could cause electric shock or unexpected operation of equipment.

Failure to disable all power to the S7-1200 and related equipment during installation or removal procedures could result in death, severe personal injury and/or property damage due to electric shock or unexpected equipment operation.

Always follow appropriate safety precautions and ensure that power to the S7-1200 is disabled before attempting to install or remove S7-1200 CPUs or related equipment.

Always ensure that whenever you replace or install an S7-1200 device you use the correct module or equivalent device.

⚠ WARNING

Incorrect installation of an S7-1200 module may cause the program in the S7-1200 to function unpredictably.

Failure to replace an S7-1200 device with the same model, orientation, or order could result in death, severe personal injury and/or property damage due to unexpected equipment operation.

Replace an S7-1200 device with the same model, and be sure to orient and position it correctly.

⚠ WARNING

Do not disconnect equipment when a flammable or combustible atmosphere is present.

Disconnection of equipment when a flammable or combustible atmosphere is present may cause a fire or explosion which could result in death, serious injury and/or property damage.

Always follow appropriate safety precautions when a flammable or combustible atmosphere is present.

Note

Electrostatic discharge can damage the device or the receptacle on the CPU.

Make contact with a grounded conductive pad and/or wear a grounded wrist strap whenever you handle the device.

4.3.2 Installing and removing the CPU

You can install the CPU on a panel or on a DIN rail.

Note

Attach any communication modules to the CPU and install the assembly as a unit. Install signal modules separately after the CPU has been installed.

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

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

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

1. Locate, drill, and tap the mounting holes (M4), using the dimensions shown in table, Mounting dimensions (mm) (Page 54).
2. Ensure that the CPU and all S7-1200 equipment are disconnected from electrical power.
3. Extend the mounting clips from the module. Make sure the DIN rail clips on the top and bottom of the CPU are in the extended position.
4. Secure the module to the panel, using a Pan Head M4 screw with spring and flat washer. Do not use a flat head screw.

Note

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

Note

Using DIN rail stops could be helpful if your CPU is in an environment with high vibration potential or if the CPU has been installed vertically. Use an end bracket (8WA1808 or 8WA1805) on the DIN rail to ensure that the modules remain connected. If your system is in a high-vibration environment, then panel-mounting the CPU will provide a greater level of vibration protection.

Table 4- 2 Installing the CPU on a DIN rail

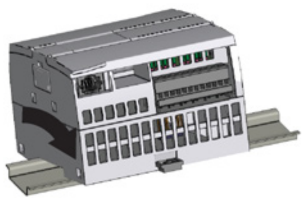
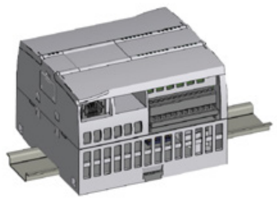
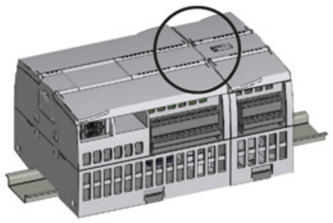
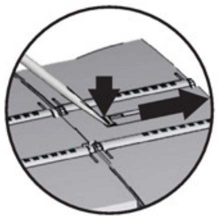
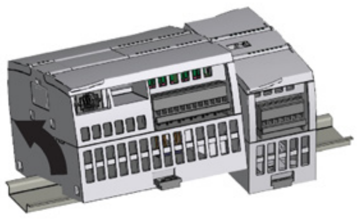
Task	Procedure
	1. Install the DIN rail. Secure the rail to the mounting panel every 75 mm. 2. Ensure that the CPU and all S7-1200 equipment are disconnected from electrical power. 3. Hook the CPU over the top of the DIN rail. 4. Pull out the DIN rail clip on the bottom of the CPU to allow the CPU to fit over the rail. 5. Rotate the CPU down into position on the rail. 6. Push in the clips to latch the CPU to the rail.
	

Table 4- 3 Removing the CPU from a DIN rail

Task		Procedure
		1. Ensure that the CPU and all S7-1200 equipment are disconnected from electrical power. 2. Disconnect the I/O connectors, wiring, and cables from the CPU (Page 65). 3. Remove the CPU and any attached communication modules as a unit. All signal modules should remain installed. 4. If an SM is connected to the CPU, retract the bus connector: – Place a screwdriver beside the tab on the top of the signal module. – Press down to disengage the connector from the CPU. – Slide the tab fully to the right. 5. Remove the CPU: – Pull out the DIN rail clip to release the CPU from the rail. – Rotate the CPU up and off the rail, and remove the CPU from the system.
		

4.3.3 Installing and removing an SB, CB, or BB

Table 4- 4 Installing an SB, CB, or BB 1297

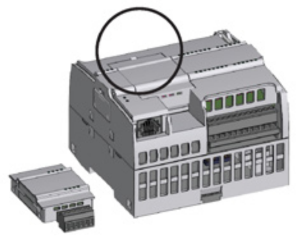
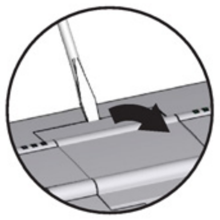
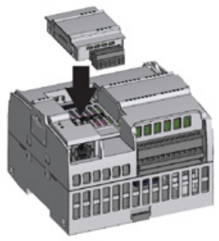
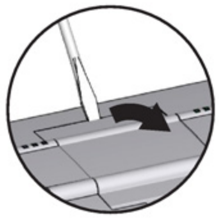
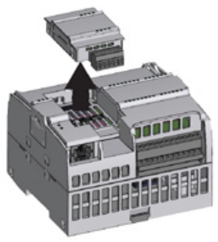
Task		Procedure
		1. Ensure that the CPU and all S7-1200 equipment are disconnected from electrical power. 2. Remove the top and bottom terminal block covers from the CPU. 3. Place a screwdriver into the slot on top of the CPU at the rear of the cover. 4. Gently pry the cover up and remove it from the CPU. 5. Place the module straight down into its mounting position in the top of the CPU. 6. Firmly press the module into position until it snaps into place. 7. Replace the terminal block covers.
		

Table 4- 5 Removing an SB, CB or BB 1297

Task		Procedure
		1. Ensure that the CPU and all S7-1200 equipment are disconnected from electrical power. 2. Remove the top and bottom terminal block covers from the CPU. 3. Place a screwdriver into the slot on top of the module. 4. Gently pry the module up to disengage it from the CPU. 5. Remove the module straight up from its mounting position in the top of the CPU. 6. Replace the cover onto the CPU. 7. Replace the terminal block covers.
		

Installing or replacing the battery in the BB 1297 battery board

The BB 1297 requires battery type CR1025. The battery is not included with the BB 1297 and must be purchased. To install or replace the battery, follow these steps:

1. In the BB 1297, install a new battery with the positive side of the battery on top, and the negative side next to the printed wiring board.
2. The BB 1297 is ready to be installed in the CPU. Follow the installation directions above to install the BB 1297.

To replace the battery in the BB 1297:

1. Remove the BB 1297 from the CPU following the removal directions above.
2. Carefully remove the old battery using a small screwdriver. Push the battery out from under the clip.
3. Install a new CR1025 replacement battery with the positive side of the battery on top and the negative side next to the printed wiring board.
4. Re-install the BB 1297 battery board following the installation directions above.

⚠ WARNING

Installing an unspecified battery in the BB 1297, or otherwise connecting an unspecified battery to the circuit can result in fire or component damage and unpredictable operation of machinery.

Fire or unpredictable operation of machinery can result in death, severe personal injury, or property damage.

Use only the specified CR1025 battery for backup of the Real-time clock.

4.3.4 Installing and removing an SM

Table 4- 6 Installing an SM

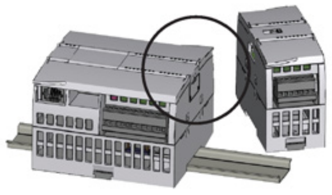
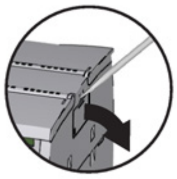
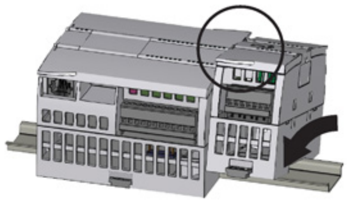
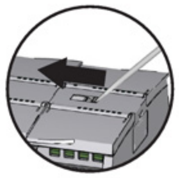
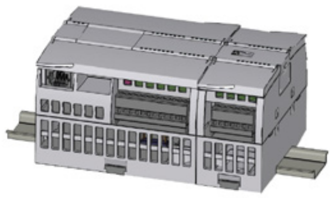
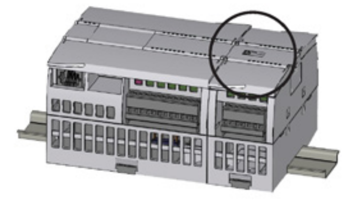
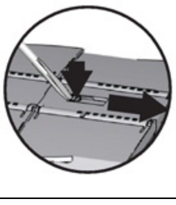
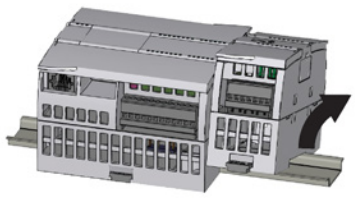
Task		Procedure
		Install your SM after installing the CPU. 1. Ensure that the CPU and all S7-1200 equipment are disconnected from electrical power. 2. Remove the cover for the connector from the right side of the CPU: – Insert a screwdriver into the slot above the cover. – Gently pry the cover out at its top and remove the cover. 3. Retain the cover for reuse.
		Connect the SM to the CPU: 1. Position the SM beside the CPU. 2. Hook the SM over the top of the DIN rail. 3. Pull out the bottom DIN rail clip to allow the SM to fit over the rail. 4. Rotate the SM down into position beside the CPU and push the bottom clip in to latch the SM onto the rail.
		Extending the bus connector makes both mechanical and electrical connections for the SM. 1. Place a screwdriver beside the tab on the top of the SM. 2. Slide the tab fully to the left to extend the bus connector into the CPU. Follow the same procedure to install a signal module to a signal module.

Table 4- 7 Removing an SM

Task	Procedure
 	You can remove any SM without removing the CPU or other SMs in place. 1. Ensure that the CPU and all S7-1200 equipment are disconnected from electrical power. 2. Remove the I/O connectors and wiring from the SM (Page 65). 3. Retract the bus connector. – Place a screwdriver beside the tab on the top of the SM. – Press down to disengage the connector from the CPU. – Slide the tab fully to the right. If there is another SM to the right, repeat this procedure for that SM.
	Remove the SM: 1. Pull out the bottom DIN rail clip to release the SM from the rail. 2. Rotate the SM up and off the rail. Remove the SM from the system. 3. If required, cover the bus connector on the CPU to avoid contamination. Follow the same procedure to remove a signal module from a signal module.

4.3.5 Installing and removing a CM or CP

Attach any communication modules to the CPU and install the assembly as a unit, as shown in Installing and removing the CPU (Page 58).

Table 4- 8 Installing a CM or CP

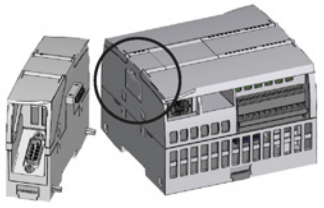
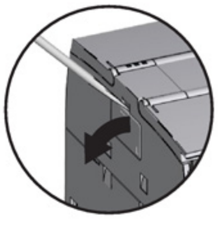
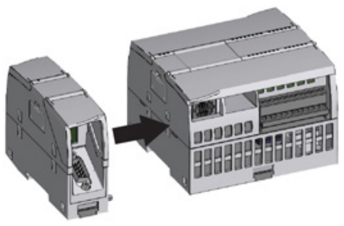
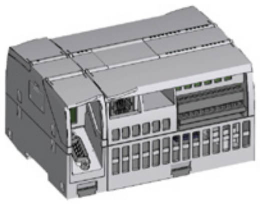
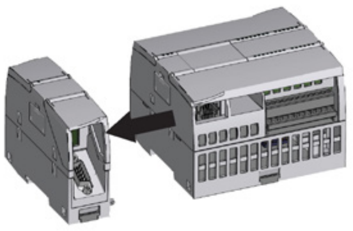
Task		Procedure
		1. Ensure that the CPU and all S7-1200 equipment are disconnected from electrical power. 2. Attach the CM to the CPU before installing the assembly as a unit to the DIN rail or panel. 3. Remove the bus cover from the left side of the CPU: – Insert a screwdriver into the slot above the bus cover. – Gently pry out the cover at its top. 4. Remove the bus cover. Retain the cover for reuse. 5. Connect the CM or CP to the CPU: – Align the bus connector and the posts of the CM with the holes of the CPU – Firmly press the units together until the posts snap in-to place. 6. Install the CPU and CP on a DIN rail or panel.
		

Table 4- 9 Removing a CM or CP

Task	Procedure
	Remove the CPU and CM as a unit from the DIN rail or panel. 1. Ensure that the CPU and all S7-1200 equipment are disconnected from electrical power. 2. Remove the I/O connectors and all wiring and cables from the CPU and CMs. 3. For DIN rail mounting, move the lower DIN rail clips on the CPU and CMs to the extended position. 4. Remove the CPU and CMs from the DIN rail or panel. 5. Grasp the CPU and CMs firmly and pull apart.
	

NOTICE
Separate modules without using a tool.
Do not use a tool to separate the modules because this can damage the units.

4.3.6 Removing and reinstalling the S7-1200 terminal block connector

The CPU, SB and SM modules provide removable connectors to make connecting the wiring easy.

Table 4- 10 Removing the connector

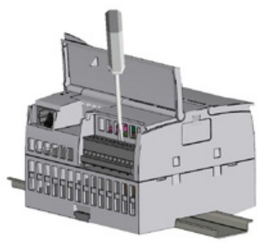
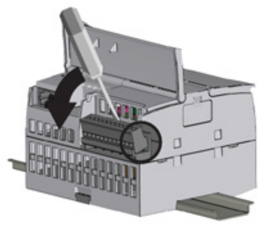
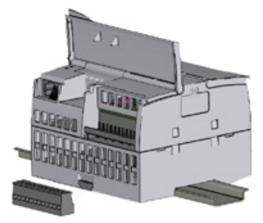
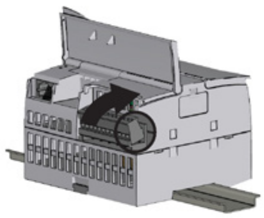
Task	Procedure
	Prepare the system for terminal block connector removal by removing the power from the CPU and opening the cover above the connector. 1. Ensure that the CPU and all S7-1200 equipment are disconnected from electrical power. 2. Inspect the top of the connector and locate the slot for the tip of the screwdriver. 3. Insert a screwdriver into the slot. 4. Gently pry the top of the connector away from the CPU. The connector will release with a snap. 5. Grasp the connector and remove it from the CPU.
	

Table 4- 11 Installing the connector

Task	Procedure
	Prepare the components for terminal block installation by removing power from the CPU and opening the cover for connector. 1. Ensure that the CPU and all S7-1200 equipment are disconnected from electrical power. 2. Align the connector with the pins on the unit. 3. Align the wiring edge of the connector inside the rim of the connector base. 4. Press firmly down and rotate the connector until it snaps into place. Check carefully to ensure that the connector is properly aligned and fully engaged.
	

4.3.7 Installing and removing the expansion cable

The S7-1200 expansion cable provides additional flexibility in configuring the layout of your S7-1200 system. Only one expansion cable is allowed per CPU system. You install the expansion cable either between the CPU and the first SM, or between any two SMs.

Table 4- 12 Installing and removing the male connector of the expansion cable

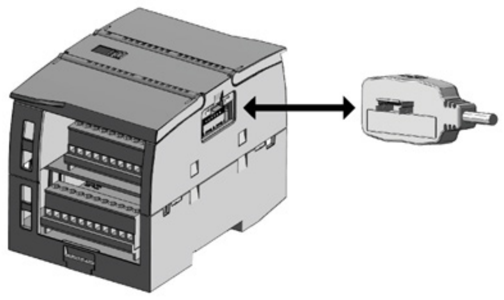
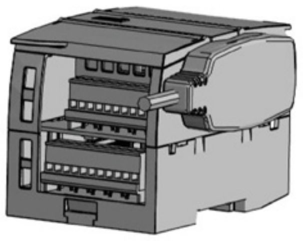
Task	Procedure
 	To install the male connector: 1. Ensure that the CPU and all S7-1200 equipment are disconnected from electrical power. 2. Push the connector into the bus connector on the right side of the signal module or CPU. To remove the male connector: 1. Ensure that the CPU and all S7-1200 equipment are disconnected from electrical power. 2. Pull out the male connector to release it from the signal module or CPU.

Table 4- 13 Installing the female connector of the expansion cable

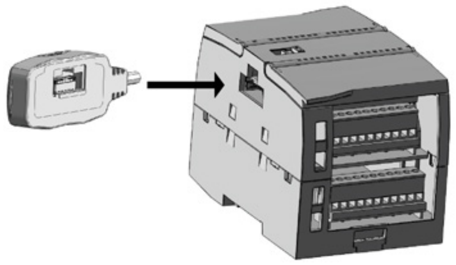
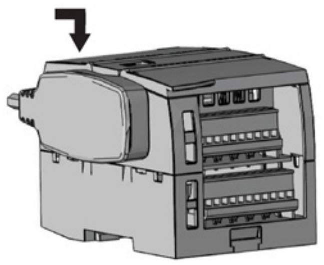
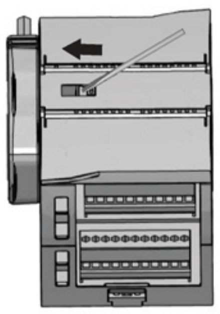
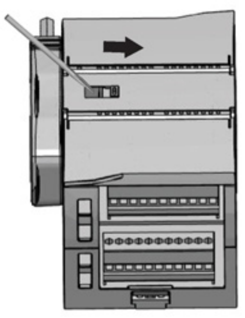
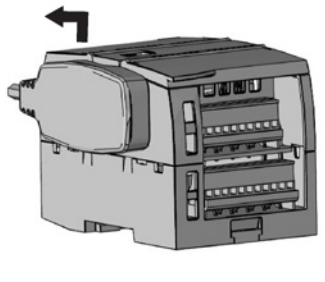
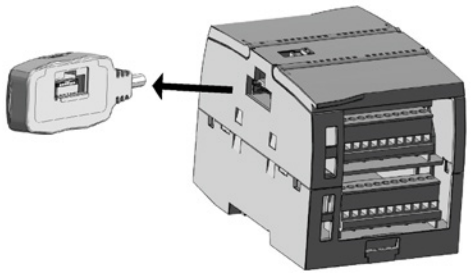
Task	Procedure
  	1. Ensure that the CPU and all S7-1200 equipment are disconnected from electrical power. 2. Place the female connector to the bus connector on the left side of the signal module. 3. Slip the hook extension of the female connector into the housing at the bus connector and press down slightly to engage the hook. 4. Lock the connector into place: – Place a screwdriver beside the tab on the top of the signal module. – Slide the tab fully to the left. To engage the connector, you must slide the connector tab all the way to the left. The connector tab must be locked into place.

Table 4- 14 Removing the female connector of the expansion cable

Task		Procedure
		1. Ensure that the CPU and all S7-1200 equipment are disconnected from electrical power. 2. Unlock the connector: – Place a screwdriver beside the tab on the top of the signal module. – Press down slightly and slide the tab fully to the right. 3. Lift the connector up slightly to disengage the hook extension. 4. Remove the female connector.
		

Note

Installing the expansion cable in a vibration environment

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

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

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

4.3.8 TS (TeleService) adapter

4.3.8.1 Connecting the TeleService adapter

Before installing the TS (TeleService) Adapter IE Basic or TS (TeleService) Adapter IE Advanced, you must first connect the TS Adapter and a TS module.

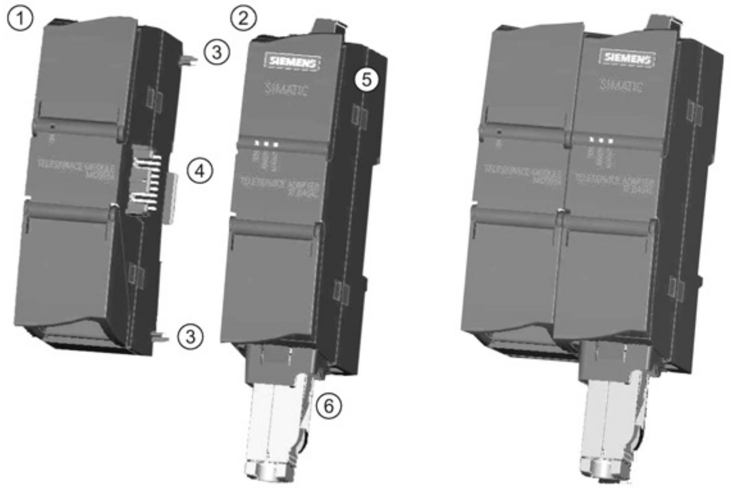
Available TS modules:

- TS module RS232
- TS module Modem
- TS module GSM
- TS module ISDN

Note

The TS module can be damaged if you touch the contacts of the plug connector ④ of the TS module.

Follow ESD guidelines in order to avoid damaging the TS module through electrostatic discharge. Before connecting a TS module and TS Adapter, make sure that both are in an idle state.



①	TS module	④	Plug connector from the TS module
②	TS Adapter	⑤	Cannot be opened
③	Elements	⑥	Ethernet port

Note

Before connecting a TS module and TS adapter basic unit, ensure that the contact pins ④ are not bent.

When connecting, ensure that the male connector and guide pins are positioned correctly.

Only connect a TS module into the TS adapter. Do not force a connection of the TS adapter to a different device, such as an S7-1200 CPU. Do not change the mechanical construction of the connector, and do not remove or damage the guide pins.

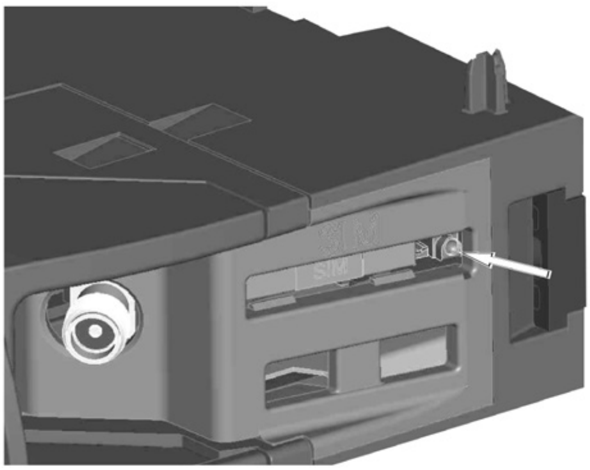
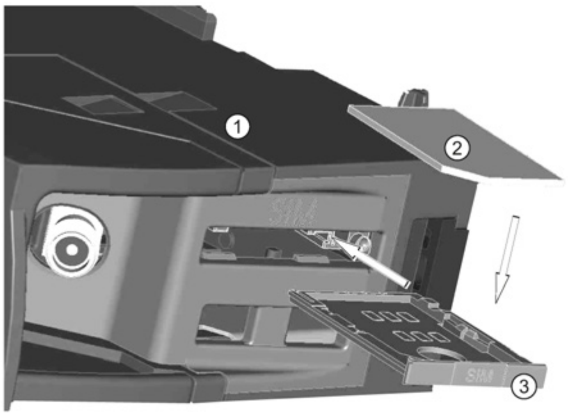
4.3.8.2 Installing the SIM card

Locate the SIM card slot on the underside of the TS module GSM.

Note

The SIM card may only be removed or inserted if the TS module GSM is de-energized.

Table 4- 15 Installing the SIM card

Task	Procedure
	Use a sharp object to press the eject button of the SIM card tray (in the direction of the arrow) and remove the SIM card tray.
	Place the SIM card in the SIM card tray as shown and put the SIM card tray back into its slot.
	① TS Module GSM
	② SIM card
	③ SIM card tray

Note

Ensure that the SIM card is correctly oriented in the card tray. Otherwise, the SIM card will not make connection with the module, and the eject button may not remove the card tray.

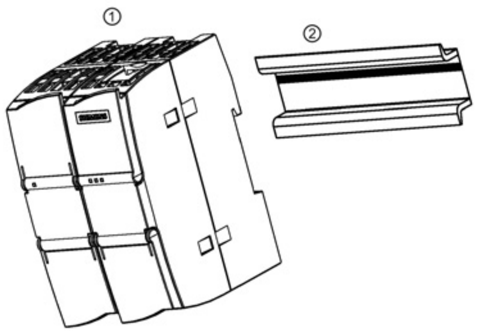
4.3.8.3 Installing the TS adapter unit on a DIN rail

Prerequisites: You must have connected the TS Adapter and a TS module together, and the DIN rail must have been installed.

Note

If you install the TS unit vertically or in high-vibration environment, the TS module can become disconnected from the TS Adapter. Use an end bracket 8WA1808 on the DIN rail to ensure that the modules remain connected.

Table 4- 16 Installing and removing the TS Adapter

Task	Procedure
	Installation: 1. Hook the TS Adapter with attached TS module ① on the DIN rail ②. 2. Rotate the unit back until it engages. 3. Push in the DIN rail clip on each module to attach each module to the rail.
	Removal: 1. Remove the analog cable and Ethernet cable from the underside of the TS Adapter. 2. Remove power from the TS Adapter. 3. Use a screwdriver to disengage the rail clips on both modules. 4. Rotate the unit upwards to remove the unit from the DIN rail.

⚠ WARNING

Safety requirements for installing or removing the TS Adapter.

Before you remove power from the unit, disconnect the grounding of the TS Adapter by removing the analog cable and Ethernet cable. Failure to observe this precaution could result in death, severe personal injury and/or property damage due to unexpected equipment operation.

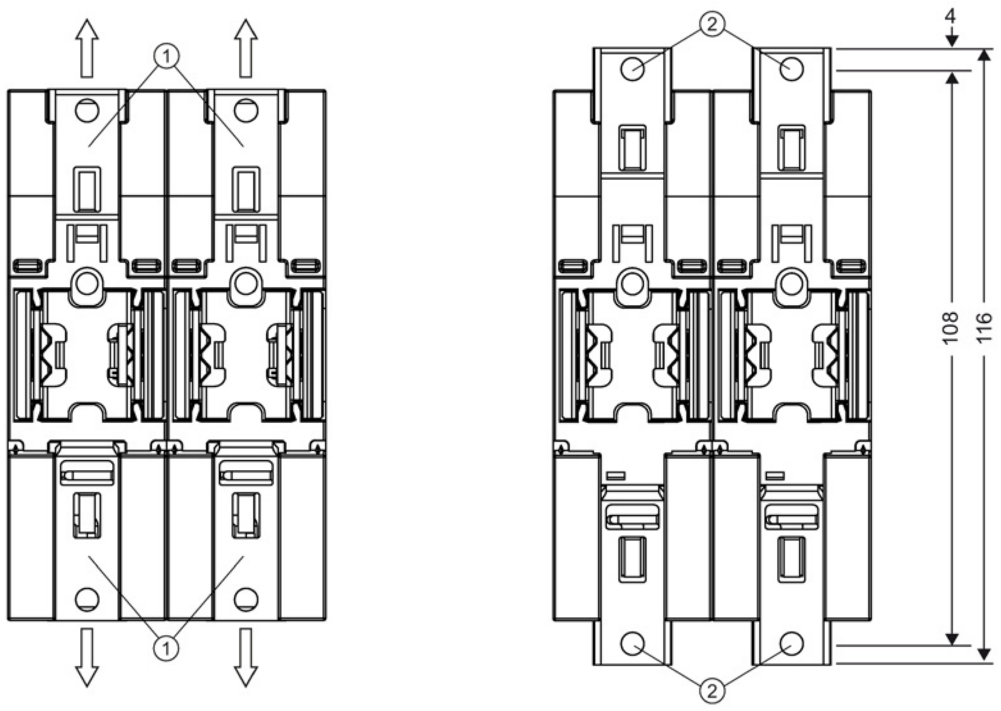
Always follow these requirements when installing or removing the TS Adapter.

4.3.8.4 Installing the TS adapter on a panel

Prerequisites: You must have connected the TS Adapter and TS module.

1. Move the attachment slider ① to the backside of the TS Adapter and TS module in the direction of the arrow until it engages.
2. Screw the TS Adapter and TS module to the position marked with ② to the designated assembly wall.

The following illustration shows the TS Adapter from behind, with the attachment sliders ① in both positions:



①	Attachment slider
②	Drill holes for wall mounting

4.4 Wiring guidelines

Proper grounding and wiring of all electrical equipment is important to help ensure the optimum operation of your system and to provide additional electrical noise protection for your application and the S7-1200. Refer to the technical specifications (Page 1099) for the S7-1200 wiring diagrams.

Prerequisites

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

Ensure that you follow all applicable electrical codes when wiring the S7-1200 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

Installation or wiring the S7-1200 or related equipment with power applied could cause electric shock or unexpected operation of equipment.

Failure to disable all power to the S7-1200 and related equipment during installation or removal procedures could result in death, severe personal injury, and/or damage due to electric shock or unexpected equipment operation.

Always follow appropriate safety precautions and ensure that power to the S7-1200 is disabled before attempting to install or remove the S7-1200 or related equipment.

Always take safety into consideration as you design the grounding and wiring of your S7-1200 system. Electronic control devices, such as the S7-1200, can fail and can cause unexpected operation of the equipment that is being controlled or monitored. For this reason, you should implement safeguards that are independent of the S7-1200 to protect against possible personal injury or equipment damage.

⚠ WARNING

Control devices can fail in an unsafe condition, resulting in unexpected operation of controlled equipment.

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

Use an emergency stop function, electromechanical overrides, or other redundant safeguards that are independent of the S7-1200.

Guidelines for isolation

S7-1200 AC power supply boundaries and I/O boundaries to AC circuits have been designed and approved to provide safe separation between AC line voltages and low voltage circuits. These boundaries include double or reinforced insulation, or basic plus supplementary insulation, according to various standards. Components which cross these boundaries such as optical couplers, capacitors, transformers, and relays have been approved as providing safe separation. Only circuits rated for AC line voltage include safety isolation to other circuits. Isolation boundaries between 24 VDC circuits are functional only, and you should not depend on these boundaries for safety.

The sensor supply output, communications circuits, and internal logic circuits of an S7-1200 with included AC power supply are sourced as SELV (safety extra-low voltage) according to EN 61131-2.

To maintain the safe character of the S7-1200 low voltage circuits, external connections to communications ports, analog circuits, and all 24 VDC nominal power supply and I/O circuits must be powered from approved sources that meet the requirements of SELV, PELV, Class 2, Limited Voltage, or Limited Power according to various standards.

WARNING

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 cause electric shock resulting in death, severe personal injury and/or property damage.

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

Guidelines for grounding the S7-1200

The best way to ground your application is to ensure that all the common and ground connections of your S7-1200 and related equipment are grounded to a single point. This single point should be connected directly to the earth ground for your system.

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

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

Guidelines for wiring the S7-1200

When designing the wiring for your S7-1200, provide a single disconnect switch that simultaneously removes power from the S7-1200 CPU power supply, from all input circuits, and from all output circuits. Provide over-current protection, such as a fuse or circuit breaker, to limit fault currents on supply wiring. Consider providing additional protection by placing a fuse or other current limit in each output circuit.

Install appropriate surge suppression devices for any wiring that could be subject to lightning surges. For more information, see Surge immunity (Page 1099)in the General technical specifications section.

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. The CPU and SM connectors accept wire sizes from 2 mm^2 to 0.3 mm^2 (14 AWG to 22 AWG). Wire strip length is 6.4 mm. The SB connector accepts wire sizes from 1.3 mm^2 to 0.3 mm^2 (16 AWG to 22 AWG). Wire strip length is 6.3 to 7 mm.

Wire and cable should have a temperature rating 30 °C higher than the ambient temperature around the S7-1200 (for example, a minimum of 85 °C-rated conductors for 55 °C ambient temperature). You should determine other wiring type and material requirements from the specific electrical circuit ratings and your installation environment.

Use shielded wires for optimum protection against electrical noise. Typically, grounding the shield at the S7-1200 gives the best results. You should ground communication cable shields to S7-1200 communication connector shells using connectors that engage the cable shield, or by bonding the communication cable shields to a separate ground. You should ground other cable shields using clamps or copper tape around the shield to provide a high surface area connection to the grounding point.

When wiring input circuits that are powered by an external power supply, include an overcurrent protection device in that circuit. External protection is not necessary for circuits that are powered by the 24 VDC sensor supply from the S7-1200 because the sensor supply is already current-limited.

All S7-1200 modules have removable connectors for user wiring. To prevent loose connections, ensure that the connector is seated securely and that the wire is installed securely into the connector. Siemens recommends that the wire insulation is removed from the wire approximately 6 mm to ensure a proper connection.To avoid damaging the connector, be careful that you do not over-tighten the screws. The maximum torque for the CPU and SM connector screw is 0.56 N-m (5 inch-pounds). The maximum torque for the SB, simulator, and potentiometer module connector screw is 0.33 N-m (3 inch-pounds).

To help prevent unwanted current flows in your installation, the S7-1200 provides isolation boundaries at certain points. When you plan the wiring for your system, you should consider these isolation boundaries. Refer to the technical specifications (Page 1099) for the amount of isolation provided and the location of the isolation boundaries. Circuits rated for AC line voltage include safety isolation to other circuits. Isolation boundaries between 24 VDC circuits are functional only, and you should not depend on these boundaries for safety.

See also

General specifications and features (Page 1155)

Guidelines for lamp loads

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

Guidelines for inductive loads

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

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

S7-1200 DC outputs include internal suppressor circuits that are adequate for inductive loads in most applications. Since S7-1200 relay output contacts can be used to switch either a DC or an AC load, internal protection is not provided.

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

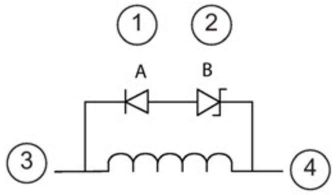
For AC loads, a metal oxide varistor (MOV) or other voltage clamping device may be used with a parallel RC circuit, but is not as effective when used alone. An MOV suppressor with no parallel RC circuit often results in significant high frequency noise up to the clamp voltage.

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

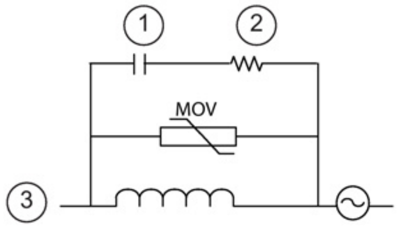
② 8.2 V Zener (DC outputs),
36 V Zener (Relay outputs)

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- 17 AC suppressor circuit resistor and capacitor values

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

Conditions satisfied by the table values:

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

Guidelines for differential inputs and outputs

Differential inputs and outputs behave differently than standard inputs and outputs. There are two pins per differential input and output. Determining whether a differential input or output is on or off requires that you measure the voltage difference between these two pins.

See the detailed specifications for the CPU 1217C in Appendix A (Page 1155).

PLC concepts 5

5.1 Execution of the user program

The CPU supports the following types of code blocks that allow you to create an efficient structure for your user program:

- Organization blocks (OBs) define the structure of the program. Some OBs have predefined behavior and start events, but you can also create OBs with custom start events.
- Functions (FCs) and function blocks (FBs) contain the program code that corresponds to specific tasks or combinations of parameters. Each FC or FB provides a set of input and output parameters for sharing data with the calling block. An FB also uses an associated data block (called an instance DB) to maintain the data values for that instance of the FB call. You can call an FB multiple times, each time with a unique instance DB. Calls to the same FB with different instance DBs do not affect the data values in any of the other instance DBs.
- Data blocks (DBs) store data that can be used by the program blocks.

Execution of the user program begins with one or more optional startup organization blocks (OBs) which execute once upon entering RUN mode, followed by one or more program cycle OBs that execute cyclically. You can also associate an OB with an interrupt event, which can be either a standard event or an error event. These OBs execute whenever the corresponding standard or error event occurs.

A function (FC) or a function block (FB) is a block of program code that can be called from an OB or from another FC or FB, down to the following nesting depths:

- 16 from the program cycle or startup OB
- 6 from any interrupt event OB

FCs are not associated with any particular data block (DB). FBs are tied directly to a DB and use the DB for passing parameters and storing interim values and results.

The size of the user program, data, and configuration is limited by the available load memory and work memory in the CPU. There is no specific limit to the number of each individual OB, FC, FB and DB block. However, the total number of blocks is limited to 1024.

Each cycle includes writing the outputs, reading the inputs, executing the user program instructions, and performing background processing. The cycle is referred to as a scan cycle or scan.

Your S7-1200 automation solution can consist of a central rack with the S7-1200 CPU and additional modules. The term "central rack" refers to either the rail or panel installation of the CPU and associated modules. The modules (SM, SB, BB, CB, CM or CP) are detected and logged in only upon powerup.

- Inserting or removing a module in the central rack under power (hot) is not supported. Never insert or remove a module from the central rack when the CPU has power.

WARNING

Safety requirements for inserting or removing modules

Insertion or removal of a module (SM, SB, BB, CD, CM or CP) from the central rack when the CPU has power could cause unpredictable behavior, resulting in damage to equipment and/or injury to personnel.

Always remove power from the CPU and central rack and follow appropriate safety precautions before inserting or removing a module from the central rack.

- You can insert or remove a SIMATIC memory card while the CPU is under power. However, inserting or removing a memory card when the CPU is in RUN mode causes the CPU to go to STOP mode.

NOTICE

Risks with removing memory card when CPU is in RUN mode.

Insertion or removal of a memory card when the CPU is in RUN mode causes the CPU to go to STOP, which might result in damage to the equipment or the process being controlled.

Whenever you insert or remove a memory card, the CPU immediately goes to STOP mode. Before inserting or removing a memory card, always ensure that the CPU is not actively controlling a machine or process. Always install an emergency stop circuit for your application or process.

- If you insert or remove a module in a distributed I/O rack (AS-i, PROFINET, or PROFIBUS) when the CPU is in RUN mode, the CPU generates an entry in the diagnostics buffer, executes the pull or plug of modules OB if present, and by default remains in RUN mode.

Process image update and process image partitions

The CPU updates local digital and analog I/O points synchronously with the scan cycle using an internal memory area called the process image. The process image contains a snapshot of the physical inputs and outputs (the physical I/O points on the CPU, signal board, and signal modules).

You can configure I/O points to be updated in the process image every scan cycle or when a specific event interrupt occurs. You can also configure an I/O point to be excluded from process image updates. For example, your process might only need certain data values when an event such as a hardware interrupt occurs. By configuring the process image update for these I/O points to be associated with a partition that you assign to a hardware interrupt OB, you avoid having the CPU update data values unnecessarily every scan cycle when your process does not need a continual update.

For I/O that is updated every scan cycle, the CPU performs the following tasks during each scan cycle:

- The CPU writes the outputs from the process image output area to the physical outputs.
- The CPU reads the physical inputs just prior to the execution of the user program and stores the input values in the process image input area. These values thus remain consistent throughout the execution of the user instructions.
- The CPU executes the logic of the user instructions and updates the output values in the process image output area instead of writing to the actual physical outputs.

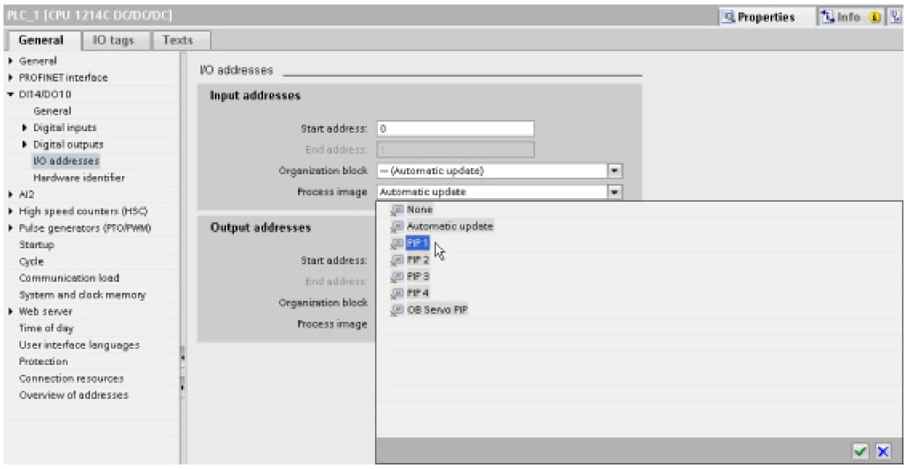
  This process provides consistent logic through the execution of the user instructions for a given cycle and prevents the flickering of physical output points that might change state multiple times in the process image output area.

For controlling whether your process updates I/O points automatically on every scan cycle, or upon the triggering of events, the S7-1200 provides five process image partitions. The first process image partition, PIP0, is designated for I/O that is to be automatically updated every scan cycle, and is the default assignment. You can use the remaining four partitions, PIP1, PIP2, PIP3, and PIP4 for assigning I/O process image updates to various interrupt events. You assign I/O to process image partitions in Device Configuration and you assign process image partitions to interrupt events when you create interrupt OBs (Page 176) or edit OB properties (Page 176).

By default, when you insert a module in the device view, STEP 7 sets its I/O process image update to "Automatic update". For I/O configured for "Automatic update", the CPU handles the data exchange between the module and the process image area automatically during every scan cycle.

To assign digital or analog points to a process image partition, or to exclude I/O points from process image updates, follow these steps:

1. View the Properties tab for the appropriate device in Device configuration.
2. Expand the selections under "General" as necessary to locate the desired I/O points.
3. Select "I/O addresses".
4. Optionally select a specific OB from the "Organization block" drop-down list.
5. From the "Process image" drop-down list, change "Automatic update" to "PIP1", "PIP2", "PIP3", "PIP4" or "None". A selection of "None" means that you can only read from and write to this I/O using immediate instructions. To add the points back to the process image automatic update, change this selection back to "Automatic update".



I/O addresses

Input addresses

Start address: 0
End address: 1
Organization block: --- (Automatic update)
Process image: Automatic update

Output addresses

Start address:
End address:
Organization block:
Process image:

You can immediately read physical input values and immediately write physical output values when an instruction executes. An immediate read accesses the current state of the physical input and does not update the process image input area, regardless of whether the point is configured to be stored in the process image. An immediate write to the physical output updates both the process image output area (if the point is configured to be stored in the process image) and the physical output point. Append the suffix ":P" to the I/O address if you want the program to immediately access I/O data directly from the physical point instead of using the process image.

Note

Use of process image partitions

If you assign I/O to one of the process image partitions PIP1 - PIP4, and do not assign an OB to that partition, then the CPU never updates that I/O to or from the process image. Assigning I/O to a PIP that does not have a corresponding OB assignment, is the same as assigning the process image to "None". You can read the I/O directly from the physical I/O with an immediate read instruction, or write to the physical I/O with an immediate write instruction. The CPU does not update the process image.

The CPU supports distributed I/O for both PROFINET and PROFIBUS networks (Page 611).

5.1.1 Operating modes of the CPU

The CPU has three modes of operation: STOP mode, STARTUP mode, and RUN mode. Status LEDs on the front of the CPU indicate the current mode of operation.

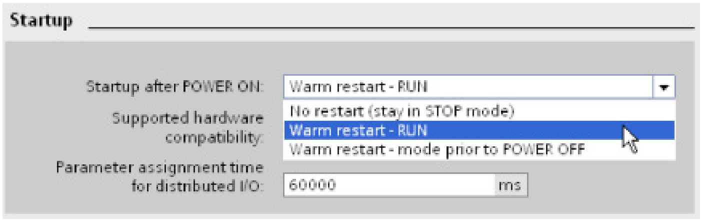
- In STOP mode, the CPU is not executing the program. You can download a project.
- In STARTUP mode, the startup OBs (if present) execute once. The CPU does not process interrupt events during the startup mode.
- In RUN mode, the program cycle OBs execute repeatedly. Interrupt events can occur and the corresponding interrupt event OBs can execute at any point within the RUN mode. You can download some parts of a project in RUN mode (Page 1087).

The CPU supports a warm restart for entering the RUN mode. Warm restart does not include a memory reset. The CPU initializes all non-retentive system and user data at warm restart, and retains the values of all retentive user data.

A memory reset clears all work memory, clears retentive and non-retentive memory areas, copies load memory to work memory, and sets outputs to the configured "Reaction to CPU STOP". A memory reset does not clear the diagnostics buffer or the permanently saved values of the IP address.

You can configure the "startup after POWER ON" setting of the CPU. This configuration item appears under the "Device configuration" for the CPU under "Startup". Upon powering up, the CPU performs a sequence of power-up diagnostic checks and system initialization. During system initialization the CPU deletes all non-retentive bit (M) memory and resets all non-retentive DB contents to the initial values from load memory. The CPU retains retentive bit (M) memory and retentive DB contents and then enters the appropriate operating mode. Certain detected errors prevent the CPU from entering the RUN mode. The CPU supports the following configuration choices:

- No restart (stay in STOP mode)
- Warm restart - RUN
- Warm restart - mode prior to POWER OFF



Startup

Startup after POWER ON: Warm restart - RUN
No restart (stay in STOP mode)
Warm restart - RUN
Warm restart - mode prior to POWER OFF
Supported hardware compatibility:
Parameter assignment time for distributed I/O: 60000 ms

NOTICE

Repairable faults can cause the CPU to enter STOP mode.

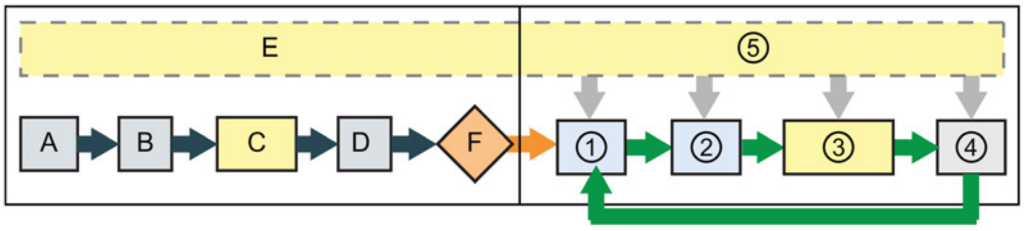
The CPU can enter STOP mode due to repairable faults, such as failure of a replaceable signal module, or temporary faults, such as power line disturbance or erratic power up event. Such conditions could result in property damage.

If you have configured the CPU to "Warm restart - mode prior to POWER OFF", the CPU goes to the operating mode it was in prior to the loss of power or fault. If the CPU was in STOP mode at the time of power loss or fault, the CPU goes to STOP mode on power up and stays in STOP mode until it receives a command to go to RUN mode. If the CPU was in RUN mode at the time of power loss or fault, the CPU goes to RUN mode on the next power up providing it detects no errors that would inhibit a transition to RUN mode.

Configure CPUs that are intended to operate independently of a STEP 7 connection to "Warm restart - RUN" so that the CPU can return to RUN mode on the next power cycle.

You can change the current operating mode using the "STOP" or "RUN" commands (Page 1075) from the online tools of the programming software. You can also include a STP instruction (Page 289) in your program to change the CPU to STOP mode. This allows you to stop the execution of your program based on the program logic.

- In STOP mode, the CPU handles any communication requests (as appropriate) and performs self-diagnostics. The CPU does not execute the user program, and the automatic updates of the process image do not occur.
- In STARTUP and RUN modes, the CPU performs the tasks shown in the following figure.



STARTUP		RUN	
A	Clears the I (image) memory area	①	Writes Q memory to the physical outputs
B	Initializes the Q output (image) memory area with either zero, the last value, or the substitute value, as configured, and zeroes PB, PN, and AS-i outputs	②	Copies the state of the physical inputs to I memory
C	Initializes non-retentive M memory and data blocks to their initial value and enables configured cyclic interrupt and time of day events. Executes the startup OBs.	③	Executes the program cycle OBs
D	Copies the state of the physical inputs to I memory	④	Performs self-test diagnostics
E	Stores any interrupt events into the queue to be processed after entering RUN mode	⑤	Processes interrupts and communications during any part of the scan cycle
F	Enables the writing of Q memory to the physical outputs		

STARTUP processing

Whenever the operating mode changes from STOP to RUN, the CPU clears the process image inputs, initializes the process image outputs and processes the startup OBs. Any read accesses to the process-image inputs by instructions in the startup OBs read zero rather than the current physical input value. Therefore, to read the current state of a physical input during the startup mode, you must perform an immediate read. The startup OBs and any associated FCs and FBs are executed next. If more than one startup OB exists, each is executed in order according to the OB number, with the lowest OB number executing first.

Each startup OB includes startup information that helps you determine the validity of retentive data and the time-of-day clock. You can program instructions inside the startup OBs to examine these startup values and to take appropriate action. The following startup locations are supported by the Startup OBs:

Table 5- 1 Startup locations supported by the startup OB

Input	Data Type	Description
LostRetentive	Bool	This bit is true if the retentive data storage areas have been lost
LostRTC	Bool	This bit is true if the time-of-day clock (Real time Clock) has been lost

The CPU also performs the following tasks during the startup processing.

- Interrupts are queued but not processed during the startup phase
- No cycle time monitoring is performed during the startup phase
- Configuration changes to HSC (high-speed counter), PWM (pulse-width modulation), and PtP (point-to-point communication) modules can be made in startup
- Actual operation of HSC, PWM and point-to-point communication modules only occurs in RUN

After the execution of the startup OBs finishes, the CPU goes to RUN mode and processes the control tasks in a continuous scan cycle.

5.1.2 Processing the scan cycle in RUN mode

For each scan cycle, the CPU writes the outputs, reads the inputs, executes the user program, updates communication modules, and responds to user interrupt events and communication requests. Communication requests are handled periodically throughout the scan.

These actions (except for user interrupt events) are serviced regularly and in sequential order. User interrupt events that are enabled are serviced according to priority in the order in which they occur. For interrupt events, the CPU reads the inputs, executes the OB, and then writes the outputs, using the associated process image partition (PIP), if applicable.

The system guarantees that the scan cycle will be completed in a time period called the maximum cycle time; otherwise a time error event is generated.

- Each scan cycle begins by retrieving the current values of the digital and analog outputs from the process image and then writing them to the physical outputs of the CPU, SB, and SM modules configured for automatic I/O update (default configuration). When a physical output is accessed by an instruction, both the output process image and the physical output itself are updated.
- The scan cycle continues by reading the current values of the digital and analog inputs from the CPU, SB, and SMs configured for automatic I/O update (default configuration), and then writing these values to the process image. When a physical input is accessed by an instruction, the value of the physical input is accessed by the instruction, but the input process image is not updated.
- After reading the inputs, the user program is executed from the first instruction through the end instruction. This includes all the program cycle OBs plus all their associated FCs and FBs. The program cycle OBs are executed in order according to the OB number with the lowest OB number executing first.

Communications processing occurs periodically throughout the scan, possibly interrupting user program execution.

Self-diagnostic checks include periodic checks of the system and the I/O module status checks.

Interrupts can occur during any part of the scan cycle, and are event-driven. When an event occurs, the CPU interrupts the scan cycle and calls the OB that was configured to process that event. After the OB finishes processing the event, the CPU resumes execution of the user program at the point of interruption.

5.1.3 Organization blocks (OBs)

OBs control the execution of the user program. Specific events in the CPU trigger the execution of an organization block. OBs cannot call each other or be called from an FC or FB. Only an event such as a diagnostic interrupt or a time interval, can start the execution of an OB. The CPU handles OBs according to their respective priority classes, with higher priority OBs executing before lower priority OBs. The lowest priority class is 1 (for the main program cycle), and the highest priority class is 26.

5.1.3.1 Program cycle OB

Program cycle OBs execute cyclically while the CPU is in RUN mode. The main block of the program is a program cycle OB. This is where you place the instructions that control your program and where you call additional user blocks. You can have multiple program cycle OBs, which the CPU executes in numerical order. Main (OB1) is the default.

Program cycle events

The program cycle event happens once during each program cycle (or scan). During the program cycle, the CPU writes the outputs, reads the inputs and executes program cycle OBs. The program cycle event is required and is always enabled. You might have no program cycle OBs, or you might have multiple OBs selected for the program cycle event. After the program cycle event occurs, the CPU executes the lowest numbered program cycle OB (usually "Main" OB 1). The CPU executes the other program cycle OBs sequentially (in numerical order) within the program cycle. Program execution is cyclical such that the program cycle event occurs at the following times:

- When the last startup OB finishes execution
- When the last program cycle OB finishes execution

Table 5- 2 Start information for a program cycle OB

Input	Data type	Description
Initial_Call	Bool	True for initial call of the OB
Remanence	Bool	True if retentive data are available

5.1.3.2 Startup OB

Startup OBs execute one time when the operating mode of the CPU changes from STOP to RUN, including powering up in the RUN mode and in commanded STOP-to-RUN transitions. After completion, the main "Program cycle" begins executing.

Startup events

The startup event happens one time on a STOP to RUN transition and causes the CPU to execute the startup OBs. You can configure multiple OBs for the startup event. The startup OBs execute in numerical order.

Table 5- 3 Start information for a startup OB

Input	Data type	Description
LostRetentive	Bool	True if retentive data are lost
LostRTC	Bool	True if date and time are lost

5.1.3.3 Time delay interrupt OB

Time delay interrupt OBs execute after a time delay that you configure.

Time delay interrupt events

You configure time delay interrupt events to occur after a specified delay time has expired. You assign the delay time with the SRT_DINT instruction. The time delay events interrupt the program cycle to execute the corresponding time delay interrupt OB. You can attach only one time delay interrupt OB to a time delay event. The CPU supports four time delay events.

Table 5- 4 Start information for a time delay interrupt OB

Input	Data type	Description
Sign	Word	Identifier passed to triggering call of SRT_DINT

5.1.3.4 Cyclic interrupt OB

Cyclic interrupt OBs execute at a specified interval. You can configure up to a total of four cyclic interrupt events, with one OB corresponding to each cyclic interrupt event.

Cyclic interrupt events

The cyclic interrupt events allow you to configure the execution of an interrupt OB at a configured cycle time. You configure the initial cycle time when you create the cyclic interrupt OB. A cyclic event interrupts the program cycle and executes the corresponding cyclic interrupt OB. Note that the cyclic interrupt event is at a higher priority class than the program cycle event.

You can attach only one cyclic interrupt OB to a cyclic event.

You can assign a phase shift to each cyclic interrupt so that the execution of cyclic interrupts can be offset from one another by the phase offset amount. For example, if you have 1ms cyclic event and a 2ms cyclic event, every two ms both events occur at the same moment. If you phase shift the 1ms event by 500 μsec and the 2ms event by 0 μsec, then the two events never occur at the same moment.

The default phase offset is 0. To change the initial phase shift, or to change the initial cycle time for a cyclic event, right click the cyclic interrupt OB in the project tree, select "Properties" from the context menu, then click "Cyclic interrupt", and enter the new initial values. You can also query and change the scan time and the phase shift from your program using the Query cyclic interrupt (QRY_CINT) and Set cyclic interrupt (SET_CINT) instructions. Scan time and phase shift values set by the SET_CINT instruction do not persist through a power cycle or a transition to STOP mode; scan time and phase shift values return to the initial values following a power cycle or a transition to STOP. The CPU supports a total of four cyclic interrupt events.

5.1.3.5 Hardware interrupt OB

Hardware interrupt OBs execute when the relevant hardware event occurs. A hardware interrupt OB interrupts normal cyclic program execution in reaction to a signal from a hardware event.

Hardware interrupt events

Changes in the hardware, such as a rising or falling edge on an input point, or an HSC (High Speed Counter) event trigger hardware interrupt events. The S7-1200 supports one interrupt OB for each hardware interrupt event. You enable the hardware events in the device configuration, and assign an OB for an event in the device configuration or with an ATTACH instruction in the user program. The CPU supports several hardware interrupt events. The CPU model and the number of input points determine the exact events that are available.

Limits on hardware interrupt events are as follows:

Edges:

- Rising edge events: maximum of 16
- Falling edge events: maximum of 16

HSC events:

- CV=PV: maximum of 6
- Direction changed: maximum of 6
- External reset: maximum of 6

5.1.3.6 Time error interrupt OB

If configured, the time error interrupt OB (OB 80) executes when either the scan cycle exceeds the maximum cycle time or a time error event occurs. If triggered, it executes, interrupting normal cyclic program execution or any other event OB.

The occurrence of either of these events generates a diagnostic buffer entry describing the event. The diagnostic buffer entry is generated regardless of the existence of the time error interrupt OB.

Time error interrupt events

The occurrence of any of several different time error conditions results in a time error event:

- Scan cycle exceeds maximum cycle time

  The "maximum cycle time exceeded" condition results if the program cycle does not complete within the specified maximum scan cycle time. See the section on "Monitoring the cycle time in the S7-1200 System Manual" (Page 101) for more information regarding the maximum cycle time condition, how to configure the maximum scan cycle time in the properties of the CPU, and how to reset the cycle timer.
- CPU cannot start requested OB because a second time interrupt (cyclic or time-delay) starts before the CPU finishes execution of the first interrupt OB
- Queue overflow occurred

  The "queue overflow occurred" condition results if the interrupts are occurring faster than the CPU can process them. The CPU limits the number of pending (queued) events by using a different queue for each event type. If an event occurs when the corresponding queue is full, the CPU generates a time error event.

All time error events trigger the execution of the time error interrupt OB if it exists. If the time error interrupt OB does not exist, then the device configuration of the CPU determines the CPU reaction to the time error:

- The default configuration for time errors, such as starting a second cyclic interrupt before the CPU has finished the execution of the first, is for the CPU to stay in RUN.
- The default configuration for exceeding the maximum time is for the CPU to change to STOP.

The user program can extend the program cycle execution time up to ten times the configured maximum cycle time by executing the RE_TRIGR instruction (Page 288) to restart the cycle time monitor. However, if two "maximum cycle time exceeded" conditions occur within the same program cycle without resetting the cycle timer, then the CPU transitions to STOP, regardless of whether the time error interrupt OB exists. See the section on "Monitoring the cycle time in the S7-1200 System Manual" (Page 101).

Time error interrupt OB includes start information that helps you determine which event and OB generated the time error. You can program instructions inside the OB to examine these start values and to take appropriate action.

Table 5- 5 Start information for the time error OB (OB 80)

Input	Data type	Description
fault_id	BYTE	16#01 - maximum cycle time exceeded 16#02 - requested OB cannot be started 16#07 and 16#09 - queue overflow occurred
csg_OBnr	OB_ANY	Number of the OB which was being executed when the error occurred
csg_prio	UINT	Priority of the OB causing the error

To include a time error interrupt OB in your project, you must add a time error interrupt by double-clicking "Add new block" under "Program blocks" in the tree, then choose "Organization block", and then "Time error interrupt".

The priority for a new V4.0 CPU is 22. If you exchange a V3.0 CPU for a V4.0 CPU (Page 1287), the priority is 26, the priority that was in effect for V3.0. In either case, the priority field is editable and you can set the priority to any value in the range 22 to 26.

5.1.3.7 Diagnostic error interrupt OB

The diagnostic error interrupt OB executes when the CPU detects a diagnostic error, or if a diagnostics-capable module recognizes an error and you have enabled the diagnostic error interrupt for the module. The diagnostic error interrupt OB interrupts the normal cyclic program execution. You can include an STP instruction in the diagnostic error interrupt OB to put the CPU in STOP mode if you desire your CPU to enter STOP mode upon receiving this type of error.

If you do not include a diagnostic error interrupt OB in your program, the CPU ignores the error and stays in RUN mode.

Diagnostic error events

Analog (local), PROFINET, PROFIBUS, and some digital (local) devices are capable of detecting and reporting diagnostic errors. The occurrence or removal of any of several different diagnostic error conditions results in a diagnostic error event. The following diagnostic errors are supported:

- No user power
- High limit exceeded
- Low limit exceeded
- Wire break
- Short circuit

Diagnostic error events trigger the execution of the diagnostic error interrupt OB (OB 82) if it exists. If it does not exist, then the CPU ignores the error.

To include a diagnostic error interrupt OB in your project, you must add a diagnostic error interrupt by double-clicking "Add new block" under "Program blocks" in the tree, then choose "Organization block", and then "Diagnostic error interrupt".

Note

Diagnostic errors for multi-channel local analog devices (I/O, RTD, and Thermocouple)

The diagnostic error interrupt OB can process only one channel's diagnostic error at a time.

If two channels of a multi-channel device have an error, then the second error only triggers the diagnostic error interrupt OB under the following conditions: the first channel error clears, the execution of the diagnostic error interrupt OB that the first error triggered is complete, and the second error still exists.

The diagnostic error interrupt OB includes startup information that helps you determine whether the event is due to the occurrence or removal of an error, and the device and channel which reported the error. You can program instructions inside the diagnostic error interrupt OB to examine these startup values and to take appropriate action.

Note

Diagnostic error OB Start information references the submodule as a whole if no diagnostic event is pending

In V3.0, the start information for an outgoing diagnostic error event always indicated the source of the event. In V4.0, if the outgoing event leaves the submodule with no pending diagnostics, the start information references the submodule as a whole (16#8000) even if the source of the event was a specific channel.

For example, if a wire break triggers a diagnostic error event on channel 2, the fault is then corrected, and the diagnostic error event is cleared, the Start information will not reference channel 2, but the submodule (16#8000).

Table 5- 6 Startup information for the diagnostic error interrupt OB

Input	Data type	Description
IOstate	WORD	IO state of the device: • Bit 0 = 1 if the configuration is correct, and = 0 if the configuration is no longer correct. • Bit 4 = 1 if an error is present (such as a wire break). (Bit 4 = 0 if there is no error.) • Bit 5 = 1 if the configuration is **not** correct, and = 0 if the configuration is correct again. • Bit 6 = 1 if an I/O access error has occurred. Refer to LADDR for the hardware identifier of the I/O with the access error. (Bit 6 = 0 if there is no error.)
LADDR	HW_ANY	Hardware identifier of the device or functional unit that reported the error[1]
Channel	UINT	Channel number
MultiError	BOOL	TRUE if more than one error is present

[1] The LADDR input contains the hardware identifier of the device or functional unit which returned the error. The hardware identifier is assigned automatically when components are inserted in the device or network view and appears in the Constants tab of PLC tags. A name is also assigned automatically for the hardware identifier. These entries in the Constants tab of the PLC tags cannot be changed.

5.1.3.8 Pull or plug of modules OB

The "Pull or plug of modules" OB executes when a configured and non-disabled distributed I/O module or submodule (PROFIBUS, PROFINET, AS-i) generates an event related to inserting or removing a module.

Pull or plug of modules event

The following conditions generate a pull of plug of modules event:

- Someone removes or inserts a configured module
- A configured module is not physically present in an expansion rack
- An incompatible module is in an expansion rack that does not correspond to the configured module
- A compatible module for a configured module is in an expansion rack, but the configuration does not allow substitutes
- A module or submodule has parameterization errors

If you have not programmed this OB, the CPU remains in RUN mode when any of these conditions occur with a configured and non-disabled distributed I/O module.

Regardless of whether you have programmed this OB, the CPU changes to STOP mode when any of these conditions occur with a module in the central rack.

Table 5- 7 Start information for pull or plug of modules OB

Input	Data type	Description
LADDR	HW_IO	Hardware identifier
Event_Class	Byte	16#38: module inserted 16#29: module removed
Fault_ID	Byte	Fault identifier

5.1.3.9 Rack or station failure OB

The "Rack or station failure" OB executes when the CPU detects the failure or communication loss of a distributed rack or station.

Rack or station failure event

The CPU generates a rack or station failure event when it detects one of the following:

- The failure of a DP master system or of a PROFINET IO system (in the case of either an incoming or an outgoing event).
- The failure of a DP slave or of an IO device (in the case of either an incoming or an outgoing event)
- Failure of some of the submodules of a PROFINET I-device

If you have not programmed this OB, the CPU remains in RUN mode when any of these conditions occur.

Table 5- 8 Start information for rack or station failure OB

Input	Data type	Description
LADDR	HW_IO	Hardware identifier
Event_Class	Byte	16#32: Activation of a DP slave or an IO device 16#33: Deactivation of a DP slave or an IO device 16#38: outgoing event 16#39: incoming event
Fault_ID	Byte	Fault identifier

5.1.3.10 Time of day OB

Time of day OBs execute based on configured clock time conditions. The CPU supports two time of day OBs.

Time of day events

You can configure a time of day interrupt event to occur once on a specified date or time or cyclically with one of the following cycles:

- Every minute: The interrupt occurs every minute.
- Hourly: The interrupt occurs every hour.
- Daily: The interrupt occurs every day at a specified time (hour and minute).
- Weekly: The interrupt occurs every week at a specified time on a specified day of the week (for example, every Tuesday at 4:30 in the afternoon).
- Monthly: The interrupt occurs every month at a specified time on a specified day of the month. The day number must be between 1 and 28, inclusive.
- Every end of month: The interrupt occurs on the last day of every month at a specified time.
- Yearly: The interrupt occurs every year on the specified date (month and day). You cannot specify a date of February 29.

Table 5- 9 Start information for a time of day event OB

Input	Data type	Description
CaughtUp	Bool	OB call is caught up because time was set forward
SecondTimes	Bool	OB call is started a second time because time was set backward

5.1.3.11 Status OB

Status OBs execute if a DPV1 or PNIO slave triggers a status interrupt. This might be the case if a component (module or rack) of a DPV1 or PNIO slave changes its operating mode, for example from RUN to STOP.

Status events

For detailed information on events that trigger a status interrupt, refer to the manufacturer's documentation for the DPV1 or PNIO slave.

Table 5- 10 Start information for status OB

Input	Data type	Description
LADDR	HW_IO	Hardware identifier
Slot	UInt	Slot number
Specifier	Word	Alarm specifier

5.1.3.12 Update OB

Update OBs execute if a DPV1 or PNIO slave triggers an update interrupt.

Update events

For detailed information on events that trigger an update interrupt, refer to the manufacturer's documentation for the DPV1 or PNIO slave.

Table 5- 11 Start information for update OB

Input	Data type	Description
LADDR	HW_IO	Hardware identifier
Slot	UInt	Slot number
Specifier	Word	Alarm specifier

5.1.3.13 Profile OB

Profile OBs execute if a DPV1 or PNIO slave triggers a profile-specific interrupt.

Profile events

For detailed information on events that trigger a profile interrupt, refer to the manufacturer's documentation for the DPV1 or PNIO slave.

Table 5- 12 Start information for profile OB

Input	Data type	Description
LADDR	HW_IO	Hardware identifier
Slot	UInt	Slot number
Specifier	Word	Alarm specifier

5.1.3.14 Event execution priorities and queuing

The CPU processing is controlled by events. An event triggers an interrupt OB to be executed. You can specify the interrupt OB for an event during the creation of the block, during the device configuration, or with an ATTACH or DETACH instruction. Some events happen on a regular basis like the program cycle or cyclic events. Other events happen only a single time, like the startup event and time delay events. Some events happen when the hardware triggers an event, such as an edge event on an input point or a high speed counter event. Events like the diagnostic error and time error event only happen when an error occurs. The event priorities and queues are used to determine the processing order for the event interrupt OBs.

The CPU processes events in order of priority where 1 is the lowest priority and 26 is the highest priority. Prior to V4.0 of the S7-1200 CPU, each type of OB belonged to a fixed priority class (1 to 26). From V4.0 forward, you can assign a priority class to each OB that you configure. You configure the priority number in the attributes of the OB properties.

Interruptible and non-interruptible execution modes

OBs (Page 88) execute in priority order of the events that trigger them. From V4.0 forward, you can configure OB execution to be interruptible or non-interruptible. Note that program cycle OBs are always interruptible, but you can configure all other OBs to be either interruptible or non-interruptible.

If you set interruptible mode, then if an OB is executing and a higher priority event occurs before the OB completes its execution, the running OB is interrupted to allow the higher-priority event OB to run. The higher-priority event runs, and at its completion, the OB that was interrupted continues. When multiple events occur while an interruptible OB is executing, the CPU processes those events in priority order.

If you do not set interruptible mode, then an OB runs to completion when triggered regardless of any other events that trigger during the time that it is running.

Consider the following two cases where interrupt events trigger a cyclic OB and a time delay OB. In both cases, the time delay OB (OB201) has no process image partition assignment (Page 79) and executes at priority 4. The cyclic OB (OB200) has a process image partition assignment of PIP1 and executes at priority 2. The following illustrations show the difference in execution between non-interruptible and interruptible execution modes:

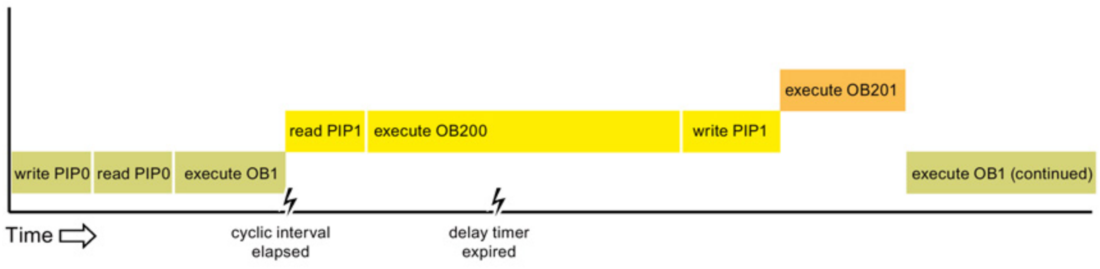


Figure 5-1 Case 1: Non-interruptible OB execution

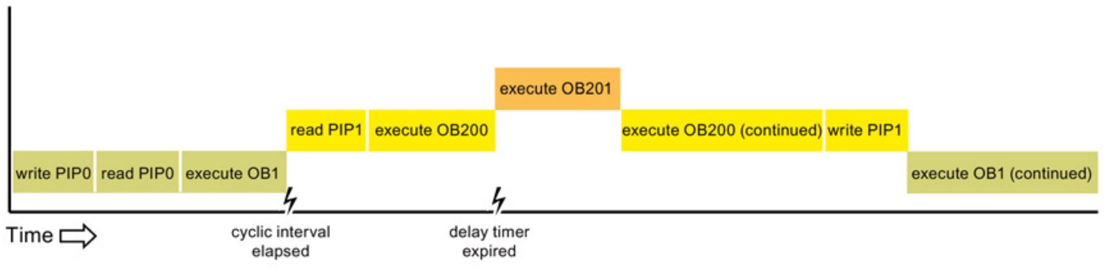


Figure 5-2 Case 2: Interruptible OB execution

Note

If you configure the OB execution mode to be non-interruptible, then a time error OB cannot interrupt OBs other than program cycle OBs. Prior to V4.0 of the S7-1200 CPU, a time error OB could interrupt any executing OB. From V4.0 forward, you must configure OB execution to be interruptible if you want a time error OB (or any other higher priority OB) to be able to interrupt executing OBs that are not program cycle OBs.

Understanding event execution priorities and queuing

The CPU limits the number of pending (queued) events from a single source, using a different queue for each event type. Upon reaching the limit of pending events for a given event type, the next event is lost. You can use a time error interrupt OB (Page 91) to respond to queue overflows.

Each CPU event has an associated priority. In general, the CPU services events in order of priority (highest priority first). The CPU services events of the same priority on a "first-come, first-served" basis.

Table 5- 13 OB events

Event	Quantity allowed	Default OB priority
Program cycle	1 program cycle event Multiple OBs allowed	1[4]
Startup	1 startup event [1] Multiple OBs allowed	1[4]
Time delay	Up to 4 time events 1 OB per event	3
Cyclic interrupt	Up to 4 events 1 OB per event	8
Hardware interrupt	Up to 50 hardware interrupt events[2]	18
	1 OB per event, but you can use the same OB for multiple events	18
Time error	1 event (only if configured)[3]	22 or 26[4]
Diagnostic error	1 event (only if configured)	5
Pull or plug of modules	1 event	6
Rack or station failure	1 event	6
Time of day	Up to 2 events	2
Status	1 event	4
Update	1 event	4
Profile	1 event	4

[1] The startup event and the program cycle event never occur at the same time because the startup event runs to completion before the program cycle event starts.

[2] You can have more than 50 hardware interrupt event OBs if you use the DETACH and ATTACH instructions.

[3] You can configure the CPU to stay in RUN if the scan cycle exceeds the maximum scan cycle time or you can use the RE_TRIGR instruction to reset the cycle time. However, the CPU goes to STOP mode the second time that one scan cycle exceeds the maximum scan cycle time.

[4] The priority for a new V4.0 or V4.1 CPU is 22. If you exchange a V3.0 CPU for a V4.0 or V4.1 CPU, the priority is 26: the priority that was in effect for V3.0. In either case, the priority field is editable and you can set the priority to any value in the range 22 to 26.

Refer to the topic "Exchanging a V3.0 CPU for a V4.1 CPU (Page 1287)" for more details.

In addition, the CPU recognizes other events that do not have associated OBs. The following table describes these events and the corresponding CPU actions:

Table 5- 14 Additional events

Event	Description	CPU action
I/O access error	Direct I/O read/write error	The CPU logs the first occurrence in the diagnostic buffer and stays in RUN mode.
Max cycle time error	CPU exceeds the configured cycle time twice	The CPU logs the error in the diagnostic buffer and transitions to STOP mode.
Peripheral access error	I/O error during process image update	The CPU logs the first occurrence in the diagnostic buffer and stays in RUN mode.
Programming error	program execution error	If the block with the error provides error handling, it updates the error structure; if not, the CPU logs the error in the diagnostic buffer and stays in RUN mode.

Interrupt latency

The interrupt event latency (the time from notification of the CPU that an event has occurred until the CPU begins execution of the first instruction in the OB that services the event) is approximately 175 µsec, provided that a program cycle OB is the only event service routine active at the time of the interrupt event.

5.1.4 Monitoring and configuring the cycle time

The cycle time is the time that the CPU operating system requires to execute the cyclic phase of the RUN mode. The CPU provides two methods of monitoring the cycle time:

- Maximum scan cycle time
- Minimum scan cycle time

Scan cycle monitoring begins after the startup event is complete. Configuration for this feature appears under the "Device Configuration" for the CPU under "Cycle time".

The CPU always monitors the scan cycle and reacts if the maximum scan cycle time is exceeded. If the configured maximum scan cycle time is exceeded, an error is generated and is handled one of two ways:

- If the user program does not include a time error interrupt OB, then the timer error event generates a diagnostic buffer entry, but the CPU remains in RUN mode. (You can change the configuration of the CPU to go to STOP mode when it detects a time error, but the default configuration is to remain in RUN mode.)
- If the user program includes a time error interrupt OB, then the CPU executes it.

The RE_TRIGR instruction (Page 288) (re-trigger cycle time monitoring) allows you to reset the timer that measures the cycle time. If the elapsed time for the current program cycle execution is less than ten times the configured maximum scan cycle time, the RE_TRIGR instruction retriggers the cycle time monitoring and returns with ENO = TRUE. If not, the RE_TRIGR instruction does not retrigger the cycle time monitoring. It returns ENO = FALSE.

Typically, the scan cycle executes as fast as it can be executed and the next scan cycle begins as soon as the current one completes. Depending upon the user program and communication tasks, the time period for a scan cycle can vary from scan to scan. To eliminate this variation, the CPU supports an optional minimum scan cycle time. If you enable this optional feature and provide a minimum scan cycle time in ms, then the CPU delays after the execution of the program cycle OBs until the minimum scan cycle time elapses before repeating the program cycle.

In the event that the CPU completes the normal scan cycle in less time than the specified minimum cycle time, the CPU spends the additional time of the scan cycle performing runtime diagnostics and/or processing communication requests.

In the event that the CPU does not complete the scan cycle in the specified minimum cycle time, the CPU completes the scan normally (including communication processing) and does not create any system reaction as a result of exceeding the minimum scan time. The following table defines the ranges and defaults for the cycle time monitoring functions.

Table 5- 15 Range for the cycle time

Cycle time	Range (ms)	Default
Maximum scan cycle time[1]	1 to 6000	150 ms
Minimum scan cycle time[2]	1 to maximum scan cycle time	Disabled

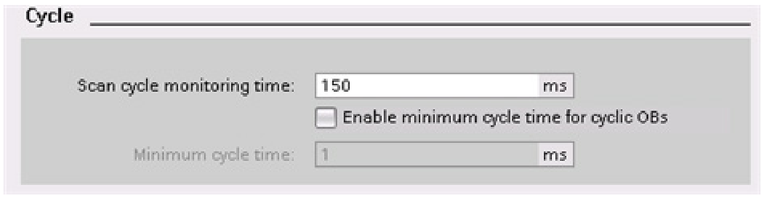
1 The maximum scan cycle time is always enabled. Configure a cycle time between 1 ms to 6000 ms. The default is 150 ms.

2 The minimum scan cycle time is optional, and is disabled by default. If required, configure a cycle time between 1 ms and the maximum scan cycle time.

Configuring the cycle time and communication load

You use the CPU properties in the Device configuration to configure the following parameters:

- Cycle: You can enter a maximum scan cycle monitoring time. You can also enable and enter a minimum scan cycle time.



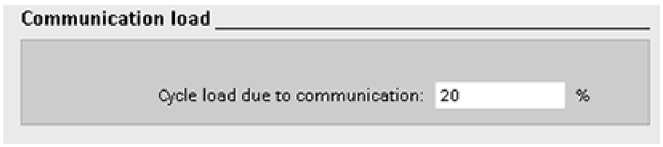
Cycle

Scan cycle monitoring time: 150 ms

☐ Enable minimum cycle time for cyclic OBs

Minimum cycle time: 1 ms

- Communication load: You can configure a percentage of the time to be dedicated for communication tasks.



Communication load

Cycle load due to communication: 20 %

For more information about the scan cycle, see "Monitoring the cycle time". (Page 101)

5.1.5 CPU memory

Memory management

The CPU provides the following memory areas to store the user program, data, and configuration:

- Load memory is non-volatile storage for the user program, data and configuration. When you download a project to the CPU, the CPU first stores the program in the Load memory area. This area is located either in a memory card (if present) or in the CPU. The CPU maintains this non-volatile memory area through a power loss. The memory card supports a larger storage space than that built-in to the CPU.
- Work memory is volatile storage for some elements of the user project while executing the user program. The CPU copies some elements of the project from load memory into work memory. This volatile area is lost when power is removed, and is restored by the CPU when power is restored.
- Retentive memory is non-volatile storage for a limited quantity of work memory values. The CPU uses the retentive memory area to store the values of selected user memory locations during power loss. When a power down or power loss occurs, the CPU restores these retentive values upon power up.

To display the memory usage for a compiled program block, right-click the block in the "Program blocks" folder in the STEP 7 project tree and select "Resources" from the context menu. The Compiliation properties display the load memory and work memory for the compiled block.

To display the memory usage for the online CPU, double-click "Online and diagnostics" in STEP 7, expand "Diagnostics", and select "Memory".

Retentive memory

You can avoid data loss after power failure by marking certain data as retentive. The CPU allows you to configure the following data as retentive:

- Bit memory (M): You can define the size of retentive memory for bit memory in the PLC tag table or in the assignment list. Retentive bit memory always starts at MB0 and runs consecutively up through a specified number of bytes. Specify this value from the PLC tag table or in the assignment list by clicking the "Retain" toolbar icon. Enter the number of M bytes to retain starting at MB0.

  Note: For any block, you can display the assignment list by selecting a block in the Program Blocks folder and then selecting he **Tools > Assignment list** menu command.
- Tags of a function block (FB): If an FB was created with "Optimized" selected, then the interface editor for this FB includes a "Retain" column. In this column, you can select either "Retentive", "Non-retentive", or "Set in IDB" individually for each tag. An instance DB that was created when this FB is placed in the program editor shows this retain column as well. You can only change the retentive state of a tag from within the instance DB interface editor if you selected "Set in IDB" (Set in instance data block) in the Retain selection for the tag in the optimized FB.

If an FB was created with "Standard - compatible with S7-300/400" selected, then the interface editor for this FB does not include a "Retain" column. An instance DB created when this FB is inserted in the program editor shows a "Retain" column which is available for edit. In this case, selecting the "Retain" option for any tag results in **all** tags being selected. Similarly, deselecting the option for any tag results in **all** tags being deselected. For an FB that was configured to be "Standard - compatible with S7-300/400", you can change the retentive state from within the instance DB editor, but all tags are set to the same retentive state together.

After you create the FB, you cannot change the option for "Standard - compatible with S7-300/400". You can only select this option when you create the FB. To determine whether an existing FB was configured for "Optimized" or "Standard - compatible with S7-300/400", right-click the FB in the Project tree, select "Properties", and then select "Attributes". The check box "Optimized block access" when selected shows you whether a block is optimized. Otherwise, it is standard and compatible with S7-300/400 CPUs.

- Tags of a global data block: The behavior of a global DB with regard to retentive state assignment is similar to that of an FB. Depending on the block access setting you can define the retentive state either for individual tags or for all tags of a global data block.
  - If you select "Optimized" when you create the DB, you can set the retentive state for each individual tag.
  - If you select "Standard - compatible with S7-300/400" when you create the DB, the retentive-state setting applies to all tags of the DB; either all tags are retentive or no tag is retentive.

The CPU supports a total of 10240 bytes of retentive data. To see how much is available, from the PLC tag table or the assignment list, click the "Retain" toolbar icon. Although this is where the retentive range is specified for M memory, the second row indicates the total remaining memory available for M and DB combined. Note that for this value to be accurate, you must compile all data blocks with retentive tags.

Note

Downloading a program does not clear or make any changes to existing values in retentive memory. If you want to clear retentive memory before a download, then reset your CPU to factory settings prior to downloading the program.

5.1.5.1 System and clock memory

You use the CPU properties to enable bytes for "system memory" and "clock memory". Your program logic can reference the individual bits of these functions by their tag names.

- You can assign one byte in M memory for system memory. The byte of system memory provides the following four bits that can be referenced by your user program by the following tag names:
  - First cycle: (Tag name "FirstScan") bit is set to1 for the duration of the first scan after the startup OB finishes. (After the execution of the first scan, the "first scan" bit is set to 0.)
  - Diagnostics status changed: (Tag name: "DiagStatusUpdate") is set to 1 for one scan after the CPU logs a diagnostic event. Because the CPU does not set the "DiagStatusUpdate" bit until the end of the first execution of the program cycle OBs, your user program cannot detect if there has been a diagnostic change either during the execution of the startup OBs or the first execution of the program cycle OBs.
  - Always 1 (high): (Tag name "AlwaysTRUE") bit is always set to 1.
  - Always 0 (low): (Tag name "AlwaysFALSE") bit is always set to 0.
- You can assign one byte in M memory for clock memory. Each bit of the byte configured as clock memory generates a square wave pulse. The byte of clock memory provides 8 different frequencies, from 0.5 Hz (slow) to 10 Hz (fast). You can use these bits as control bits, especially when combined with edge instructions, to trigger actions in the user program on a cyclic basis.

The CPU initializes these bytes on the transition from STOP mode to STARTUP mode. The bits of the clock memory change synchronously to the CPU clock throughout the STARTUP and RUN modes.

CAUTION

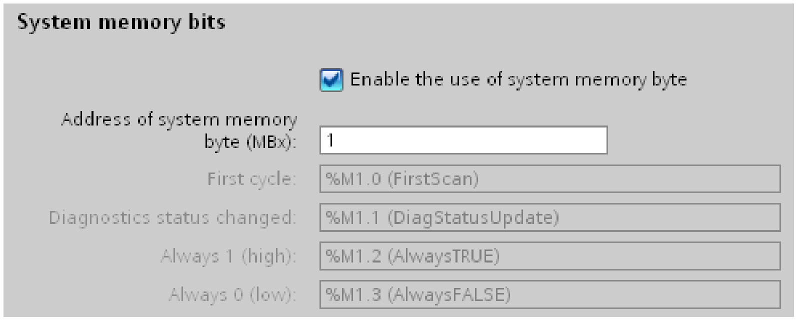
Risks with overwriting the system memory or clock memory bits

Overwriting the system memory or clock memory bits can corrupt the data in these functions and cause your user program to operate incorrectly, which can cause damage to equipment and injury to personnel.

Because both the clock memory and system memory are unreserved in M memory, instructions or communications can write to these locations and corrupt the data.

Avoid writing data to these locations to ensure the proper operation of these functions, and always implement an emergency stop circuit for your process or machine.

System memory configures a byte with bits that turn on (value = 1) for a specific event.


System memory bits

☑ Enable the use of system memory byte

Address of system memory byte (MBx): 1

First cycle: %M1.0 (FirstScan)

Diagnostics status changed: %M1.1 (DiagStatusUpdate)

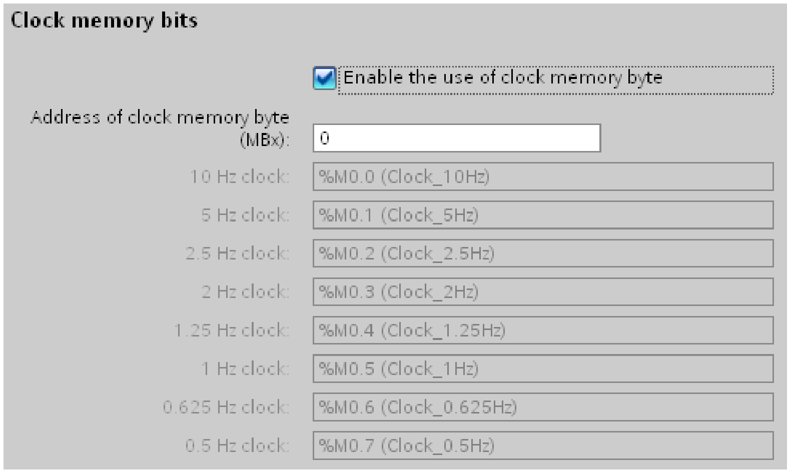
Always 1 (high): %M1.2 (AlwaysTRUE)

Always 0 (low): %M1.3 (AlwaysFALSE)

Table 5- 16 System memory

7	6	5	4	3	2	1	0
Reserved Value 0				Always off Value 0	Always on Value 1	Diagnostic status indicator • 1: Change • 0: No change	First scan indicator • 1: First scan after startup • 0: Not first scan

Clock memory configures a byte that cycles the individual bits on and off at fixed intervals. Each clock bit generates a square wave pulse on the corresponding M memory bit. These bits can be used as control bits, especially when combined with edge instructions, to trigger actions in the user code on a cyclic basis.


Clock memory bits

☑ Enable the use of clock memory byte

Address of clock memory byte (MBx): 0

10 Hz clock: %M0.0 (Clock_10Hz)

5 Hz clock: %M0.1 (Clock_5Hz)

2.5 Hz clock: %M0.2 (Clock_2.5Hz)

2 Hz clock: %M0.3 (Clock_2Hz)

1.25 Hz clock: %M0.4 (Clock_1.25Hz)

1 Hz clock: %M0.5 (Clock_1Hz)

0.625 Hz clock: %M0.6 (Clock_0.625Hz)

0.5 Hz clock: %M0.7 (Clock_0.5Hz)

Table 5- 17 Clock memory

Bit number	7	6	5	4	3	2	1	0
Tag name								
Period (s)	2.0	1.6	1.0	0.8	0.5	0.4	0.2	0.1
Frequency (Hz)	0.5	0.625	1	1.25	2	2.5	5	10

Because clock memory runs asynchronously to the CPU cycle, the status of the clock memory can change several times during a long cycle.

5.1.6 Diagnostics buffer

The CPU supports a diagnostics buffer which contains an entry for each diagnostic event. Each entry includes a date and time the event occurred, an event category, and an event description. The entries are displayed in chronological order with the most recent event at the top. Up to 50 most recent events are available in this log. When the log is full, a new event replaces the oldest event in the log. When power is lost, the events are saved.

The following types of events are recorded in the diagnostics buffer:

- Each system diagnostic event; for example, CPU errors and module errors
- Each state change of the CPU (each power up, each transition to STOP, each transition to RUN)

To access the diagnostics buffer (Page 1076), you must be online. From the "Online & diagnostics" view, locate the diagnostics buffer under "Diagnostics > Diagnostics buffer".

5.1.7 Time of day clock

The CPU supports a time-of-day clock. A super-capacitor supplies the energy required to keep the clock running during times when the CPU is powered down. The super-capacitor charges while the CPU has power. After the CPU has been powered up at least 24 hours, then the super-capacitor has sufficient charge to keep the clock running for typically 20 days.

STEP 7 sets the time-of-day clock to system time, which has a default value out of the box or following a factory reset. To utilize the time-of-day clock, you must set it. Timestamps such as those for diagnostic buffer entries, data log files, and data log entries are based on the system time. You set the time of day from the "Set time of day" function (Page 1072) in the "Online & diagnostics" view of the online CPU. STEP 7 then calculates the system time from the time you set plus or minus the Windows operating system offset from UTC (Coordinated Universal Time). Setting the time of day to the current local time produces a system time of UTC if your Windows operating system settings for time zone and daylight savings time correspond to your locale.

STEP 7 includes instructions (Page 318) to read and write the system time (RD_SYS_T and WR_SYS_T), to read the local time (RD_LOC_T), and to set the time zone (SET_TIMEZONE). The RD_LOC_T instruction calculates local time using the time zone and daylight saving time offsets that you set in the "Time of day" configuration in the general properties of the CPU (Page 162). These settings enable you to set your time zone for local time, optionally enable daylight saving time, and specify the start and end dates and times for daylight saving time. You can also use the SET_TIMEZONE instructions to configure these settings.

5.1.8 Configuring the outputs on a RUN-to-STOP transition

You can configure the behavior of the digital and analog outputs when the CPU is in STOP mode. For any output of a CPU, SB or SM, you can set the outputs to either freeze the value or use a substitute value:

- Substituting a specified output value (default): You enter a substitute value for each output (channel) of that CPU, SB, or SM device.

  The default substitute value for digital output channels is OFF, and the default substitute value for analog output channels is 0.
- Freezing the outputs to remain in last state: The outputs retain their current value at the time of the transition from RUN to STOP. After power up, the outputs are set to the default substitute value.

You configure the behavior of the outputs in Device Configuration. Select the individual devices and use the "Properties" tab to configure the outputs for each device.

Note

Some distibuted I/O modules offer additional settings for the reaction to CPU stop mode. Select from the list of choices in Device Configuration for those modules.

When the CPU changes from RUN to STOP, the CPU retains the process image and writes the appropriate values for both the digital and analog outputs, based upon the configuration.

5.2 Data storage, memory areas, I/O and addressing

5.2.1 Accessing the data of the S7-1200

STEP 7 facilitates symbolic programming. You create symbolic names or "tags" for the addresses of the data, whether as PLC tags relating to memory addresses and I/O points or as local variables used within a code block. To use these tags in your user program, simply enter the tag name for the instruction parameter.

For a better understanding of how the CPU structures and addresses the memory areas, the following paragraphs explain the "absolute" addressing that is referenced by the PLC tags. The CPU provides several options for storing data during the execution of the user program:

- Global memory: The CPU provides a variety of specialized memory areas, including inputs (I), outputs (Q) and bit memory (M). This memory is accessible by all code blocks without restriction.
- PLC tag table: You can enter symbolic names in the STEP 7 PLC tag table for specific memory locations. These tags are global to the STEP 7 program and allow programming with names that are meaningful for your application.
- Data block (DB): You can include DBs in your user program to store data for the code blocks. The data stored persists when the execution of the associated code block comes to an end. A "global" DB stores data that can be used by all code blocks, while an instance DB stores data for a specific FB and is structured by the parameters for the FB.
- Temp memory: Whenever a code block is called, the operating system of the CPU allocates the temporary, or local, memory (L) to be used during the execution of the block. When the execution of the code block finishes, the CPU reallocates the local memory for the execution of other code blocks.

Each different memory location has a unique address. Your user program uses these addresses to access the information in the memory location. References to the input (I) or output (Q) memory areas, such as I0.3 or Q1.7, access the process image. To immediately access the physical input or output, append the reference with ":P" (such as I0.3:P, Q1.7:P, or "Stop:P").

Table 5- 18 Memory areas

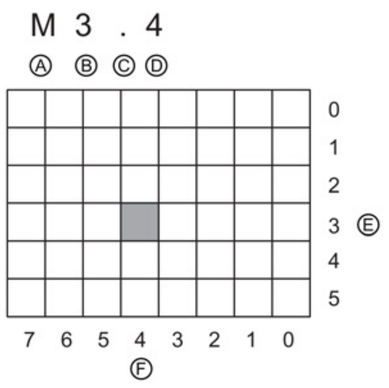
Memory area	Description	Force	Retentive
I Process image input	Copied from physical inputs at the beginning of the scan cycle	No	No
I_:P [1] (Physical input)	Immediate read of the physical input points on the CPU, SB, and SM	Yes	No
Q Process image output	Copied to physical outputs at the beginning of the scan cycle	No	No
Q_:P [1] (Physical output)	Immediate write to the physical output points on the CPU, SB, and SM	Yes	No
M Bit memory	Control and data memory	No	Yes (optional)
L Temp memory	Temporary data for a block local to that block	No	No
DB Data block	Data memory and also parameter memory for FBs	No	Yes (optional)

[1] To immediately access (read or write) the physical inputs and physical outputs, append a ":P" to the address or tag (such as I0.3:P, Q1.7:P, or "Stop:P").

Each different memory location has a unique address. Your user program uses these addresses to access the information in the memory location. The absolute address consists of the following elements:

- Memory area identifier (such as I, Q, or M)
- Size of the data to be accessed ("B' for Byte, "W" for Word, or "D" for DWord)
- Starting address of the data (such as byte 3 or word 3)

When accessing a bit in the address for a Boolean value, you do not enter a mnemonic for the size. You enter only the memory area, the byte location, and the bit location for the data (such as I0.0, Q0.1, or M3.4).



A	Memory area identifier	E	Bytes of the memory area
B	Byte address: byte 3	F	Bits of the selected byte
C	Separator ("byte.bit")		
D	Bit location of the byte (bit 4 of 8)		

In the example, the memory area and byte address (M = bit memory area, and 3 = Byte 3) are followed by a period (".") to separate the bit address (bit 4).

Accessing the data in the memory areas of the CPU

STEP 7 facilitates symbolic programming. Typically, tags are created either in PLC tags, a data block, or in the interface at the top of an OB, FC, or FB. These tags include a name, data type, offset, and comment. Additionally, in a data block, a start value can be specified. You can use these tags when programming by entering the tag name at the instruction parameter. Optionally you can enter the absolute operand (memory area, size and offset) at the instruction parameter. The examples in the following sections show how to enter absolute operands. The % character is inserted automatically in front of the absolute operand by the program editor. You can toggle the view in the program editor to one of these: symbolic, symbolic and absolute, or absolute.

I (process image input): The CPU samples the peripheral (physical) input points just prior to the cyclic OB execution of each scan cycle and writes these values to the input process image. You can access the input process image as bits, bytes, words, or double words. Both read and write access is permitted, but typically, process image inputs are only read.

Table 5- 19 Absolute addressing for I memory

Bit	I[byte address].[bit address]	I0.1
Byte, Word, or Double Word	I[size][starting byte address]	IB4, IW5, or ID12

By appending a ":P" to the address, you can immediately read the digital and analog inputs of the CPU, SB, SM or distributed module. The difference between an access using I_:P instead of I is that the data comes directly from the points being accessed rather than from the input process image. This I_:P access is referred to as an "immediate read" access because the data is retrieved immediately from the source instead of from a copy that was made the last time the input process image was updated.

Because the physical input points receive their values directly from the field devices connected to these points, writing to these points is prohibited. That is, I_:P accesses are read-only, as opposed to I accesses which can be read or write.

I_:P accesses are also restricted to the size of inputs supported by a single CPU, SB, or SM, rounded up to the nearest byte. For example, if the inputs of a 2 DI / 2 DQ SB are configured to start at I4.0, then the input points can be accessed as I4.0:P and I4.1:P or as IB4:P. Accesses to I4.2:P through I4.7:P are not rejected, but make no sense since these points are not used. Accesses to IW4:P and ID4:P are prohibited since they exceed the byte offset associated with the SB.

Accesses using I_:P do not affect the corresponding value stored in the input process image.

Table 5- 20 Absolute addressing for I memory (immediate)

Bit	I[byte address].[bit address]:P	I0.1:P
Byte, Word, or Double word	I[size][starting byte address]:P	IB4:P, IW5:P, or ID12:P

Q (process image output): The CPU copies the values stored in the output process image to the physical output points. You can access the output process image in bits, bytes, words, or double words. Both read and write access is permitted for process image outputs.

Table 5- 21 Absolute addressing for Q memory

Bit	Q[byte address].[bit address]	Q1.1
Byte, Word, or Double word	Q[size][starting byte address]	QB5, QW10, QD40

By appending a ":P" to the address, you can immediately write to the physical digital and analog outputs of the CPU, SB, SM or distributed module. The difference between an access using Q_:P instead of Q is that the data goes directly to the points being accessed in addition to the output process image (writes to both places). This Q_:P access is sometimes referred to as an "immediate write" access because the data is sent immediately to the target point; the target point does not have to wait for the next update from the output process image.

Because the physical output points directly control field devices that are connected to these points, reading from these points is prohibited. That is, Q_:P accesses are write-only, as opposed to Q accesses which can be read or write.

Q_:P accesses are also restricted to the size of outputs supported by a single CPU, SB, or SM, rounded up to the nearest byte. For example, if the outputs of a 2 DI / 2 DQ SB are configured to start at Q4.0, then the output points can be accessed as Q4.0:P and Q4.1:P or as QB4:P. Accesses to Q4.2:P through Q4.7:P are not rejected, but make no sense since these points are not used. Accesses to QW4:P and QD4:P are prohibited since they exceed the byte offset associated with the SB.

Accesses using Q_:P affect both the physical output as well as the corresponding value stored in the output process image.

Table 5- 22 Absolute addressing for Q memory (immediate)

Bit	Q[byte address].[bit address]:P	Q1.1:P
Byte, Word, or Double word	Q[size][starting byte address]:P	QB5:P, QW10:P or QD40:P

M (bit memory area): Use the bit memory area (M memory) for both control relays and data to store the intermediate status of an operation or other control information. You can access the bit memory area in bits, bytes, words, or double words. Both read and write access is permitted for M memory.

Table 5- 23 Absolute addressing for M memory

Bit	M[byte address].[bit address]	M26.7
Byte, Word, or Double Word	M[size][starting byte address]	MB20, MW30, MD50

Temp (temporary memory): The CPU allocates the temp memory on an as-needed basis. The CPU allocates the temp memory for the code block and initializes the memory locations to 0 at the time when it starts the code block (for an OB) or calls the code block (for an FC or FB).

Temp memory is similar to M memory with one major exception: M memory has a "global" scope, and temp memory has a "local" scope:

- M memory: Any OB, FC, or FB can access the data in M memory, meaning that the data is available globally for all of the elements of the user program.
- Temp memory: The CPU restricts access to the data in temp memory to the OB, FC, or FB that created or declared the temp memory location. Temp memory locations remain local and different code blocks do not share temp memory, even when the code block calls another code block. For example: When an OB calls an FC, the FC cannot access the temp memory of the OB that called it.

The CPU provides temp (local) memory for each OB priority level:

- 16 Kbytes for startup and program cycle, including associated FBs and FCs
- 6 Kbytes for each additional interrupt event thread, including associated FBs and FCs

You access temp memory by symbolic addressing only.

DB (data block): Use the DB memory for storing various types of data, including intermediate status of an operation or other control information parameters for FBs, and data structures required for many instructions such as timers and counters. You can access data block memory in bits, bytes, words, or double words. Both read and write access is permitted for read/write data blocks. Only read access is permitted for read-only data blocks.

Table 5- 24 Absolute addressing for DB memory

Bit	DB[data block number].DBX[byte address].[bit address]	DB1.DBX2.3
Byte, Word, or Double Word	DB[data block number].DB [size][starting byte address]	DB1.DBB4, DB10.DBW2, DB20.DBD8

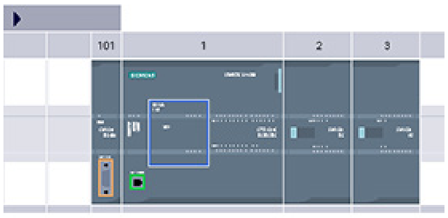
Note

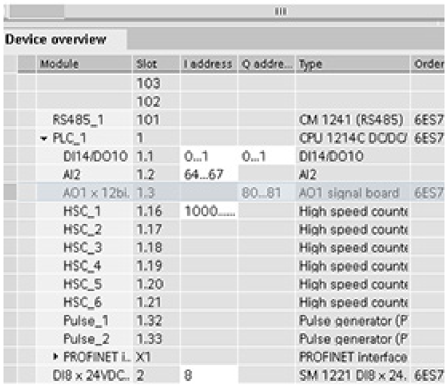
When you specify an absolute address in LAD or FBD, STEP 7 precedes this address with a "%" character to indicate that it is an absolute address. While programming, you can enter an absolute address either with or without the "%" character (for example %I0.0 or I.0). If omitted, STEP 7 supplies the "%" character.

In SCL, you must enter the "%" before the address to indicate that it is an absolute address. Without the "%", STEP 7 generates an undefined tag error at compile time

---

Configuring the I/O in the CPU and I/O modules




Device overview

Module	Slot	I address	Q addre...	Type	Order
	103				
	102				
RS485_1	101			CM 1241 (RS485)	6ES7
▾ PLC_1	1			CPU 1214C DC/DC/	6ES7
DI14/DO10	1.1	0...1	0...1	DI14/DO10	
AI2	1.2	64...67		AI2	
AO1 x 12bi.	1.3		80...81	AO1 signal board	6ES7
HSC_1	1.16	1000.....		High speed counte	
HSC_2	1.17			High speed counte	
HSC_3	1.18			High speed counte	
HSC_4	1.19			High speed counte	
HSC_5	1.20			High speed counte	
HSC_6	1.21			High speed counte	
Pulse_1	1.32			Pulse generator (P	
Pulse_2	1.33			Pulse generator (P	
▸ PROFINET i..	X1			PROFINET interface	
DI8 x 24VDC..	2	8		SM 1221 DI8 x 24.	6ES7

When you add a CPU and I/O modules to your device configuration, STEP 7 automatically assigns I and Q addresses. You can change the default addressing by selecting the address field in the device configuration and entering new numbers.

- STEP 7 assigns digital inputs and outputs in groups of 8 points (1 byte), whether the module uses all the points or not.
- STEP 7 allocates analog inputs and outputs in groups of 2, where each analog poing occupies 2 bytes (16 bits).

The figure shows an example of a CPU 1214C with two SMs and one SB. In this example, you could change the address of the DI8 module to 2 instead of 8. The tool assists you by changing address ranges that are the wrong size or conflict with other addresses.

5.3 Processing of analog values

Analog signal modules provide input signals or expect output values that represent either a voltage range or a current range. These ranges are ±10 V, ±5 V, ±2.5 V, or 0 - 20 mA. The values returned by the modules are integer values where 0 to 27648 represents the rated range for current, and -27648 to 27648 for voltage. Anything outside the range represents either an overflow or underflow. See the tables for analog input representation (Page 1194) and analog output representation (Page 1195) for details about the types of out-of-range values.

In your control program, you probably need to use these values in engineering units, for example to represent a volume, temperature, weight or other quantitative value. To do this for an analog input, you must first normalize the analog value to a real (floating point) value from 0.0 to 1.0. Then you must scale it to the minimum and maximum values of the engineering units that it represents. For values that are in engineering units that you need to convert to an analog output value, you first normalize the value in engineering units to a value between 0.0 and 1.0, and then scale it between 0 and 27648 or -27648 to 27648, depending on the range of the analog module. STEP 7 provides the NORM_X and SCALE_X instructions (Page 275) for this purpose. You can also use the CALCULATE instruction (Page 237) to scale the analog values (Page 39).

Example: analog value processing

Consider, for example, an analog input that has a current range of 0 - 20 mA. The analog input module returns values in the range 0 to 24768 for measured values. For this example, consider that you are using this analog input value to measure a temperature range from 50 °C to 100 °C. A few sample values would have the following meanings:

Analog input value	Engineering units
0	50 °C
6192	62.5 °C
12384	75 °C
18576	87.5 °C
24768	100 °C

The calculation for determining engineering units from the analog input value in this example is as follows:

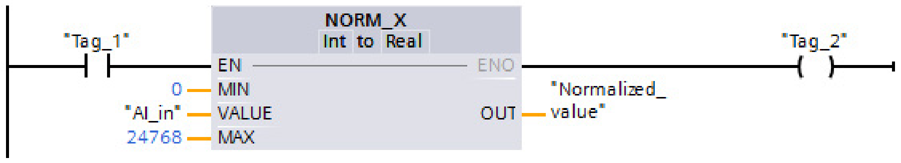
Engineering units value = 50 + (Analog input value) * (100 - 50) / (24768 - 0)

For the general case, the equation would be:

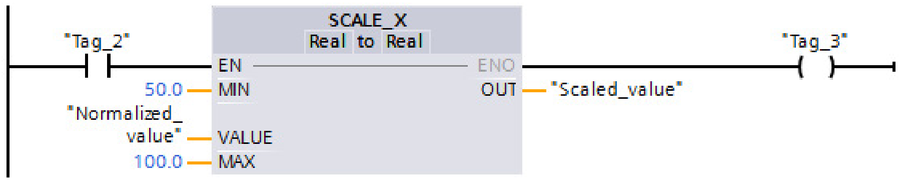
Englineering units value = (Low range of engineering units) +
(Analog input value) *
(High range of engineering units - Low range of engineering units) /
(Maximum analog input range - Minimum analog input range)

In PLC applications, the typical method is to normalize the analog input value to a floating point value between 0.0 and 1.0. Then, you would scale the resulting value to a floating point value in the range of your engineering units. For simplicity, the following LAD instructions use constant values for the ranges; you might actually choose to use tags.

Network 1



Network 2



5.4 Data types

Data types are used to specify both the size of a data element as well as how the data are to be interpreted. Each instruction parameter supports at least one data type, and some parameters support multiple data types. Hold the cursor over the parameter field of an instruction to see which data types are supported for a given parameter.

A formal parameter is the identifier on an instruction that marks the location of data to be used by that instruction (example: the IN1 input of an ADD instruction). An actual parameter is the memory location (preceded by a "%" character) or constant containing the data to be used by the instruction (example %MD400 "Number_of_Widgets"). The data type of the actual parameter specified by you must match one of the supported data types of the formal parameter specified by the instruction.

When specifying an actual parameter, you must specify either a tag (symbol) or an absolute (direct) memory address. Tags associate a symbolic name (tag name) with a data type, memory area, memory offset, and comment, and can be created either in the PLC tags editor or in the Interface editor for a block (OB, FC, FB and DB). If you enter an absolute address that has no associated tag, you must use an appropriate size that matches a supported data type, and a default tag will be created upon entry.

All data types except String, Struct, Array, and DTL are available in the PLC tags editor and the block Interface editors. String, Struct, Array, and DTL are available only in the block Interface editors. You can also enter a constant value for many of the input parameters.

- Bit and Bit sequences (Page 118): Bool (Boolean or bit value), Byte (8-bit byte value), Word (16-bit value), DWord (32-bit double-word value)
- Integer (Page 119)
  - USInt (unsigned 8-bit integer), SInt (signed 8-bit integer),
  - UInt (unsigned 16-bit integer), Int (signed 16-bit integer)
  - UDInt (unsigned 32-bit integer), DInt (signed 32-bit integer)
- Floating-point Real (Page 119): Real (32-bit Real or floating-point value), LReal (64-bit Real or floating-point value)
- Time and Date (Page 120): Time (32-bit IEC time value), Date (16-bit date value), TOD (32-bit time-of-day value), DTL (12-byte date-and-time structure)
- Character and String (Page 122): Char (8-bit single character), String (variable-length string of up to 254 characters)
- Array (Page 124)
- Data structure (Page 125): Struct
- PLC Data type (Page 125)
- Pointers (Page 126): Pointer, Any, Variant

Although not available as data types, the following BCD numeric format is supported by the conversion instructions.

Table 5- 25 Size and range of the BCD format

Format	Size (bits)	Numeric Range	Constant Entry Examples
BCD16	16	-999 to 999	123, -123
BCD32	32	-9999999 to 9999999	1234567, -1234567

5.4.1 Bool, Byte, Word, and DWord data types

Table 5- 26 Bit and bit sequence data types

Data type	Bit size	Number type	Number range	Constant examples	Address examples
Bool	1	Boolean	FALSE or TRUE	TRUE, 1,	I1.0 Q0.1 M50.7 DB1.DBX2.3 Tag_name
		Binary	0 or 1	0, 2#0	
		Octal	8#0 or 8#1	8#1	
		Hexadecimal	16#0 or 16#1	16#1	
Byte	8	Binary	2#0 to 2#11111111	2#00001111	IB2 MB10 DB1.DBB4 Tag_name
		Unsigned integer	0 to 255	15	
		Octal	8#0 to 8#377	8#17	
		Hexadecimal	B#16#0 to B#16#FF	B#16#F, 16#F	
Word	16	Binary	2#0 to 2#1111111111111111	2#1111000011110000	MW10 DB1.DBW2 Tag_name
		Unsigned integer	0 to 65535	61680	
		Octal	8#0 to 8#177777	8#170360	
		Hexadecimal	W#16#0 to W#16#FFFF, 16#0 to 16#FFFF	W#16#F0F0, 16#F0F0	
DWord	32	Binary	2#0 to 2#11111111111111111111111111111111	2#111100001111111100001111	MD10 DB1.DBD8 Tag_name
		Unsigned integer	0 to 4294967295	15793935	
		Octal	8#0 to 8#37777777777	8#74177417	
		Hexadecimal	DW#16#0000_0000 to DW#16#FFFF_FFFF, 16#0000_0000 to 16#FFFF_FFFF	DW#16#F0FF0F, 16#F0FF0F	

5.4.2 Integer data types

Table 5- 27 Integer data types (U = unsigned, S = short, D= double)

Data type	Bit size	Number Range	Constant examples	Address examples
USInt	8	0 to 255	78, 2#01001110	MB0, DB1.DBB4, Tag_name
SInt	8	-128 to 127	+50, 16#50	
UInt	16	0 to 65,535	65295, 0	MW2, DB1.DBW2, Tag_name
Int	16	-32,768 to 32,767	30000, +30000	
UDInt	32	0 to 4,294,967,295	4042322160	MD6, DB1.DBD8, Tag_name
DInt	32	-2,147,483,648 to 2,147,483,647	-2131754992	

5.4.3 Floating-point real data types

Real (or floating-point) numbers are represented as 32-bit single-precision numbers (Real), or 64-bit double-precision numbers (LReal) as described in the ANSI/IEEE 754-1985 standard. Single-precision floating-point numbers are accurate up to 6 significant digits and double-precision floating point numbers are accurate up to 15 significant digits. You can specify a maximum of 6 significant digits (Real) or 15 (LReal) when entering a floating-point constant to maintain precision.

Table 5- 28 Floating-point real data types (L=Long)

Data type	Bit size	Number range	Constant Examples	Address examples
Real	32	-3.402823e+38 to -1.175 495e-38, ±0, +1.175 495e-38 to +3.402823e+38	123.456, -3.4, 1.0e-5	MD100, DB1.DBD8, Tag_name
LReal	64	-1.7976931348623158e+308 to -2.2250738585072014e-308, ±0, +2.2250738585072014e-308 to +1.7976931348623158e+308	12345.123456789e40, 1.2E+40	DB_name.var_name Rules: • No direct addressing support • Can be assigned in an OB, FB, or FC block interface table

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 (Real), or 15 (LReal). For example (Real): 100 000 000 + 1 = 100 000 000.

5.4.4 Time and Date data types

Table 5- 29 Time and date data types

Data type	Size	Range	Constant Entry Examples
Time	32 bits	T#-24d_20h_31m_23s_648ms to T#24d_20h_31m_23s_647ms Stored as: -2,147,483,648 ms to +2,147,483,647 ms	T#5m_30s T#1d_2h_15m_30s_45ms TIME#10d20h30m20s630ms 500h10000ms 10d20h30m20s630ms
Date	16 bits	D#1990-1-1 to D#2168-12-31	D#2009-12-31 DATE#2009-12-31 2009-12-31
Time_of_Day	32 bits	TOD#0:0:0.0 to TOD#23:59:59.999	TOD#10:20:30.400 TIME_OF_DAY#10:20:30.400 23:10:1
DTL (Date and Time Long)	12 bytes	Min.: DTL#1970-01-01-00:00:00.0 Max.: DTL#2262-04-11:23:47:16.854 775 807	DTL#2008-12-16-20:30:20.250

Time

TIME data is stored as a signed double integer interpreted as milliseconds. The editor format can use information for day (d), hours (h), minutes (m), seconds (s) and milliseconds (ms).

It is not necessary to specify all units of time. For example T#5h10s and 500h are valid.

The combined value of all specified unit values cannot exceed the upper or lower limits in milliseconds for the Time data type (-2,147,483,648 ms to +2,147,483,647 ms).

Date

DATE data is stored as an unsigned integer value which is interpreted as the number of days added to the base date 01/01/1990, to obtain the specified date. The editor format must specify a year, month and day.

TOD

TOD (TIME_OF_DAY) data is stored as an unsigned double integer which is interpreted as the number of milliseconds since midnight for the specified time of day (Midnight = 0 ms). The hour (24hr/day), minute, and second must be specified. The fractional second specification is optional.

DTL

DTL (Date and Time Long) data type uses a12 byte structure that saves information on date and time. You can define DTL data in either the Temp memory of a block or in a DB. A value for all components must be entered in the "Start value" column of the DB editor.

Table 5- 30 Size and range for DTL

Length (bytes)	Format	Value range	Example of value input
12	Clock and calendar Year-Month-Day:Hour:Minute: Second.Nanoseconds	Min.: DTL#1970-01-01-00:00:00.0 Max.: DTL#2554-12-31-23:59:59.999 999 999	DTL#2008-12-16-20:30:20.250

Each component of the DTL contains a different data type and range of values. The data type of a specified value must match the data type of the corresponding components.

Table 5- 31 Elements of the DTL structure

Byte	Component	Data type	Value range
0	Year	UINT	1970 to 2554
1			
2	Month	USINT	1 to 12
3	Day	USINT	1 to 31
4	Weekday [1]	USINT	1(Sunday) to 7(Saturday) [1]
5	Hour	USINT	0 to 23
6	Minute	USINT	0 to 59
7	Second	USINT	0 to 59
8	Nanoseconds	UDINT	0 to 999 999 999
9			
10			
11			

[1] The format Year-Month-Day:Hour:Minute: Second.Nanosecond does not include the weekday.

5.4.5 Character and String data types

Table 5- 32 Character and String data types

Data type	Size	Range	Constant Entry Examples
Char	8 bits	16#00 to 16#FF	'A', 't', '@', 'ä', '∑'
WChar	16 bits	16#0000 to 16#FFFF	'A', 't', '@', 'ä', '∑', Asian characters, Cyrillic characters, and others
String	n+ 2 bytes	n = (0 to 254 bytes)	"ABC"
WString	n+ 2 words	n = (0 to 65534 words)	"ä123@XYZ.COM"

Char and WChar

A Char occupies one byte in memory and stores a single character coded in ASCII format, including the extended ASCII character codes. A WChar occupies one word in memory and can contain any double-byte character representation.

The editor syntax uses a single quote character before and after the character. You can use visible characters and control characters.

String and WString

The CPU supports the String data type for storing a sequence of single-byte characters. The String data type contains a total character count (number of characters in the string) and the current character count. The String type provides up to 256 bytes for storing the maximum total character count (1 byte), the current character count (1 byte), and up to 254 bytes in the string. Each byte in a String data type can be any value from 16#00 - 16#FF.

The WString data type provides for longer strings of one-word (double-byte) values. The first word contains the maximum total character count; the next word contains the total character count, and the following string can contain up to 65534 words. Each word in a WString data type can be any value from 16#0000 - 16#FFFF

You can use literal strings (constants) for instruction parameters of type IN using single quotes. For example, ‘ABC’ is a three-character string that could be used as input for parameter IN of the S_CONV instruction. You can also create string variables by selecting data type "String" or "WString" in the block interface editors for OB, FC, FB, and DB. You cannot create a string in the PLC tags editor.

You can specify the maximum string size in bytes (String) or words (WString) by entering square brackets after the keyword "String" or "WString" after you select one of those data types from the data type drop-down list. For example, "MyString String[10]" would specify a 10-byte maximum size for MyString. If you do not include the square brackets with a maximum size, then 254 is assumed for a string and 65534 for a WString. "MyWString WString[1000]" would specify a 1000-word WString.

The following example defines a String with maximum character count of 10 and current character count of 3. This means the String currently contains 3 one-byte characters, but could be expanded to contain up to 10 one-byte characters.

Table 5- 33 Example of a String data type

Total Character Count	Current Character Count	Character 1	Character 2	Character 3	...	Character 10
10	3	'C' (16#43)	'A' (16#41)	'T' (16#54)	...	-
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	...	Byte 11

The following example defines a WString with maximum character count of 500 and current character count of 300. This means the String currently contains 300 one-word characters, but could be expanded to contain up to 500 one-word characters.

Table 5- 34 Example of a WString data type

Total Character Count	Current Character Count	Character 1	Characters 2 to 299	Character 300	...	Character 500
500	300	'ä' (16#0084)	ASCII character words	'M' (16#004D)	...	-
Word 0	Word 1	Word 2	Words 3 to 300	Word 301	...	Word 501

ASCII control characters can be used in Char, Wchar, String and WString data. The following table shows examples of control character syntax.

Table 5- 35 Valid ASCII control characters

Control characters	ASCII Hex value (Char)	ASCII Hex value (WChar)	Control function	Examples
$L or $l	16#0A	16#000A	Line feed	'$LText', '$0AText'
$N or $n	16#0A and 16#0D	16#000A and 16#000D	Line break The new line shows two characters in the string.	'$NText', '$0A$0DText'
$P or $p	16#0C	16#000C	Form feed	'$PText', '$0CText'
$R or $r	16#0D	16#000D	Carriage return (CR)	'$RText','$0DText'
$T or $t	16#09	16#0009	Tab	'$TText', '$09Text'
$$	16#24	16#0024	Dollar sign	'100$$', '100$24'
$'	16#27	16#0027	Single quote	'$'Text$'','$27Text$27'

5.4.6 Array data type

Arrays

You can create an array that contains multiple elements of the same data type. Arrays can be created in the block interface editors for OB, FC, FB, and DB. You cannot create an array in the PLC tags editor.

To create an array from the block interface editor, name the array and choose data type "Array [lo .. hi] of type", then edit "lo", "hi", and "type" as follows:

- lo - the starting (lowest) index for your array
- hi - the ending (highest) index for your array
- type - one of the data types, such as BOOL, SINT, UDINT

Table 5- 36 ARRAY data type rules

Data Type	Array syntax		
ARRAY	Name [index1_min..index1_max, index2_min..index2_max] of <data type> • All array elements must be the same data type. • The index can be negative, but the lower limit must be less than or equal to the upper limit. • Arrays can have one to six dimensions. • Multi-dimensional index min..max declarations are separated by comma characters. • Nested arrays, or arrays of arrays, are not allowed. • The memory size of an array = (size of one element * total number of elements in array)		
	Array index	**Valid index data types**	**Array index rules**
	Constant or variable	USInt, SInt, UInt, Int, UDInt, DInt	• Value limits: -32768 to +32767 • Valid: Mixed constants and variables • Valid: Constant expressions • Not valid: Variable expressions

Example: array declarations	ARRAY[1..20] of REAL	One dimension, 20 elements
	ARRAY[-5..5] of INT	One dimension, 11 elements
	ARRAY[1..2, 3..4] of CHAR	Two dimensions, 4 elements
Example: array addresses	ARRAY1[0]	ARRAY1 element 0
	ARRAY2[1,2]	ARRAY2 element [1,2]
	ARRAY3[i,j]	If i =3 and j=4, then ARRAY3 element [3, 4] is addressed

5.4.7 Data structure data type

You can use the data type "Struct" to define a structure of data consisting of other data types. The struct data type can be used to handle a group of related process data as a single data unit. A Struct data type is named and the internal data structure declared in the data block editor or a block interface editor.

Arrays and structures can also be assembled into a larger structure. A structure can be nested up to eight levels deep. For example, you can create a structure of structures that contain arrays.

5.4.8 PLC data type

The PLC data type editor lets you define data structures that you can use multiple times in your program. You create a PLC data type by opening the "PLC data types" branch of the project tree and double-clicking the "Add new data type" item. On the newly created PLC data type item, use two single-clicks to rename the default name and double-click to open the PLC data type editor.

You create a custom PLC data type structure using the same editing methods that are used in the data block editor. Add new rows for any data types that are necessary to create the data structure that you want.

If a new PLC data type is created, then the new PLC type name will appear in the data type selector drop-down lists in the DB editor and code block interface editor.

Potential uses of PLC data types:

- PLC data types can be used directly as a data type in a code block interface or in data blocks.
- PLC data types can be used as a template for the creation of multiple global data blocks that use the same data structure.

For example, a PLC data type could be a recipe for mixing colors. You can then assign this PLC data type to multiple data blocks. Each data block can then have the variables adjusted to create a specific color.

5.4.9 Pointer data types

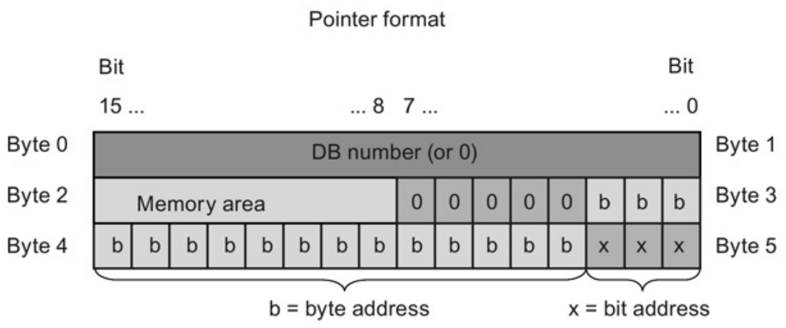
The pointer data types (Pointer, Any, and Variant) can be used in the block interface tables for FB and FC code blocks. You can select a pointer data type from the block interface data type drop-down lists.

The Variant data type is also used for instruction parameters.

5.4.9.1 "Pointer" pointer data type

The data type Pointer points to a particular variable. It occupies 6 bytes (48 bits) in memory and can include the following information:

- DB number or 0 if the data is not stored in a DB
- Storage area in the CPU
- Variable address



Depending on the instruction, you can declare the following three types of pointers:

- Area-internal pointer: contains data on the address of a variable
- Area-crossing pointer: contains data on the memory area and the address of a variable
- DB-pointer: contains a data block number and the address of a variable

Table 5- 37 Pointer types:

Type	Format	Example entry
Area-internal pointer	P#Byte.Bit	P#20.0
Area-crossing pointer	P#Memory_area_Byte.Bit	P#M20.0
DB-pointer	P#Data_block.Data_element	P#DB10.DBX20.0

You can enter a parameter of type Pointer without the prefix (P #). Your entry will be automatically converted to the pointer format.

Table 5- 38 Memory area encoding in the Pointer data:

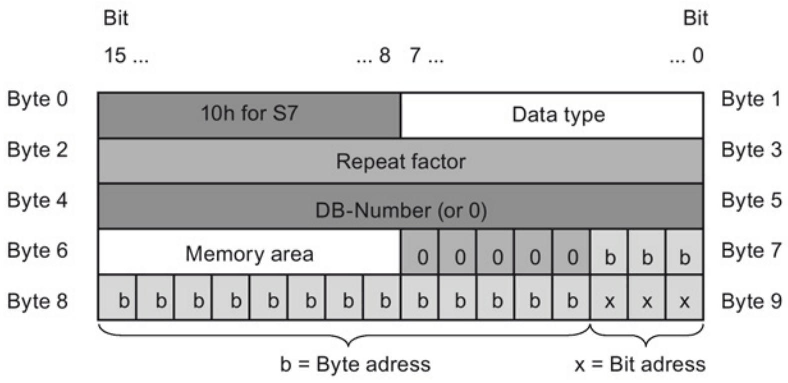
Hexadecimal code	Data type	Description
b#16#81	I	Input memory area
b#16#82	Q	Output memory area
b#16#83	M	Marker memory area
b#16#84	DBX	Data block
b#16#85	DIX	Instance data block
b#16#86	L	Local data
b#16#87	V	Previous local data

5.4.9.2 "Any" pointer data type

The pointer data type ANY ("Any") points to the beginning of a data area and specifies its length. The ANY pointer uses 10 bytes in memory and can include the following information:

- Data type: Data type of the data elements
- Repeat factor: Number of data elements
- DB Number: Data block in which data elements are stored
- Storage area: Memory area of the CPU, in which the data elements are stored
- Start address: "Byte.Bit" starting address of the data

The following image shows the structure of the ANY pointer:



A pointer can not detect ANY structures. It can only be assigned to local variables.

Table 5- 39 Format and examples of the ANY pointer:

Format	Entry example	Description
P#Data_block.Memory_area Data_address Type Number	P#DB 11.DBX 20.0 INT 10	10 words in global DB 11 starting from DBB 20.0
P#Memory_area Data_address Type Number	P#M 20.0 BYTE 10	10 bytes starting from MB 20.0
	P#I 1.0 BOOL 1	Input I1.0

Table 5- 40 Data type encoding in the ANY pointer

Hexadecimal code	Data type	Description
b#16#00	Null	Null pointer
b#16#01	Bool	Bits
b#16#02	Byte	Bytes, 8 Bits
b#16#03	Char	8-bit character
b#16#04	Word	16-bit-word
b#16#05	Int	16-bit-integer
b#16#37	SInt	8-bit-integer
b#16#35	UInt	16-bit unsigned integer
b#16#34	USInt	8-bit unsigned integer
b#16#06	DWord	32-bit double word
b#16#07	DInt	32-bit double integer
b#16#36	UDInt	32-bit-unsigned double integer
b#16#08	Real	32-Bit floating point
b#16#0B	Time	Time
b#16#13	String	Character string

Table 5- 41 Memory area encoding in the ANY pointer:

Hexadecimal code	Memory area	Description
b#16#81	I	Input memory area
b#16#82	Q	Output memory area
b#16#83	M	Marker memory area
b#16#84	DBX	Data block
b#16#85	DIX	Instance data block
b#16#86	L	Local data
b#16#87	V	Previous local data

5.4.9.3 "Variant" pointer data type

The data type Variant can point to variables of different data types or parameters. The Variant pointer can point to structures and individual structural components. The Variant pointer does not occupy any space in memory.

Table 5- 42 Properties of the Variant pointer

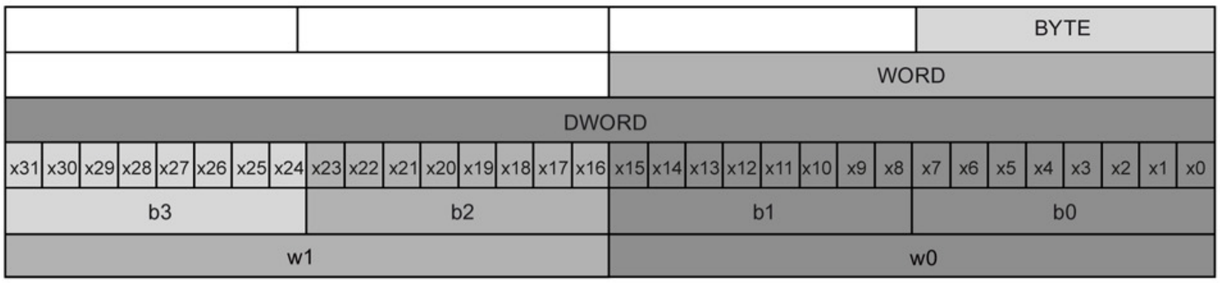
Length (Byte)	Representation	Format	Example entry
0	Symbolic	Operand	MyTag
		DB_name.Struct_name.element_name	MyDB.Struct1.pressure1
	Absolute	Operand	%MW10
		DB_number.Operand Type Length	P#DB10.DBX10.0 INT 12

5.4.10 Accessing a "slice" of a tagged data type

PLC tags and data block tags can be accessed at the bit, byte, or word level depending on their size. The syntax for accessing such a data slice is as follows:

- "<PLC tag name>".xn (bit access)
- "<PLC tag name>".bn (byte access)
- "<PLC tag name>".wn (word access)
- "<Data block name>".<tag name>.xn (bit access)
- "<Data block name>".<tag name>.bn (byte access)
- "<Data block name>".<tag name>.wn (word access)

A double word-sized tag can be accessed by bits 0 - 31, bytes 0 - 3, or word 0 - 1. A word-sized tag can be accessed by bits 0 - 15, bytes 0 - 1, or word 0. A byte-sized tag can be accessed by bits 0 - 7, or byte 0. Bit, byte, and word slices can be used anywhere that bits, bytes, or words are expected operands.

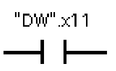
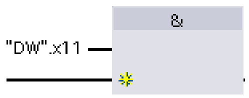
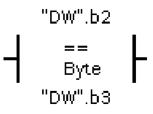
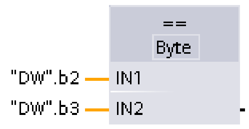
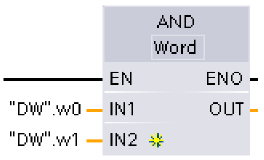
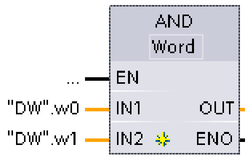

BYTE
WORD
DWORD
x31 x30 x29 x28 x27 x26 x25 x24 x23 x22 x21 x20 x19 x18 x17 x16 x15 x14 x13 x12 x11 x10 x9 x8 x7 x6 x5 x4 x3 x2 x1 x0
b3 b2 b1 b0
w1 w0

Note

Valid data types that can be accessed by slice are Byte, Char, Conn_Any, Date, DInt, DWord, Event_Any, Event_Att, Hw_Any, Hw_Device, HW_Interface, Hw_Io, Hw_Pwm, Hw_SubModule, Int, OB_Any, OB_Att, OB_Cyclic, OB_Delay, OB_WHINT, OB_PCYCLE, OB_STARTUP, OB_TIMEERROR, OB_Tod, Port, Rtm, SInt, Time, Time_Of_Day, UDInt, UInt, USInt, and Word. PLC Tags of type Real can be accessed by slice, but data block tags of type Real cannot.

Examples

In the PLC tag table, "DW" is a declared tag of type DWORD. The examples show bit, byte, and word slice access:

	LAD	FBD	SCL
Bit access			`IF "DW".x11 THEN` `...` `END_IF;`
Byte access			`IF "DW".b2 = "DW".b3 THEN` `...` `END_IF;`
Word access			`out:= "DW".w0 AND "DW".w1;`

5.4.11 Accessing a tag with an AT overlay

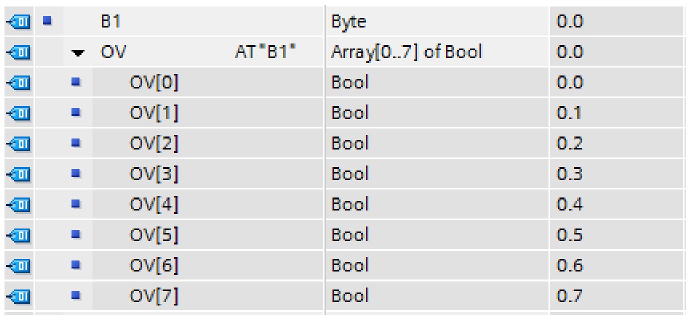
The AT tag overlay allows you to access an already-declared tag of a standard access block with an overlaid declaration of a different data type. You can, for example, address the individual bits of a tag of a Byte, Word, or DWord data type with an Array of Bool.

Declaration

To overlay a parameter, declare an additional parameter directly after the parameter that is to be overlaid and select the data type "AT". The editor creates the overlay, and you can then choose the data type, struct, or array that you wish to use for the overlay.

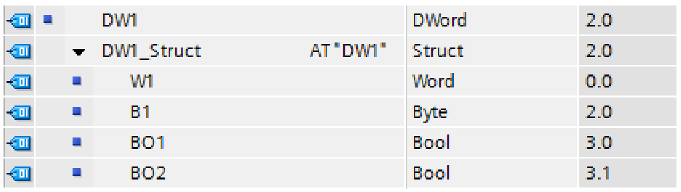
Example

This example shows the input parameters of a standard-access FB. The byte tag B1 is overlaid with an array of Booleans:



B1		Byte	0.0
OV	AT "B1"	Array[0..7] of Bool	0.0
OV[0]		Bool	0.0
OV[1]		Bool	0.1
OV[2]		Bool	0.2
OV[3]		Bool	0.3
OV[4]		Bool	0.4
OV[5]		Bool	0.5
OV[6]		Bool	0.6
OV[7]		Bool	0.7

Another example is a DWord tag overlaid with a Struct, which includes a Word, Byte, and two Booleans:

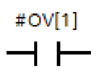
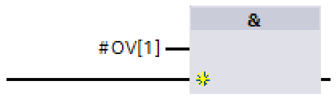


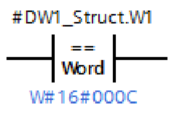
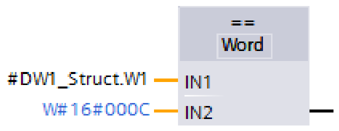
DW1		DWord	2.0
▾ DW1_Struct	AT "DW1"	Struct	2.0
W1		Word	0.0
B1		Byte	2.0
BO1		Bool	3.0
BO2		Bool	3.1

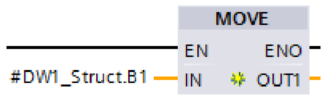
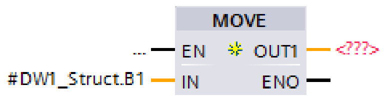
The Offset column of the block interface shows the location of the overlaid data types relative to the original tag.

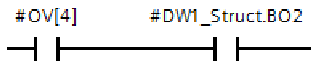
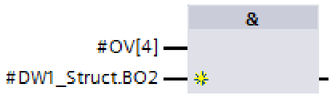
You can addresss the overlay types directly in the program logic:

LAD	FBD	SCL

		`IF #OV[1] THEN`
		`...`
		`END_IF;`

		`IF #DW1_Struct.W1 = W#16#000C THEN`
		`...`
		`END_IF;`

	 MOVE ... EN OUT1 <???> #DW1_Struct.B1 IN ENO	`out1 := #DW1_Struct.B1;`

		`IF #OV[4] AND #DW1_Struct.BO2 THEN`
		`...`
		`END_IF;`

Rules

- Overlaying of tags is only possible in FB and FC blocks with standard (not optimized) access.
- You can overlay parameters for all block types and all declaration sections.
- You can use an overlaid parameter like any other block parameter.
- You cannot overlay parameters of type VARIANT.
- The size of the overlaying parameter must be less than or equal to the size of the overlaid parameter.
- You must declare the overlaying variable immediately after the variable that it overlays and select the keyword "AT" as the initial data type selection.

5.5 Using a memory card

Note

The CPU supports only the pre-formatted SIMATIC memory cards (Page 1266).

Before you copy any program to the formatted memory card, delete any previously saved program from the memory card.

Use the memory card either as a transfer card or as a program card. Any program that you copy to the memory card contains all of the code blocks and data blocks, any technology objects, and the device configuration. A copied program does **not** contain force values.

- Use a transfer card (Page 135) to copy a program to the internal load memory of the CPU without using STEP 7. After you insert the transfer card, the CPU first erases the user program and any force values from the internal load memory, and then copies the program from the transfer card to the internal load memory. When the transfer process is complete, you must remove the transfer card.

  You can use an empty transfer card to access a password-protected CPU when the password has been lost or forgotten (Page 144). Inserting the empty transfer card deletes the password-protected program in the internal load memory of the CPU. You can then download a new program to the CPU.
- Use a program card (Page 138) as external load memory for the CPU. Inserting a program card in the CPU erases all of the CPU internal load memory (the user program and any force values). The CPU then executes the program in external load memory (the program card). Downloading to a CPU that has a program card updates only the external load memory (the program card).

  Because the internal load memory of the CPU was erased when you inserted the program card, the program card **must** remain in the CPU. If you remove the program card, the CPU goes to STOP mode. (The error LED flashes to indicate that program card has been removed.)

The copied program on a memory card includes the code blocks, the data blocks, the technology objects, and the device configuration. The memory card does **not** contain any force values. The force values are not part of the program, but are stored in the load memory, whether the internal load memory of the CPU, or the external load memory (a program card). If a program card is inserted in the CPU, STEP 7 then applies the force values only to the external load memory on the program card.

You also use a memory card when downloading firmware updates (Page 141).

5.5.1 Inserting a memory card in the CPU

NOTICE

Protect memory card and receptacle from electrostatic discharge

Electrostatic discharge can damage the memory card or the receptacle on the CPU.

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

Check that the memory card is not write-protected. Slide the protection switch away from the "Lock" position.

WARNING

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

If you insert a memory card (whether configured as a program card, transfer card, or firmware update card) into a running CPU, the CPU goes immediately to STOP mode, which might cause process disruption that could result in death or severe personal injury.

Before inserting or removing a memory card, always ensure that the CPU is not actively controlling a machine or process. Always install an emergency stop circuit for your application or process.

Note

Do not insert V3.0 program transfer cards into S7-1200 V4.0 CPUs.

Version 3.0 program transfer cards are not compatible with version S7-1200 V4.0 CPUs. Inserting a memory card that contains a V3.0 program causes a CPU error.

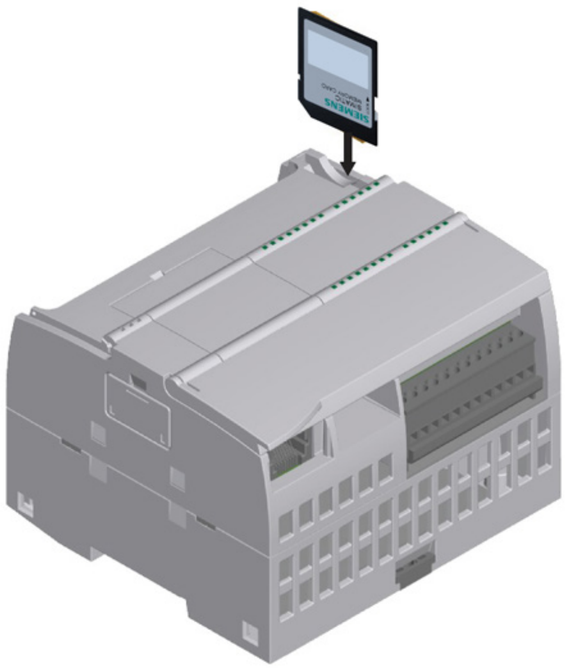
If you do insert an invalid version program transfer card (Page 135), you should remove the card, and perform a STOP to RUN transition, a memory reset (MRES), or cycle power. After you recover the CPU from the error condition, you can download a valid V4.0 CPU program.

To transfer a V3.0 program to a V4.0 program, you must use the TIA Portal to Change Device in the Hardware Configuration.

Note

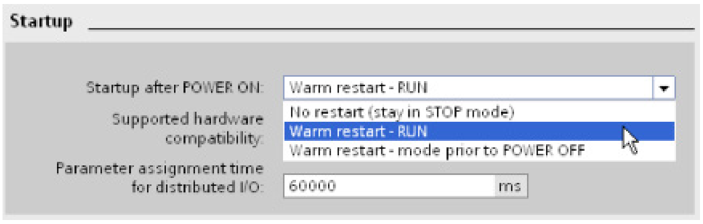
If you insert a memory card with the CPU in STOP mode, the diagnostic buffer displays a message that the memory card evaluation has been initiated. The CPU will evaluate the memory card the next time you either change the CPU to RUN mode, reset the CPU memory with an MRES, or power-cycle the CPU.

Table 5- 43 Inserting a memory card

	To insert a memory card, open the top CPU door and insert the memory card in the slot. A push-push type connector allows for easy insertion and removal. The memory card is keyed for proper installation.

5.5.2 Configuring the startup parameter of the CPU before copying the project to the memory card

When you copy a program to a transfer card or a program card, the program includes the startup parameter for the CPU. Before copying the program to the memory card, always ensure that you have configured the operating mode for the CPU following a power-cycle. Select whether the CPU starts in STOP mode, RUN mode, or in the previous mode (prior to the power cycle).


Startup

Startup after POWER ON: Warm restart - RUN

Supported hardware compatibility:

Parameter assignment time for distributed I/O: 60000 ms

No restart (stay in STOP mode)
Warm restart - RUN
Warm restart - mode prior to POWER OFF

5.5.3 Transfer card

NOTICE
Protect memory card and receptacle from electrostatic discharge
Electrostatic discharge can damage the memory card or the receptacle on the CPU. Make contact with a grounded conductive pad and/or wear a grounded wrist strap whenever you handle the memory card. Store the memory card in a conductive container.

Creating a transfer card

Always remember to configure the startup parameter of the CPU (Page 135) before copying a program to the transfer card. To create a transfer card, follow these steps:

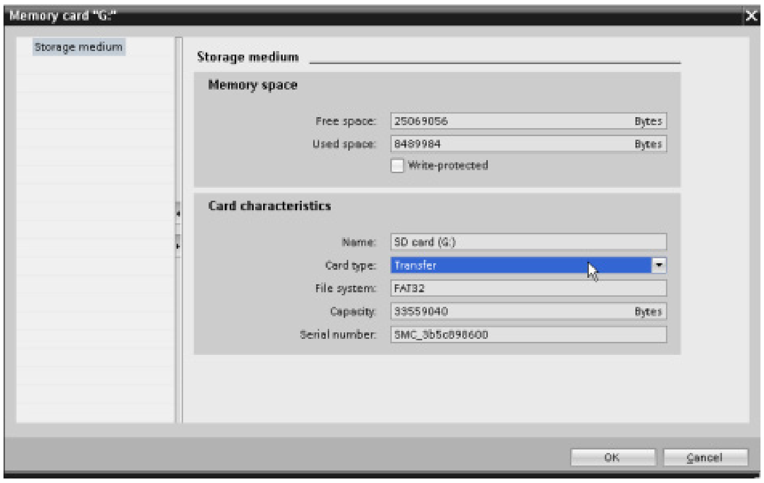
1. Insert a blank SIMATIC memory card that is not write-protected into an SD card reader/writer attached to your computer. (If the card is write-protected, slide the protection switch away from the "Lock" position.)

   If you are reusing a SIMATIC memory card that contains a user program or a firmware update, you **must** delete the program files before reusing the card. Use Windows Explorer to display the contents of the memory card and delete the "S7_JOB.S7S" file and also delete any existing "Data Logs" folders and directory folder (such as "SIMATIC.S7S" or "FWUPDATE.S7S").

NOTICE
Do NOT delete the hidden files "__LOG__" and "crdinfo.bin" from the memory card.
The "__LOG__" and "crdinfo.bin" files are required for the memory card. If you delete these files, you cannot use the memory card with the CPU.

2. In the Project tree (Project view), expand the "SIMATIC Card Reader" folder and select your card reader.
3. Display the "Memory card" dialog by right-clicking the drive letter corresponding to the memory card in the card reader and selecting "Properties" from the context menu.
4. In the "Memory card" dialog, select "Transfer" from the "Card type" drop-down menu.

   At this point, STEP 7 creates the empty transfer card. If you are creating an empty transfer card, such as to recover from a lost CPU password (Page 144), remove the transfer card from the card reader.

   Storage medium

   Memory space

   Free space: 25069056 Bytes
   Used space: 8489984 Bytes
   Write-protected: ☐

   Card characteristics

   Name: SD card (G:)
   Card type: Transfer
   File system: FAT32
   Capacity: 33559040 Bytes
   Serial number: SMC_3b5c898600

5. Add the program by selecting the CPU device (such as PLC_1 [CPU 1214C DC/DC/DC]) in the Project tree and dragging the CPU device to the memory card. (Another method is to copy the CPU device and paste it to the memory card.) Copying the CPU device to the memory card opens the "Load preview" dialog.

6. In the "Load preview" dialog, click the "Load" button to copy the CPU device to the memory card.
7. When the dialog displays a message that the CPU device (program) has been loaded without errors, click the "Finish" button.

Using a transfer card

⚠ WARNING

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

Inserting a 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 a transfer card, always ensure that the CPU is in STOP mode and your process is in a safe state.

Note

Do not insert V3.0 program transfer cards into S7-1200 V4.0 CPUs.

Version 3.0 program transfer cards are not compatible with version S7-1200 V4.0 CPUs. Inserting a memory card that contains a V3.0 program causes a CPU error.

If you do insert an invalid version program transfer card, then remove the card, perform a STOP to RUN transition, a memory reset (MRES), or cycle power. After you recover the CPU from the error condition, you can download a valid V4.0 CPU program

---

To transfer the program to a CPU, follow these steps:

1. Insert the transfer card into the CPU (Page 133). If the CPU is in RUN, the CPU will go to STOP mode. The maintenance (MAINT) LED flashes to indicate that the memory card needs to be evaluated.
2. Power-cycle the CPU to evaluate the memory card. Alternative methods for rebooting the CPU are to perform either a STOP-to-RUN transition or a memory reset (MRES) from STEP 7.
3. After the rebooting and evaluating the memory card, the CPU copies the program to the internal load memory of the CPU.

   The RUN/STOP LED alternately flashes green and yellow to indicate that the program is being copied. When the RUN/STOP LED turns on (solid yellow) and the MAINT LED flashes, the copy process has finished. You can then remove the memory card.
4. Reboot the CPU (either by restoring power or by the alternative methods for rebooting) to evaluate the new program that was transferred to internal load memory.

The CPU then goes to the start-up mode (RUN or STOP) that you configured for the project.

Note

You must remove the transfer card before setting the CPU to RUN mode.

5.5.4 Program card

NOTICE

Electrostatic discharge can damage the memory card or the receptacle on the CPU.

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

Check that the memory card is not write-protected. Slide the protection switch away from the "Lock" position.

Before you copy any program elements to the program card, delete any previously saved programs from the memory card.

Creating a program card

When used as a program card, the memory card is the external load memory of the CPU. If you remove the program card, the internal load memory of the CPU is empty.

Note

If you insert a blank memory card into the CPU and perform a memory card evaluation by either power cycling the CPU, performing a STOP to RUN transition, or performing a memory reset (MRES), the program and force values in internal load memory of the CPU are copied to the memory card. (The memory card is now a program card.) After the copy has been completed, the program in internal load memory of the CPU is then erased. The CPU then goes to the configured startup mode (RUN or STOP).

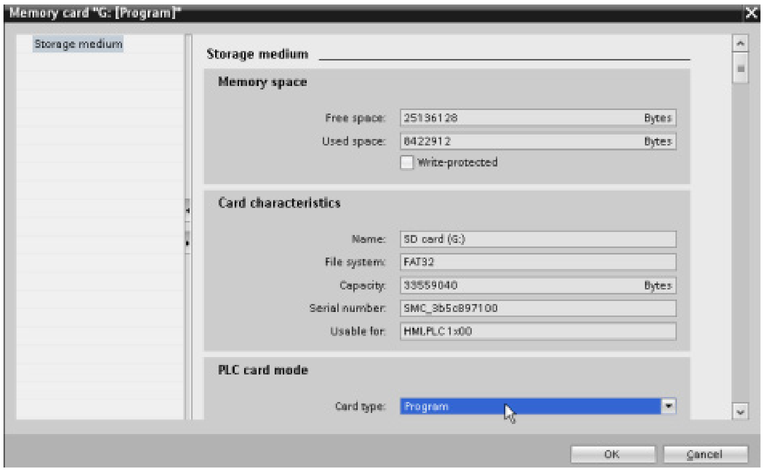
Always remember to configure the startup parameter of the CPU (Page 135) before copying a project to the program card. To create a program card, follow these steps:

1. Insert a blank SIMATIC memory card that is not write-protected into an SD card reader/writer attached to your computer. (If the card is write-protected, slide the protection switch away from the "Lock" position.)

   If you are reusing a SIMATIC memory card that contains a user program or a firmware update, you **must** delete the program files before reusing the card. Use Windows Explorer to display the contents of the memory card and delete the "S7_JOB.S7S" file and also delete any existing "Data Logs" folders and any directory folder (such as "SIMATIC.S7S" or "FWUPDATE.S7S").

NOTICE
Do NOT delete the hidden files "__LOG__" and "crdinfo.bin" from the memory card.
The "__LOG__" and "crdinfo.bin" files are required for the memory card. If you delete these files, you cannot use the memory card with the CPU.

2. In the Project tree (Project view), expand the "SIMATIC Card Reader" folder and select your card reader.
3. Display the "Memory card" dialog by right-clicking the drive letter corresponding to the memory card in the card reader and selecting "Properties" from the context menu.
4. In the "Memory card" dialog, select "Program" from the drop-down menu.



Storage medium

Memory space

Free space: 25136128 Bytes
Used space: 8422912 Bytes
Write-protected

Card characteristics

Name: SD card (G:)
File system: FAT32
Capacity: 33559040 Bytes
Serial number: SMC_3b5c897100
Usable for: HMI,PLC 1x00

PLC card mode

Card type: Program

5. Add the program by selecting the CPU device (such as PLC_1 [CPU 1214C DC/DC/DC]) in the Project tree and dragging the CPU device to the memory card. (Another method is to copy the CPU device and paste it to the memory card.) Copying the CPU device to the memory card opens the "Load preview" dialog.
6. In the "Load preview" dialog, click the "Load" button to copy the CPU device to the memory card.
7. When the dialog displays a message that the CPU device (program) has been loaded without errors, click the "Finish" button.

Using a program card as the load memory for your CPU

WARNING

Risks associated with inserting a program card

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

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

To use a program card with your CPU, follow these steps:

1. Insert the program card into the CPU. If the CPU is in RUN mode, the CPU goes to STOP mode. The maintenance (MAINT) LED flashes to indicate that the memory card needs to be evaluated.
2. Power-cycle the CPU to evaluate the memory card. Alternative methods for rebooting the CPU are to perform either a STOP-to-RUN transition or a memory reset (MRES) from STEP 7.
3. After the CPU reboots and evaluates the program card, the CPU erases the internal load memory of the CPU.

The CPU then goes to the start-up mode (RUN or STOP) that you configured for the CPU.

The program card must remain in the CPU. Removing the program card leaves the CPU with no program in internal load memory.

WARNING

Risks associated with removing a program card

If you remove the program card, the CPU loses its external load memory and generates an error. The CPU goes to STOP mode and flashes the error LED.

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

Do not remove the program card without understanding that you are removing the program from CPU.

5.5.5 Firmware update

You can use a memory card for performing a firmware update. Alternative methods include using the module information page (Page 804) of the Web server to perform a firmware update, or using the online and diagnostic functions of STEP 7 to perform a firmware update (Page 1074). This chapter explains the method that uses a memory card.

NOTICE
Protect memory card and receptacle from electrostatic discharge
Electrostatic discharge can damage the memory card or the receptacle on the CPU.
Make contact with a grounded conductive pad and/or wear a grounded wrist strap whenever you handle the memory card. Store the memory card in a conductive container.

You use a memory card when downloading firmware updates from customer support (http://www.siemens.com/tiaportal). From this Web site, navigate to **Automation Technology > Automation Systems > SIMATIC Industrial Automation Systems > PLC > Modular controllers SIMATIC S7 > SIMATIC S7-1200.** From there continue navigating to the specific type of module that you need to update. Under "Support", click the link for "Software Downloads" to proceed.

Alternatively, you can access the S7-1200 downloads Web page (http://support.automation.siemens.com/WW/view/en/34612486/133100) directly.

Note

You cannot update an S7-1200 CPU V3.0 or earlier to S7-1200 V4.0 or V4.1 by firmware update.

NOTICE
Do not use the Windows formatter utility or any other formatting utility to reformat the memory card.
If a Siemens memory card is reformatted using the Microsoft Windows formatter utility, then the memory card will no longer be usable by a S7-1200 CPU.

To download the firmware update to your memory card, follow these steps:

1. Insert a blank SIMATIC memory card that is not write-protected into an SD card reader/writer attached to your computer. (If the card is write-protected, slide the protection switch away from the "Lock" position.)

   You can reuse a SIMATIC memory card that contains a user program or another firmware update, but you must delete some of the files on the memory card.

   To reuse a memory card, you **must** delete the "S7_JOB.S7S" file and any existing "Data Logs" folders or any folder (such as "SIMATIC.S7S" or "FWUPDATE.S7S") before downloading the firmware update. Use Windows Explorer to display the contents of the memory card and to delete the file and folders.

NOTICE
Do NOT delete the hidden files "__LOG__" and "crdinfo.bin" from the memory card.
The "__LOG__" and "crdinfo.bin" files are required for the memory card. If you delete these files, you cannot use the memory card with the CPU.

2. Select the self-extracting file (.exe) for the firmware update that corresponds to your module, and download it to your computer. Double-click the update file, set the file destination path to be the root directory of the SIMATIC memory card, and start the extraction process. After the extraction is complete, the root directory (folder) of the memory card will contain a "FWUPDATE.S7S" directory and the "S7_JOB.S7S" file.
3. Safely eject the card from the card reader/writer.

To install the firmware update, follow these steps:

⚠WARNING
Verify that the CPU is not actively running a process before installing the firmware update.
Installing the firmware update 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.

1. Insert the memory card into the CPU. If the CPU is in RUN mode, the CPU then goes to STOP mode. The maintenance (MAINT) LED flashes to indicate that the memory card needs to be evaluated.
2. Power-cycle the CPU to start the firmware update. Alternative methods for rebooting the CPU are to perform either a STOP-to-RUN transition or a memory reset (MRES) from STEP 7.

   Note

   To complete the firmware upgrade for the module, you must ensure that the external 24 VDC power to the module remains on.

After the CPU reboots, the firmware update starts. The RUN/STOP LED alternately flashes green and yellow to indicate that the update is being copied. When the RUN/STOP LED turns on (solid yellow) and the MAINT LED flashes, the copy process has finished. You must then remove the memory card.

3. After removing the memory card, reboot the CPU again (either by restoring power or by the alternative methods for rebooting) to load the new firmware.

The user program and hardware configuration are not affected by the firmware update. When the CPU is powered up, the CPU enters the configured start-up state. (If the startup mode for your CPU was configured to "Warm restart - mode before POWER OFF", the CPU will be in STOP mode because the last state of the CPU was STOP.)

Note

Updating multiple modules connected to CPU

If your hardware configuration contains multiple modules that correspond to a single firmware update file on the memory card, the CPU applies the updates to all applicable modules (CM, SM, and SB) in configuration order, that is, by increasing order of the module position in Device Configuration in STEP 7.

If you have downloaded multiple firmware updates to the memory card for multiple modules, the CPU applies the updates in the order in which you downloaded them to the memory card.

5.6 Recovery from a lost password

If you have lost the password for a password-protected CPU, use an empty transfer card to delete the password-protected program. The empty transfer card erases the internal load memory of the CPU. You can then download a new user program from STEP 7 to the CPU.

For information about the creation and use of an empty transfer card, see the section of transfer cards (Page 135).

⚠ WARNING

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

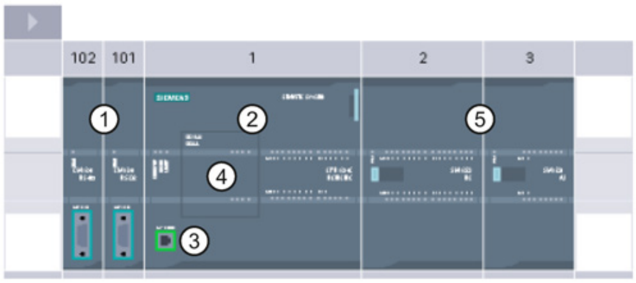
If you insert a transfer card in a running CPU, the CPU goes to STOP. Control devices can fail in an unsafe condition, resulting in unexpected operation of controlled equipment. Such unexpected operations could result in death or serious injury to personnel, and/or damage to equipment.

Before inserting a transfer card, always ensure that the CPU is in STOP mode and your process is in a safe state.

You must remove the transfer card before setting the CPU to RUN mode.

Device configuration 6

You create the device configuration for your PLC by adding a CPU and additional modules to your project.



①	Communication module (CM) or communication processor (CP): Up to 3, inserted in slots 101, 102, and 103
②	CPU: Slot 1
③	PROFINET port of CPU
④	Signal board (SB), communication board (CB) or battery board (BB): up to 1, inserted in the CPU
⑤	Signal module (SM) for digital or analog I/O: up to 8, inserted in slots 2 through 9 (CPU 1214C, CPU 1215C and CPU 1217C allow 8, CPU 1212C allows 2, CPU 1211C does not allow any)

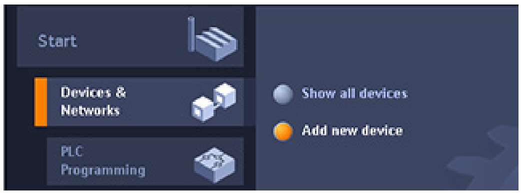
Configuration control

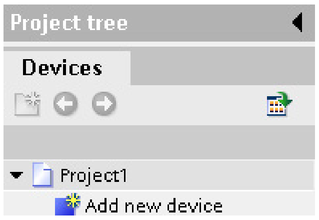
Device configuration for the S7-1200 also supports "configuration control (Page 151)" where you can configure a maximum configuration for a project including modules that you might not actually use. This feature, sometimes also called "option handling", allows you to configure a maximum configuration that you might use with variations in the installed modules in multiple applications.

6.1 Inserting a CPU

You create your device configuration by inserting a CPU into your project.

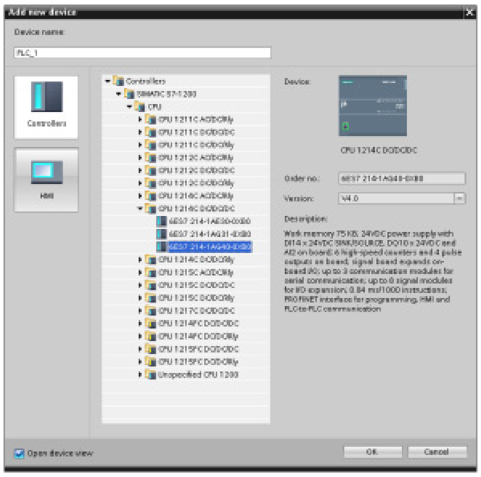
- In the Portal view, select "Devices & Networks" and click "Add new device".
- In the Project view, under the project name, double-click "Add new device".





Be sure you insert the correct model and firmware version from the list. Selecting the CPU from the "Add new device" dialog creates the rack and CPU.

"Add new device" dialog



Device: CPU 1214C DC/DC/DC

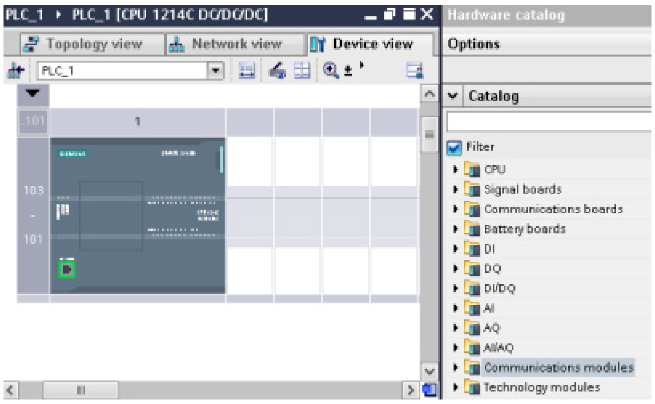
Order no.: 6ES7 214-1AG40-0XB0

Version: V4.0

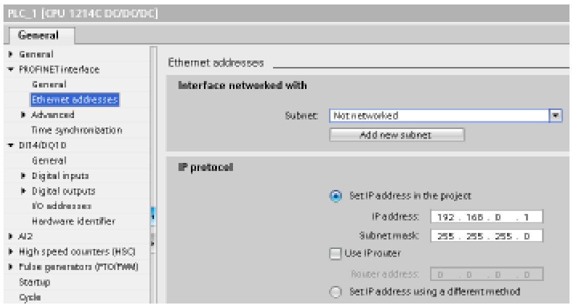
Description:

Work memory 75 KB; 24VDC power supply with DI14 x 24VDC SINK/SOURCE, DQ10 x 24VDC and AI2 on board; 6 high-speed counters and 4 pulse outputs on board; signal board expands on-board I/O; up to 3 communication modules for serial communication; up to 8 signal modules for I/O expansion; 0.04 ms/1000 instructions; PROFINET interface for programming, HMI and PLC-to-PLC communication

Device view of the hardware configuration



Selecting the CPU in the Device view displays the CPU properties in the inspector window.



Ethernet addresses

Interface networked with

Subnet: Not networked

IP protocol

[✓ Set IP address in the project]

IP address: 192 . 168 . 0 . 1

Subnet mask: 255 . 255 . 255 . 0

Use IP router

Router address: 0 . 0 . 0 . 0

Set IP address using a different method

Note

The CPU does not have a pre-configured IP address. You must manually assign an IP address for the CPU during the device configuration. If your CPU is connected to a router on the network, you also enter the IP address for a router.

6.2 Uploading the configuration of a connected CPU

STEP 7 provides two methods for uploading the hardware configuration of a connected CPU:

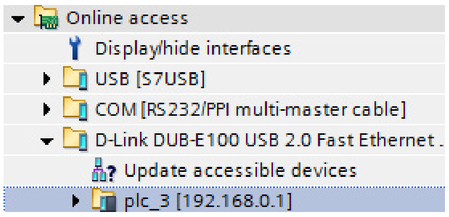
- Uploading the connected device as a new station
- Configuring an unspecified CPU and detecting the hardware configuration of the connected CPU

Note, however, that the first method uploads both the hardware configuration and the software of the connected CPU.

Uploading a device as a new station

To upload a connected device as a new station, follow these steps:

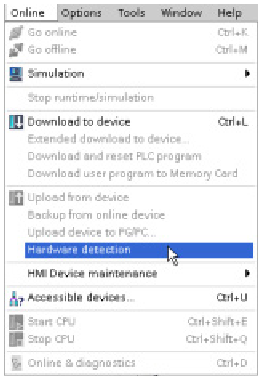
1. Expand your communications interface from the "Online access" node of the project tree.
2. Double-click "Update accessible devices".
3. Select the PLC from the detected devices.


Online access
Display/hide interfaces
USB [S7USB]
COM [RS232/PPI multi-master cable]
D-Link DUB-E100 USB 2.0 Fast Ethernet .
Update accessible devices
plc_3 [192.168.0.1]

4. From the Online menu of STEP 7, select the "Upload device as new station (hardware and software)" menu command.

STEP 7 uploads both the hardware configuration and the program blocks.

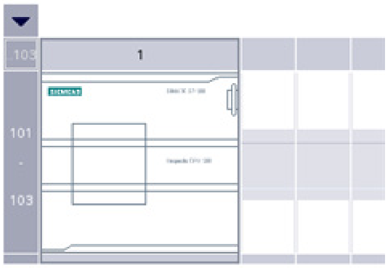
Detecting the hardware configuration of an unspecified CPU

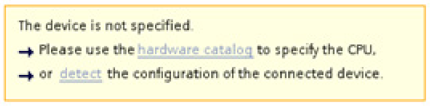

Online
Options
Tools
Window
Help
Go online Ctrl+K
Go offline Ctrl+M
Simulation
Stop runtime/simulation
Download to device Ctrl+L
Extended download to device...
Download and reset PLC program
Download user program to Memory Card
Upload from device
Backup from online device
Upload device to PG/PC...
Hardware detection
HMI Device maintenance
Accessible devices... Ctrl+U
Start CPU Ctrl+Shift+E
Stop CPU Ctrl+Shift+Q
Online & diagnostics Ctrl+D

If you are connected to a CPU, you can upload the configuration of that CPU, including any modules, to your project. Simply create a new project and select the "unspecified CPU" instead of selecting a specific CPU. (You can also skip the device configuration entirely by selecting the "Create a PLC program" from the "First steps". STEP 7 then automatically creates an unspecified CPU.)

From the program editor, you select the "Hardware detection" command from the "Online" menu.

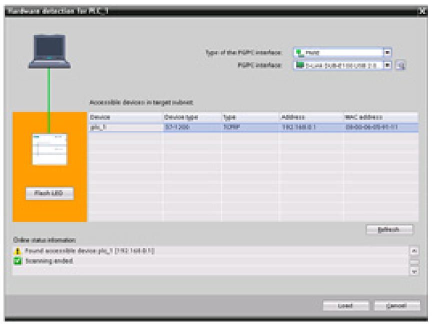
From the device configuration editor, you select the option for detecting the configuration of the connected device.





The device is not specified.
→ Please use the hardware catalog to specify the CPU,
→ or detect the configuration of the connected device.

After you select the CPU from the online dialog and click the Load button, STEP 7 uploads the hardware configuration from the CPU, including any modules (SM, SB, or CM). You can then configure the parameters for the CPU and the modules (Page 162).



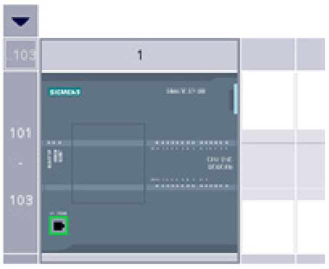
Hardware detection for PLC_1

Type of the PG/PC interface: PN/IE
PG/PC interface: [illegible]

Accessible devices in target subnet:

Device	Device type	Type	Address	MAC address
plc_1	S7-1200	TCP/IP	192.168.0.1	[illegible]

Online status information:
Found accessible device plc_1 [192.168.0.1]
Scanning ended



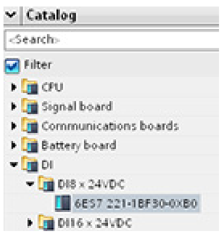
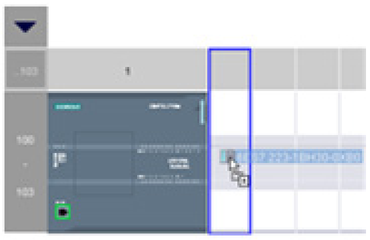
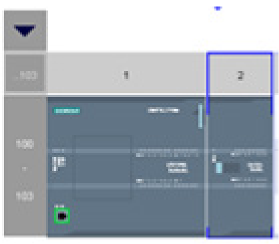
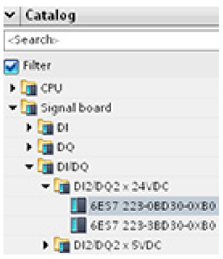
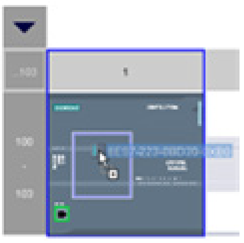
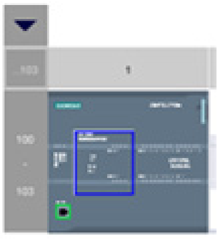
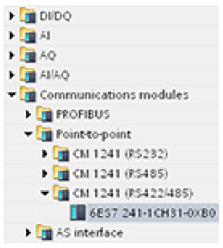
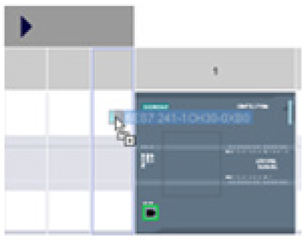
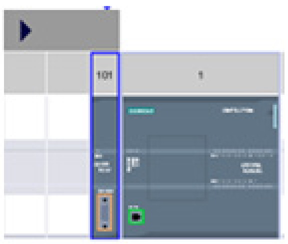
6.3 Adding modules to the configuration

Use the hardware catalog to add modules to the CPU:

- Signal module (SM) provides additional digital or analog I/O points. These modules are connected to the right side of the CPU.
- Signal board (SB) provides just a few additional I/O points for the CPU. The SB is installed on the front of the CPU.
- Battery Board 1297 (BB) provides long-term backup of the realtime clock. The BB is installed on the front of the CPU.
- Communication board (CB) provides an additional communication port (such as RS485). The CB is installed on the front of the CPU.
- Communication module (CM) and communication processor (CP) provide an additional communication port, such as for PROFIBUS or GPRS. These modules are connected to the left side of the CPU.

To insert a module into the device configuration, select the module in the hardware catalog and either double-click or drag the module to the highlighted slot. You must add the modules to the device configuration and download the hardware configuration to the CPU for the modules to be functional.

Table 6- 1 Adding a module to the device configuration

Module	Select the module	Insert the module	Result
SM	 Catalog <Search> Filter CPU Signal board Communications boards Battery board DI DI8 x 24VDC 6ES7 221-1BF30-0XB0 DI16 x 24VDC		
SB, BB or CB	 Catalog <Search> Filter CPU Signal board DI DQ DI/DQ DI2/DQ2 x 24VDC 6ES7 223-0BD30-0XB0 6ES7 223-3BD30-0XB0 DI2/DQ2 x 5VDC		
CM or CP	 DI/DQ AI AQ AI/AQ Communications modules PROFIBUS Point-to-point CM 1241 (RS232) CM 1241 (RS485) CM 1241 (RS422/485) 6ES7 241-1CH31-0XB0 AS interface		

With the "configuration control" feature (Page 151), you can add signal modules and signal boards to your device configuration that might not correspond to the actual hardware for a specific application, but that will be used in related applications that share a common user program, CPU model, and perhaps some of the configured modules.

6.4 Configuration control

6.4.1 Advantages and applications of configuration control

Configuration control can be a useful solution when you create an automation solution (machine) that you intend to use with variations in multiple installations.

You can load a STEP 7 device configuration and user program to different installed PLC configurations. You only need to make a few easy adaptations to make the STEP 7 project correspond to the actual installation.

6.4.2 Configuring the central installation and optional modules

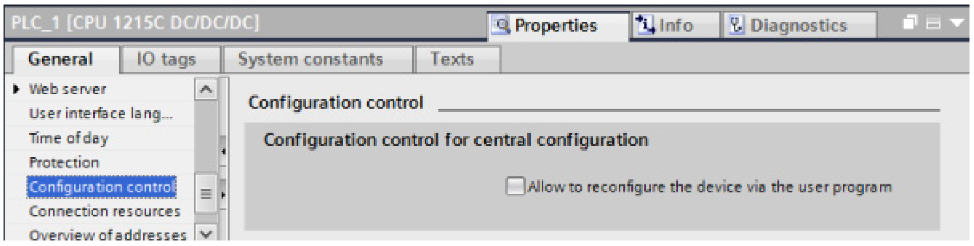
Configuration control with STEP 7 and the S7-1200 enables you to configure a maximum configuration for a standard machine and to operate versions (options) that use a subset of this configuration. The PROFINET with STEP 7 manual (http://support.automation.siemens.com/WW/view/en/49948856) refers to these types of projects as "standard machine projects".

A control data record that you program in the startup program block notifies the CPU as to which modules are missing in the real installation as compared to the configuration or which modules are located in different slots as compared to the configuration. Configuration control does not have an impact on the parameter assignment of the modules.

Configuration control gives you the flexibility to vary the installation as long as you can derive the real configuration from the maximum device configuration in STEP 7.

To activate configuration control and structure the required control data record, follow these steps:

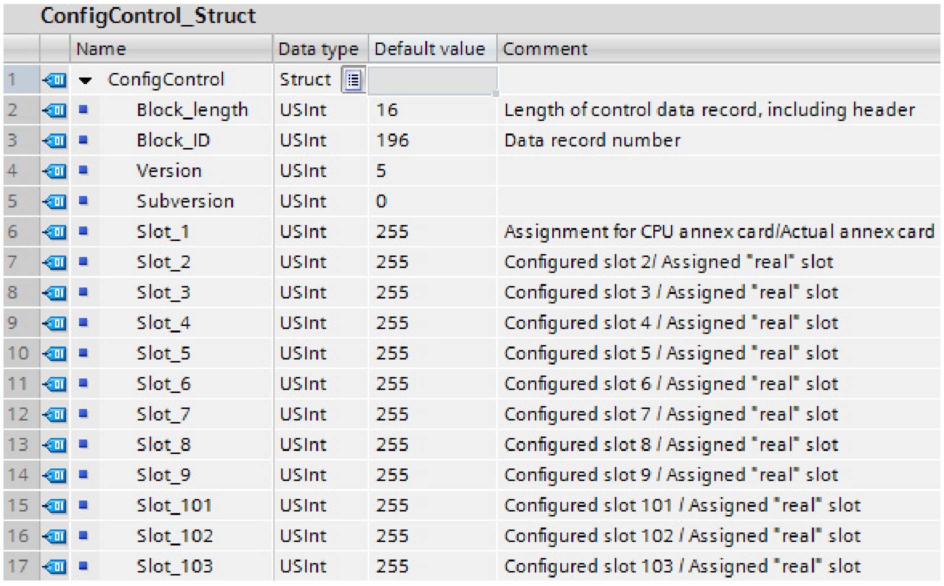
1. Optionally, reset the CPU to factory settings to ensure that an incompatible control data record is not present in the CPU.
2. Select the CPU in device configuration in STEP 7.
3. From the Configuration control node in the CPU properties, select the "Enable reconfiguration of device with user program" check box.



Configuration control for central configuration

Allow to reconfigure the device via the user program

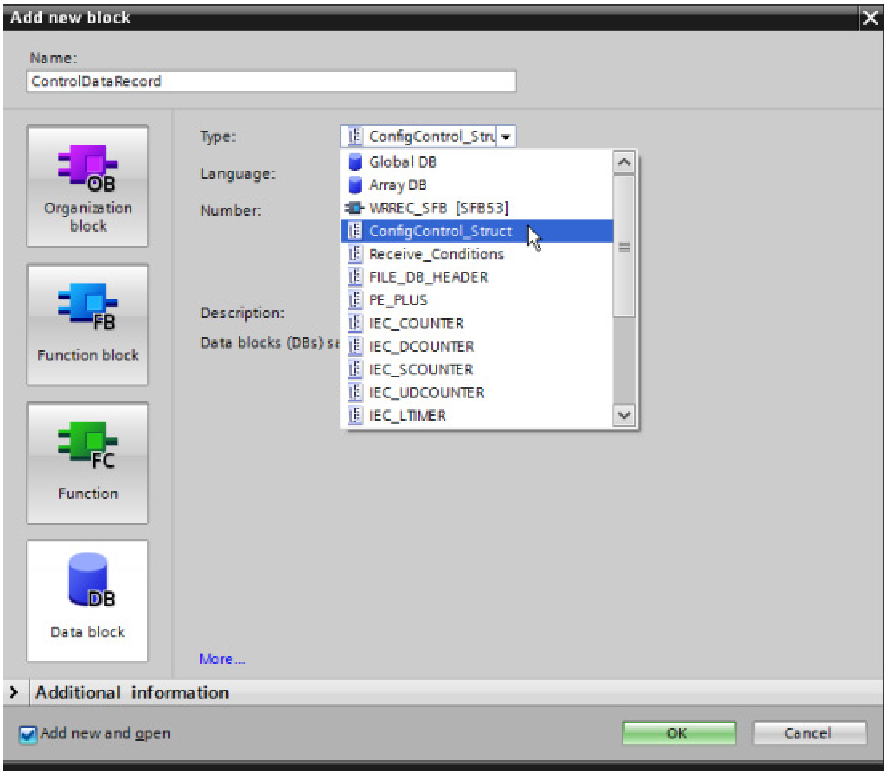
4. Create a PLC data type to contain the control data record. Configure it as a struct that includes four USints for configuration control information and additional USints to correspond to the slots of a maximum S7-1200 device configuration, as follows:



ConfigControl_Struct

	Name	Data type	Default value	Comment
1	ConfigControl	Struct		
2	Block_length	USInt	16	Length of control data record, including header
3	Block_ID	USInt	196	Data record number
4	Version	USInt	5	
5	Subversion	USInt	0	
6	Slot_1	USInt	255	Assignment for CPU annex card/Actual annex card
7	Slot_2	USInt	255	Configured slot 2/ Assigned "real" slot
8	Slot_3	USInt	255	Configured slot 3 / Assigned "real" slot
9	Slot_4	USInt	255	Configured slot 4 / Assigned "real" slot
10	Slot_5	USInt	255	Configured slot 5 / Assigned "real" slot
11	Slot_6	USInt	255	Configured slot 6 / Assigned "real" slot
12	Slot_7	USInt	255	Configured slot 7 / Assigned "real" slot
13	Slot_8	USInt	255	Configured slot 8 / Assigned "real" slot
14	Slot_9	USInt	255	Configured slot 9 / Assigned "real" slot
15	Slot_101	USInt	255	Configured slot 101 / Assigned "real" slot
16	Slot_102	USInt	255	Configured slot 102 / Assigned "real" slot
17	Slot_103	USInt	255	Configured slot 103 / Assigned "real" slot

5. Create a data block of the PLC data type that you created.



Name: ControlDataRecord

Type: ConfigControl_Stru

Language:

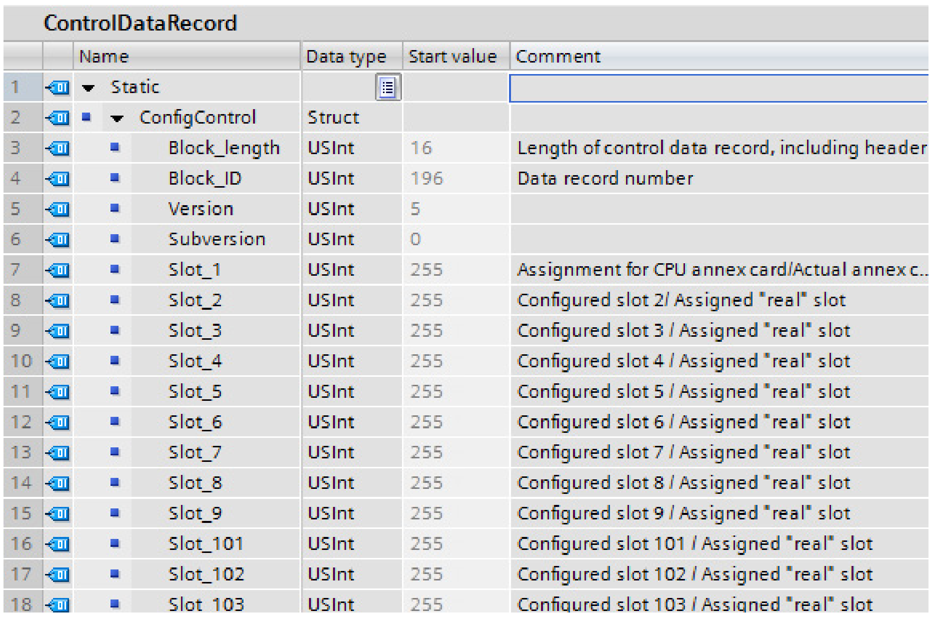
Number:

Description:

Data blocks (DBs) s

☑ Add new and open

6. In this data block, configure the Block_length, Block_ID, Version, and Subversion as shown below. Configure the values for the slots based on their presence or absence and position in your actual installation:
   - 0: Configured module is not present in the actual configuration. (The slot is empty.)
   - 1 to 9, 101 to 103: The actual slot position for the configured slot
   - 255: The STEP 7 device configuration does not include a module in this slot.



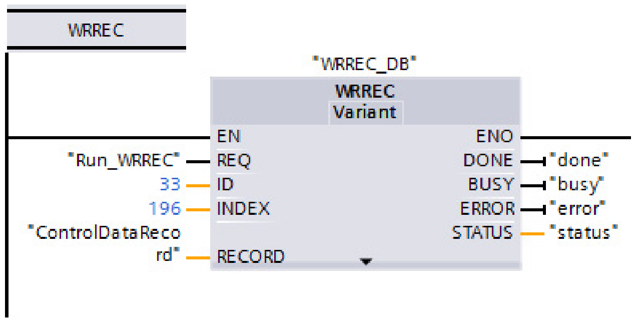
ControlDataRecord

	Name	Data type	Start value	Comment
1	Static			
2	ConfigControl	Struct		
3	Block_length	USInt	16	Length of control data record, including header
4	Block_ID	USInt	196	Data record number
5	Version	USInt	5	
6	Subversion	USInt	0	
7	Slot_1	USInt	255	Assignment for CPU annex card/Actual annex c..
8	Slot_2	USInt	255	Configured slot 2/ Assigned "real" slot
9	Slot_3	USInt	255	Configured slot 3 / Assigned "real" slot
10	Slot_4	USInt	255	Configured slot 4 / Assigned "real" slot
11	Slot_5	USInt	255	Configured slot 5 / Assigned "real" slot
12	Slot_6	USInt	255	Configured slot 6 / Assigned "real" slot
13	Slot_7	USInt	255	Configured slot 7 / Assigned "real" slot
14	Slot_8	USInt	255	Configured slot 8 / Assigned "real" slot
15	Slot_9	USInt	255	Configured slot 9 / Assigned "real" slot
16	Slot_101	USInt	255	Configured slot 101 / Assigned "real" slot
17	Slot_102	USInt	255	Configured slot 102 / Assigned "real" slot
18	Slot_103	USInt	255	Configured slot 103 / Assigned "real" slot

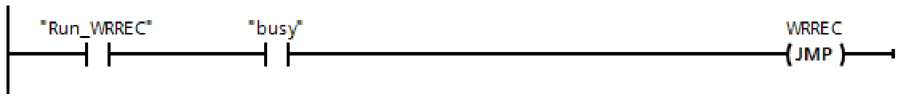
   See Example of configuration control (Page 157) for an explanation of how to assign the slot values.

7. In the startup OB, call the extended WRREC (Write data record) instruction to transfer the control data record that you created to index 196 of hardware ID 33. Use a label and JMP (jump) instruction to wait for the WRREC instruction to complete.

   Network 1:



   Network 2:



Note

Configuration control is not in effect until the WRREC instruction transfers the control data record in the startup OB. If you have enabled configuration control and the CPU does not have the control data record, it will go to STOP mode when it exits STARTUP mode. Be sure that you program the startup OB to transfer the control data record.

Module arrangement

The following table shows the slot number assignment:

Slot	Modules
1	Signal board or communication board (CPU annex card)
2 to 9	Signal modules
101 to 103	Communication modules

Control data record

A control data record 196 contains the slot assignment and represents the actual configuration, as shown below:

Byte	Element	Value	Explanation
0	Block length	16	Header
1	Block ID	196	
2	Version	5	
3	Subversion	0	
4	Assignment of CPU annex card	Actual annex card, 0, or 255*	Control element
5	Assignment of configured slot 2	Actual slot, 0, or 255*	Describes in each element which real slot in the device is assigned to the configured slot.
...	...	...	
12	Assignment of configured slot 9	Actual slot, 0, or 255*	
13	Assignment of configured slot 101	Actual slot or 255*	Unlike signal modules, the actual slot for physically-present communication modules must be the same as the configured slot.
14	Assignment of configured slot 102	Actual slot or 255*	
15	Assignment of configured slot 103	Actual slot or 255*	

***Slot values:**

0: Configured module is not present in the actual configuration. (The slot is empty.)

1 to 9, 101 to 103: The actual slot position for the configured slot

255: The STEP 7 device configuration does not include a module in this slot.

Note

Alternative to creating a PLC tag type

As an alternative to creating a custom PLC tag type, you can create a data block directly with all of the structure elements of a control data record. You could even configure multiple structs in this data block to serve as multiple control data record configurations. Either implementation is an effective way to transfer the control data record during startup.

Rules

Observe the following rules:

- Configuration control does not support position changes for communication modules. The control data record slot positions for slots 101 to 103 must correspond to the actual installation. If you have not configured a module for the slot, enter 255 for that slot position.
- You cannot have embedded empty (unused) slots between filled (used) slots. For example, if the actual configuration has a module in slot 4, then the actual configuration must also have modules in slots 2 and 3. Correspondingly, if the actual configuration has a communication module in slot 102, then the actual configuration must also have a module in slot 101.
- If you have enabled configuration control, the CPU is not ready for operation without a control data record. The CPU returns from startup to STOP if a startup OB does not transfer a valid control data record. The CPU does not initialize the central I/O in this case and enters the cause for the STOP mode in the diagnostics buffer.
- The CPU saves a successfully-transferred control data record in retentive memory, which means that it is not necessary to write the control data record 196 again at a restart if you have not changed the configuration.
- Each real slot must be present only once in the control data record.
- You can only assign a real slot to one configured slot.

Note

Modifying a configuration

The writing of a control data record with a modified configuration triggers the following automatic reaction by the CPU: Memory reset with subsequent startup with this modified configuration.

As a result of this reaction, the CPU deletes the original control data record and saves the new control data record retentively.

Behavior during operation

For the online display and for the display in the diagnostics buffer (module OK or module faulty), STEP 7 uses the device configuration and not the differing real configuration.

Example: A module outputs diagnostics data. This module is configured in slot 4, but is actually inserted in slot 3. The online view indicates that configured slot 4 is faulty. In the real configuration, the module at slot 3 indicates an error by its LED display.

If you have configured modules as missing in the control data record (0 entry), the automation system behaves as follows:

- Modules designated as not present in the control data record do not supply diagnostics and their status is always OK. The value status is OK.
- Direct writing access to the outputs or writing access to the process image of outputs that are not present proceeds with no effect; the CPU reports no access error.
- Direct read access to the inputs or read access to the process image of inputs that are not present results in a value "0" for each input; the CPU reports no access error.
- Writing a data record to a module that is not present proceeds with no effect; the CPU reports no error.
- Attempting to read data record from module that is not present resuls in an error because the CPU cannot return a valid data record.

Error messages

The CPU returns the following error messages if an error occurs during writing of the control data record:

Error code	Meaning
16#80B1	Invalid length; the length information in the control data record is not correct.
16#80B5	Configuration control parameters not assigned
16#80E2	Data record was transferred in the wrong OB context. The data record must be transferred in the startup OB.
16#80B0	Block type (byte 2) of control data record is not equal to 196.
16#80B8	Parameter error; module signals invalid parameters, for example: • The control data record attempts to modify the configuration of a communication module or a communication annex card. The real configuration for communication modules and a communication annex card must equal the STEP 7 configuration. • The assigned value for an unconfigured slot in the STEP 7 project is not equal to 255. • The assigned value for a configured slot is out of range. • The assigned configuration has an "internal" empty slot, for example, slot n is assigned and slot n-1 is not assigned.

6.4.3 Example of configuration control

This example describes a configuration consisting of a CPU and three I/O modules.The module at slot 3 is not present in the first actual installation, so you use configuration control to "hide" it.

In the second installation, the application includes the module that was initially hidden but now includes it in the last slot. A modified control data record provides the information about the slot assignments of the modules.

Example: Actual installation with configured but unused module

The device configuration contains all modules that can be present in an actual installation (maximum configuration). In this case, the module that is in slot 3 in the device configuration is not present in the real installation.

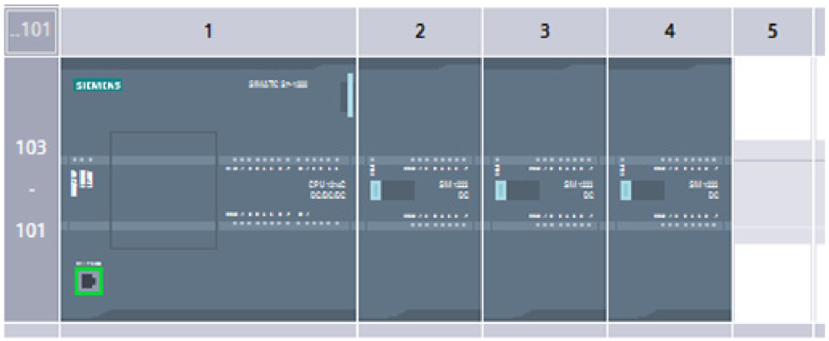


Figure 6-1 Device configuration of maximum installation

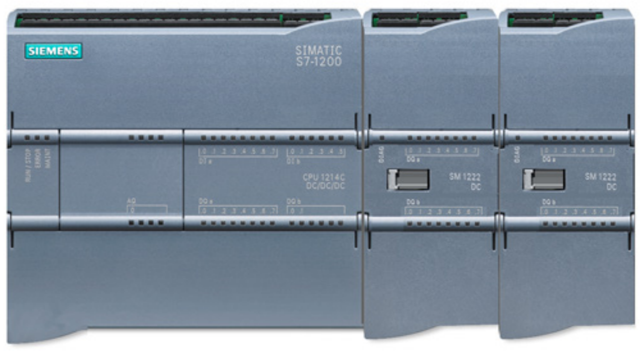
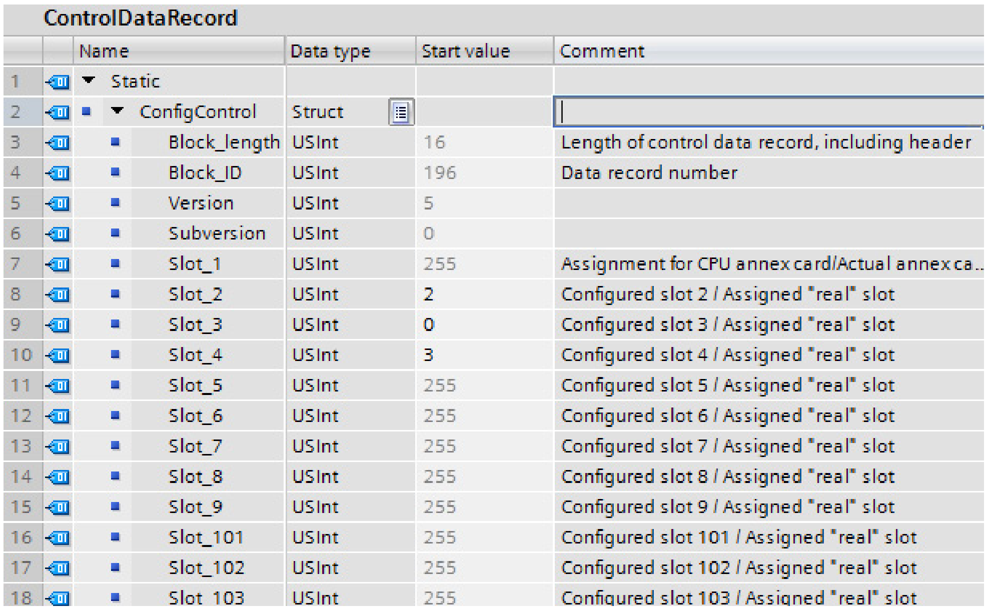


Figure 6-2 Actual installation with module configured in slot 3 absent, and module configured for slot 4 in actual slot 3

To indicate the absence of the missing module, you must configure slot 3 in the control data record with 0.



ControlDataRecord

	Name	Data type	Start value	Comment
1	Static			
2	ConfigControl	Struct		
3	Block_length	USInt	16	Length of control data record, including header
4	Block_ID	USInt	196	Data record number
5	Version	USInt	5	
6	Subversion	USInt	0	
7	Slot_1	USInt	255	Assignment for CPU annex card/Actual annex ca..
8	Slot_2	USInt	2	Configured slot 2 / Assigned "real" slot
9	Slot_3	USInt	0	Configured slot 3 / Assigned "real" slot
10	Slot_4	USInt	3	Configured slot 4 / Assigned "real" slot
11	Slot_5	USInt	255	Configured slot 5 / Assigned "real" slot
12	Slot_6	USInt	255	Configured slot 6 / Assigned "real" slot
13	Slot_7	USInt	255	Configured slot 7 / Assigned "real" slot
14	Slot_8	USInt	255	Configured slot 8 / Assigned "real" slot
15	Slot_9	USInt	255	Configured slot 9 / Assigned "real" slot
16	Slot_101	USInt	255	Configured slot 101 / Assigned "real" slot
17	Slot_102	USInt	255	Configured slot 102 / Assigned "real" slot
18	Slot_103	USInt	255	Configured slot 103 / Assigned "real" slot

Example: Actual installation with module subsequently added to a different slot

In the second example, the module in slot 3 of the device configuration is present in the actual installation but is in slot 4.

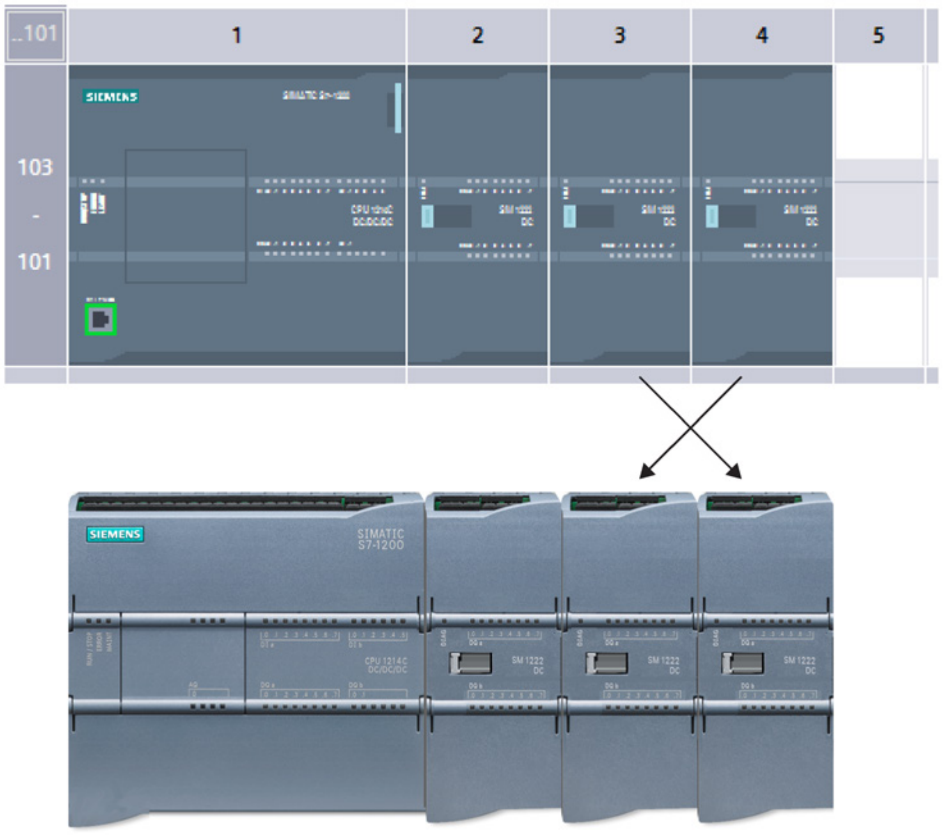
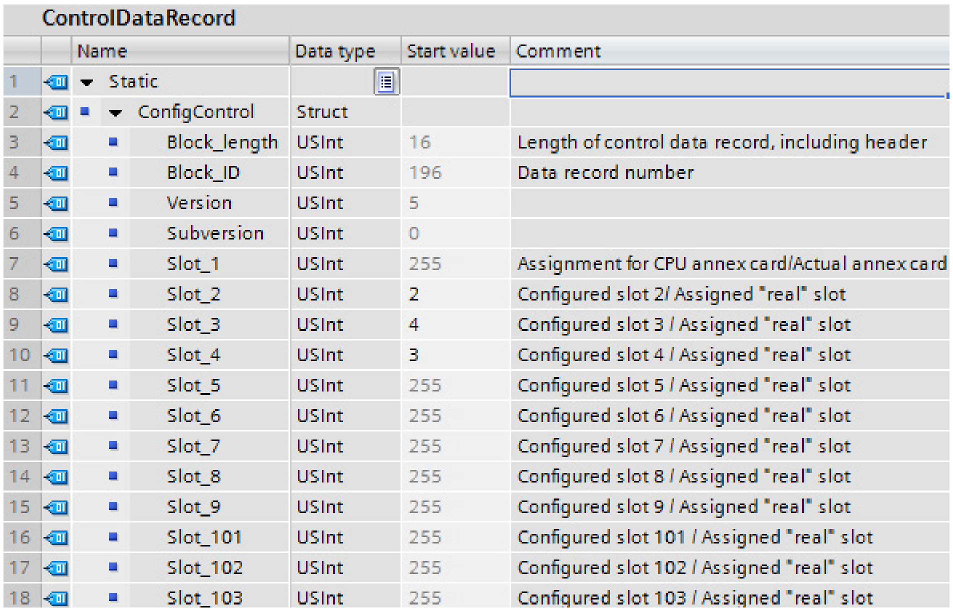


Figure 6-3 Device configuration compared to actual installation with modules in slots 3 and 4 swapped

To correlate the device configuration to the actual installation, edit the control data record to assign the modules to the correct slot positions.



ControlDataRecord

	Name	Data type	Start value	Comment
1	Static			
2	ConfigControl	Struct		
3	Block_length	USInt	16	Length of control data record, including header
4	Block_ID	USInt	196	Data record number
5	Version	USInt	5	
6	Subversion	USInt	0	
7	Slot_1	USInt	255	Assignment for CPU annex card/Actual annex card
8	Slot_2	USInt	2	Configured slot 2/ Assigned "real" slot
9	Slot_3	USInt	4	Configured slot 3 / Assigned "real" slot
10	Slot_4	USInt	3	Configured slot 4 / Assigned "real" slot
11	Slot_5	USInt	255	Configured slot 5 / Assigned "real" slot
12	Slot_6	USInt	255	Configured slot 6 / Assigned "real" slot
13	Slot_7	USInt	255	Configured slot 7 / Assigned "real" slot
14	Slot_8	USInt	255	Configured slot 8 / Assigned "real" slot
15	Slot_9	USInt	255	Configured slot 9 / Assigned "real" slot
16	Slot_101	USInt	255	Configured slot 101 / Assigned "real" slot
17	Slot_102	USInt	255	Configured slot 102 / Assigned "real" slot
18	Slot_103	USInt	255	Configured slot 103 / Assigned "real" slot

6.5 Changing a device

You can change the device type of a configured CPU or module. From Device configuration, right-click the device and select "Change device" from the context menu. From the dialog, navigate to and select the CPU or module that you want to replace. The Change device dialog shows you compatibility information between the two devices.

Note

Device exchange: replacing a V3.0 CPU with a V4.1 CPU

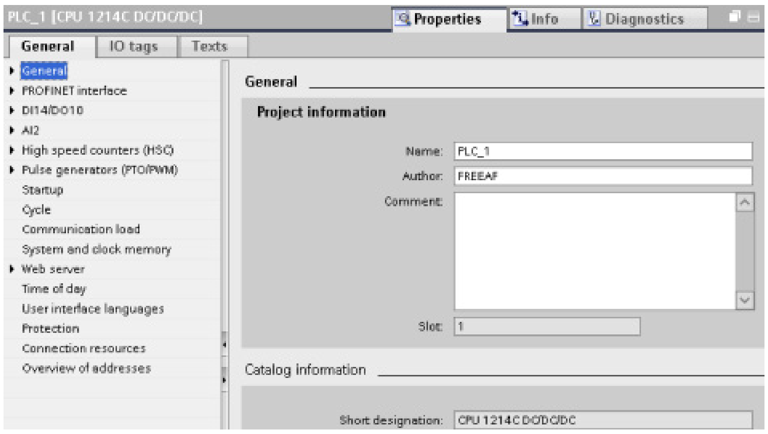
You can open a STEP 7 V12 project in STEP 7 V13 and replace V3.0 CPUs with V4.1 CPUs. You cannot replace CPUs that are from versions prior to V3.0. When you replace a V3.0 CPU with a V4.1 CPU, consider the differences (Page 1287) in features and behavior between the two versions, and actions you must take.

If you have a project for a CPU version older than V3.0, you must first upgrade the CPU to V3.0 and then upgrade it to V4.1.

6.6 Configuring the operation of the CPU

6.6.1 Overview

To configure the operational parameters for the CPU, select the CPU in the Device view (blue outline around whole CPU), and use the "Properties" tab of the inspector window.



Name: PLC_1
Author: FREEAF
Comment:
Slot: 1
Short designation: CPU 1214C DC/DC/DC

Table 6- 2 CPU properties

Property	Description
PROFINET interface	Sets the IP address for the CPU and time synchronization
DI, DO, and AI	Configures the behavior of the local (on-board) digital and analog I/O (for example, digital input filter times and digital output reaction to a CPU stop).
High-speed counters (Page 457) and pulse generators (Page 407)	Enables and configures the high-speed counters (HSC) and the pulse generators used for pulse-train operations (PTO) and pulse-width modulation (PWM) When you configure the outputs of the CPU or signal board as pulse generators (for use with the PWM or motion control instructions), the corresponding output addresses are removed from the Q memory and cannot be used for other purposes in your user program. If your user program writes a value to an output used as a pulse generator, the CPU does not write that value to the physical output.
Startup (Page 83)	**Startup after POWER ON:** Selects the behavior of the CPU following an off-to-on transition, such as to start in STOP mode or to go to RUN mode after a warm restart
	Supported hardware compatibility: Configures the substitution strategy for all system components (SM, SB, CM, CP and CPU): • Allow acceptable substitute • Allow any substitute (default) Each module internally contains substitution compatibility requirements based on the number of I/O, electrical compatibility, and other corresponding points of comparison. For example, a 16-channel SM could be an acceptable substitute for an 8-channel SM, but an 8-channel SM could not be an acceptable substitute for a 16-channel SM. If you select "Allow acceptable substitute", STEP 7 enforces the substitution rules; otherwise, STEP 7 allows any substitution.

Property	Description
	Parameter assignment time for distributed I/O: Configures a maximum amount of time (default: 60000 ms) for the distributed I/O to be brought online. (The CMs and CPs receive power and communication parameters from the CPU during startup. This assignment time allows time for the I/O connected to the CM or CP to be brought online.) The CPU goes to RUN as soon as the distributed I/O is online, regardless of the assignment time. If the distributed I/O has not been brought online within this time, the CPU still goes to RUN--without the distributed I/O. **Note:** If your configuration uses a CM 1243-5 (PROFIBUS master), do not set this parameter below 15 seconds (15000 ms) to ensure that the module can be brought online.
Cycle (Page 101)	Defines a maximum cycle time or a fixed minimum cycle time
Communication load	Allocates a percentage of the CPU time to be dedicated to communication tasks
System and clock memory (Page 105)	Enables a byte for "system memory" functions and enables a byte for "clock memory" functions (where each bit toggles on and off at a predefined frequency).
Web server (Page 785)	Enables and configures the Web server feature.
Time of day	Selects the time zone and configures daylight saving time
User interface languages	Selects a language for Web server and CPU display corresponding to the project language. For up to two project languages, you can assign a corresponding user interface language for Web server and CPU display.
Protection (Page 197)	Sets the read/write protection and passwords for accessing the CPU
Connection resources (Page 613)	Provides a summary of the communication connection resources that are available for the CPU and the number of connection resources that have been configured.
Overview of addresses	Provides a summary of the I/O addresses that have been configured for the CPU.

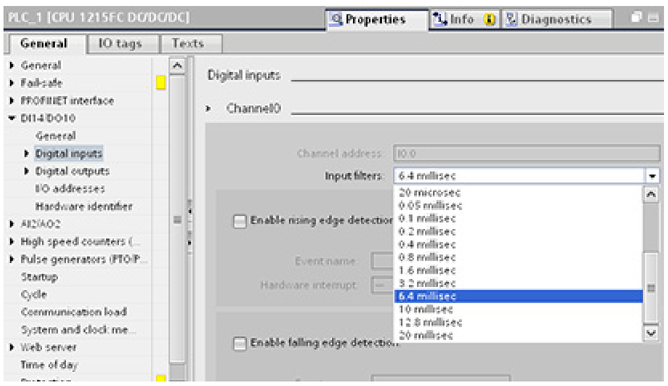
6.6.2 Configuring digital input filter times

The digital input filters protect your program from responding to unwanted fast changes in the input signals, as may result from switch contact bounce or electrical noise. The default filter time of 6.4 ms blocks unwanted transitions from typical mechanical contacts. Different points in your application can require shorter filter times to detect and respond to inputs from fast sensors, or longer filter times to block slow contact bounce or longer impulse noise.

An input filter time of 6.4 ms means that a single signal change, from ‘0’ to ‘1’ or from ‘1’ to ‘0’, must continue for approximately 6.4 ms to be detected, and a single high or low pulse shorter than approximately 6.4 ms is not detected. If an input signal switches between ‘0’ and ‘1’ more rapidly than the filter time, the input point value can change in the user program when the accumulated duration of new value pulses over old value pulses exceeds the filter time.

The digital input filter works this way:

- When a "1" is input, it counts up, stopping at the filter time. The image register point changes from "0" to "1" when the count reaches the filter time.
- When a "0" is input, it counts down, stopping at "0". The image register point changes from "1" to "0" when the count reaches "0".
- If the input is changing back and forth, the counter will count up some and count down some. The image register will change when the net accumulation of counts reaches either the filter time or "0".
- A rapidly-changing signal with more "0’s" than "1’s" will eventually go to "0", and if there are more "1’s" than "0’s", the image register will eventually change to "1".



Channel address: I0.0
Input filters: 6.4 millisec

Each input point has a single filter configuration that applies to all uses: process inputs, interrupts, pulse catch, and HSC inputs. To configure input filter times, select "Digital Inputs".

The default filter time for the digital inputs is 6.4 ms. You can select a filter time from the Input filters drop-down list. Valid filter times range from 0.1 us to 20.0 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 20.0 ms accumulated duration before the filter becomes fully responsive to new inputs. During this time, short "0" pulse events of duration less than 20.0 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.

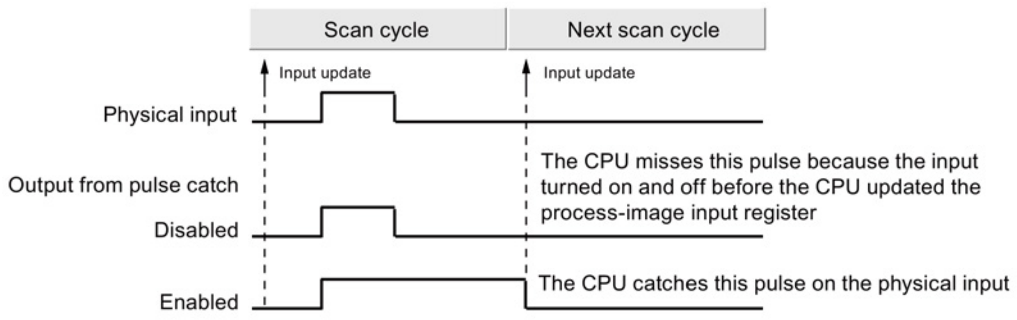
To ensure that a new filter time goes immediately into effect, a power cycle of the CPU must be applied.

6.6.3 Pulse catch

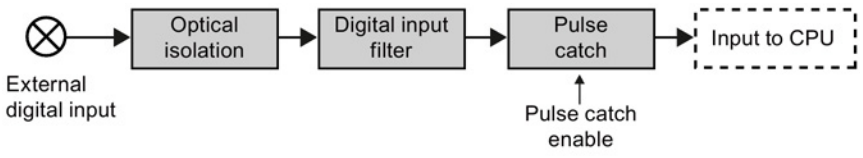
The S7-1200 CPU provides a pulse catch feature for digital input points. The pulse catch feature allows you to capture high-going pulses or low-going pulses that are of such a short duration that they would not always be seen when the CPU reads the digital inputs at the beginning of the scan cycle.

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 CPU reads the inputs.

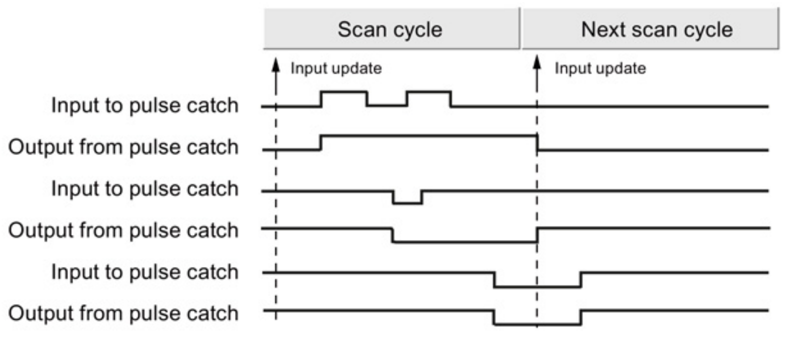
The figure below shows the basic operation of the S7-1200 CPU with and without pulse catch enabled:


Scan cycle
Next scan cycle
Input update
Input update
Physical input
Output from pulse catch
Disabled
Enabled
The CPU misses this pulse because the input turned on and off before the CPU updated the process-image input register
The CPU catches this pulse on the physical input

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:



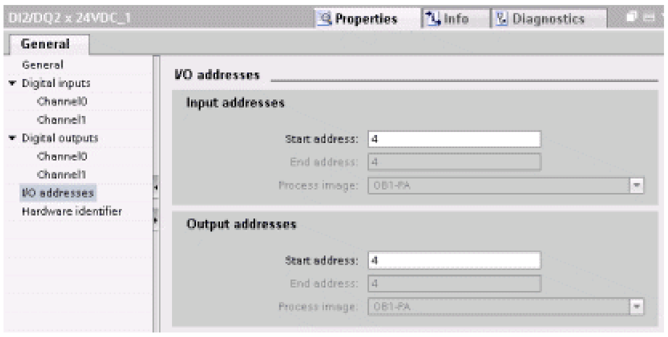
6.7 Configuring the parameters of the modules

To configure the operational parameters for the modules, select the module in the Device view and use the "Properties" tab of the inspector window to configure the parameters for the module.

Configuring a signal module (SM) or a signal board (SB)

The device configuration for signal modules and signal boards provides the means to configure the following:

- Digital I/O: You can configure inputs for rising-edge detection or falling-edge detection (associating each with an event and hardware interrupt) or for "pulse catch" (to stay on after a momentary pulse) through the next update of the input process image. Outputs can use a freeze or substitute value.
- Analog I/O: For individual inputs, configure parameters, such as measurement type (voltage or current), range and smoothing, and to enable underflow or overflow diagnostics. Analog outputs provide parameters such as output type (voltage or current) and for diagnostics, such as short circuit (for voltage outputs) or upper/lower limit diagnostics. You do not configure ranges of analog inputs and outputs in engineering units on the Properties dialog. You must handle this in your program logic as described in the topic "Processing of analog values (Page 115)".
- I/O addresses: You configure the start address for the set of inputs and outputs of the module. You can also assign the inputs and outputs to a process image partition (PIP0, PIP1, PIP2, PIP3, PIP4) or to automatically update, or to use no process image partition. See "Execution of the user program" (Page 79) for an explanation of the process image and process image partitions.



I/O addresses

Input addresses

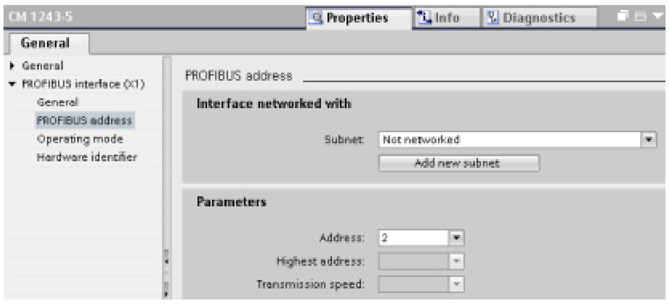
Start address: 4
End address: 4
Process image: OB1-PA

Output addresses

Start address: 4
End address: 4
Process image: OB1-PA

Configuring a communication interface (CM, CP or CB)

Depending on the type of communication interface, you configure the parameters for the network.



Subnet: Not networked

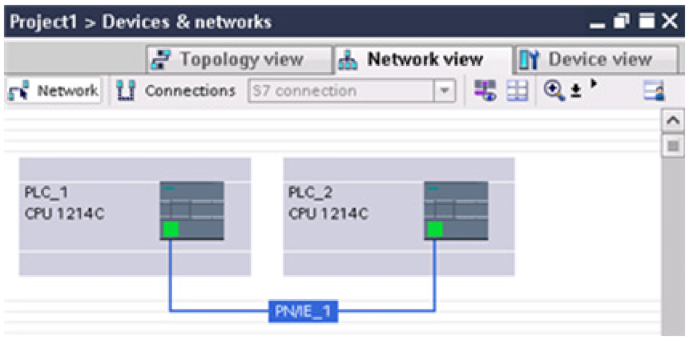
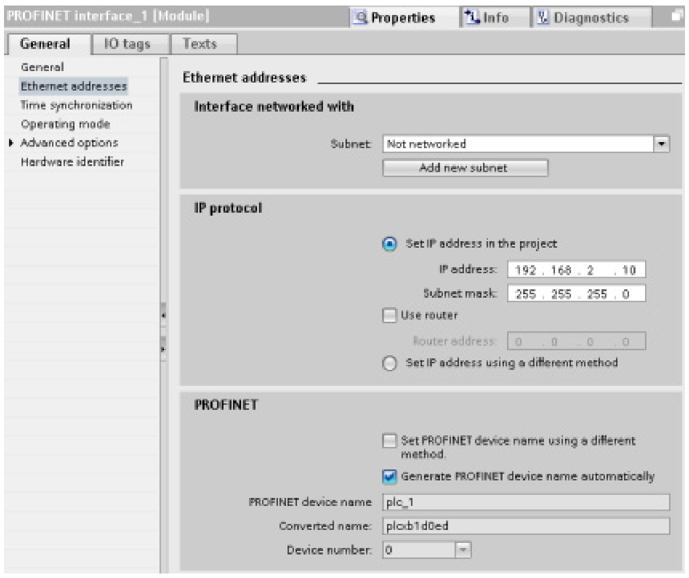
Address: 2

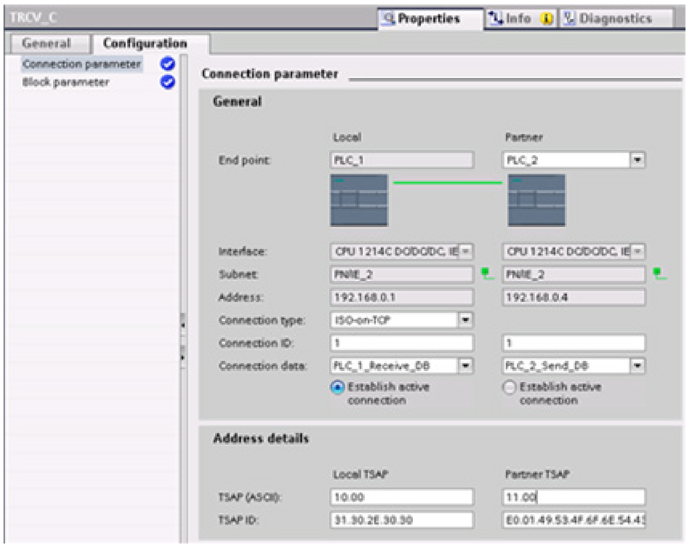
Highest address:

Transmission speed:

6.8 Configuring the CPU for communication

The S7-1200 is designed to solve your communications and networking needs by supporting not only the simplest of networks but also supporting more complex networks. The S7-1200 also provides tools that allow you to communicate with other devices, such as printers and weigh scales which use their own communications protocols.

 Project1 > Devices & networks Topology view Network view Device view Network Connections S7 connection PLC_1 CPU 1214C PLC_2 CPU 1214C PN/IE_1	Use the "Network view" of Device configuration to create the network connections between the devices in your project. After creating the network connection, use the "Properties" tab of the inspector window to configure the parameters of the network. Refer to "Creating a network connection" (Page 618) for further information.
 PROFINET interface_1 [Module] Properties · Info · Diagnostics General · IO tags · Texts General Ethernet addresses Time synchronization Operating mode Advanced options Hardware identifier **Ethernet addresses** **Interface networked with** Subnet: Not networked Add new subnet **IP protocol** [✓ Set IP address in the project] IP address: 192 . 168 . 2 . 10 Subnet mask: 255 . 255 . 255 . 0 ☐ Use router Router address: 0 . 0 . 0 . 0 Set IP address using a different method **PROFINET** ☐ Set PROFINET device name using a different method. ✓ Generate PROFINET device name automatically PROFINET device name: plc_1 Converted name: plcxb1d0ed Device number: 0	In the Properties window, select the "Ethernet addresses" configuration entry. STEP 7 displays the Ethernet address configuration dialog, which associates the software project with the IP address of the CPU that will receive that project. Note: The S7-1200 CPU does not have a pre-configured IP address. You must manually assign an IP address for the CPU. Refer to "Assigning Internet Protocol (IP) addresses" (Page 622) for further information.



Connection parameter

General

	Local	Partner
End point:	PLC_1	PLC_2
Interface:	CPU 1214C DC/DC/DC, IE	CPU 1214C DC/DC/DC, IE
Subnet:	PN/IE_2	PN/IE_2
Address:	192.168.0.1	192.168.0.4
Connection type:	ISO-on-TCP	
Connection ID:	1	1
Connection data:	PLC_1_Receive_DB	PLC_2_Send_DB
	[✓ Establish active connection]	Establish active connection

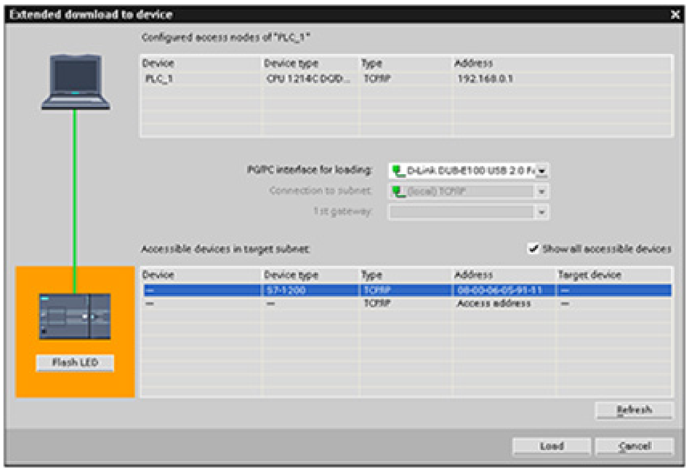
Address details

	Local TSAP	Partner TSAP
TSAP (ASCII):	10.00	11.00
TSAP ID:	31.30.2E.30.30	E0.01.49.53.4F.6F.6E.54.4

For the TCP, ISO-on-TCP, and UDP Ethernet protocols, use the "Properties" of the instruction (TSEND_C, TRCV_C, or TCON) to configure the "Local/Partner" connections.

The figure shows the "Connection properties" of the "Configuration tab" for an ISO-on-TCP connection.

Refer to "Configuring the Local/Partner connection path" (Page 619) for further information.



Configured access nodes of "PLC_1"

Device	Device type	Type	Address
PLC_1	CPU 1214C DC/D...	TCP/IP	192.168.0.1

PG/PC interface for loading: D-Link DUB-E100 USB 2.0 F

Connection to subnet: (local) TCP/IP

1st gateway:

Accessible devices in target subnet: [✓ Show all accessible devices]

Device	Device type	Type	Address	Target device
—	S7-1200	TCP/IP	00-00-06-05-91-11	—
—	—	TCP/IP	Access address	—

After completing the configuration, download the project to the CPU. All IP addresses are configured when you download the project.

Refer to "Testing the PROFINET network" (Page 629) for further information.

Note

To make a connection to your CPU, your 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.

Refer to "Assigning Internet Protocol (IP) addresses" (Page 622) for information about how to accomplish this.

Programming concepts 7

7.1 Guidelines for designing a PLC system

When designing a PLC system, you can choose from a variety of methods and criteria. 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.

Table 7- 1 Guidelines for designing a PLC system

Recommended steps	Tasks
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, such as the I/O points, the functional description of the operation, the states that must be achieved before allowing action for each actuator (such as a solenoid, a motor, or a drive), a description of the operator interface, and any interfaces with other sections of the process or machine.
Design the safety circuits	Identify any equipment that might require hard-wired logic for safety. Remember that control devices can fail in an unsafe manner, which can produce 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, consider the implementation of electro-mechanical overrides (which operate independently of the PLC) to prevent unsafe operations. The following tasks should be included in the design of safety circuits: • Identify any 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 PLC. • Identify how the PLC affects the process when power is applied and removed, and also identify how and when errors are detected. Use this information only for designing the normal and expected abnormal operation. You should not rely on this "best case" scenario for safety purposes. • Design the manual or electromechanical safety overrides that block the hazardous operation independent of the PLC. • Provide the appropriate status information from the independent circuits to the PLC so that the program and any operator interfaces have necessary information. • Identify any other safety-related requirements for safe operation of the process.
Plan system security	Determine what level of protection (Page 197) you require for access to your process. You can password-protect CPUs and program blocks from unauthorized access.

Recommended steps	Tasks
Specify the operator stations	Based on the requirements of the functional specifications, create the following drawings of the operator stations: • Overview drawing that shows the location of each operator station in relation to the process or machine. • Mechanical layout drawing of the devices for the operator station, such as display, switches, and lights. • Electrical drawings with the associated I/O of the PLC and signal modules.
Create the configura-tion drawings	Based on the requirements of the functional specification, create configuration drawings of the control equipment: • Overview drawing that shows the location of each PLC in relation to the process or machine. • Mechanical layout drawing of each PLC and any I/O modules, including any cabinets and other equipment. • Electrical drawings for each PLC and any I/O modules, including the device model numbers, communications addresses, and I/O addresses.
Create a list of symbol-ic names	Create a list of symbolic names for the absolute addresses. Include not only the physical I/O signals, but also the other elements (such as tag names) to be used in your program.

7.2 Structuring your user program

When you create a user program for the automation tasks, you insert the instructions for the program into code blocks:

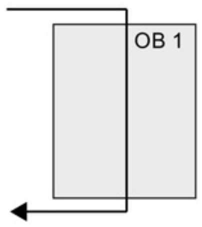
- An organization block (OB) responds to a specific event in the CPU and can interrupt the execution of the user program. The default for the cyclic execution of the user program (OB 1) provides the base structure for your user program. If you include other OBs in your program, these OBs interrupt the execution of OB 1. The other OBs perform specific functions, such as for startup tasks, for handling interrupts and errors, or for executing specific program code at specific time intervals.
- A function block (FB) is a subroutine that is executed when called from another code block (OB, FB, or FC). The calling block passes parameters to the FB and also identifies a specific data block (DB) that stores the data for the specific call or instance of that FB. Changing the instance DB allows a generic FB to control the operation of a set of devices. For example, one FB can control several pumps or valves, with different instance DBs containing the specific operational parameters for each pump or valve.
- A function (FC) is a subroutine that is executed when called from another code block (OB, FB, or FC). The FC does not have an associated instance DB. The calling block passes parameters to the FC. The output values from the FC must be written to a memory address or to a global DB.

Choosing the type of structure for your user program

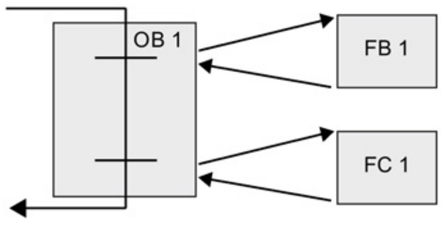
Based on the requirements of your application, you can choose either a linear structure or a modular structure for creating your user program:

- A linear program executes all of the instructions for your automation tasks in sequence, one after the other. Typically, the linear program puts all of the program instructions into the OB for the cyclic execution of the program (OB 1).
- A modular program calls specific code blocks that perform specific tasks. To create a modular structure, you divide the complex automation task into smaller subordinate tasks that correspond to the technological functions of the process. Each code block provides the program segment for each subordinate task. You structure your program by calling one of the code blocks from another block.

Linear structure:



Modular structure:



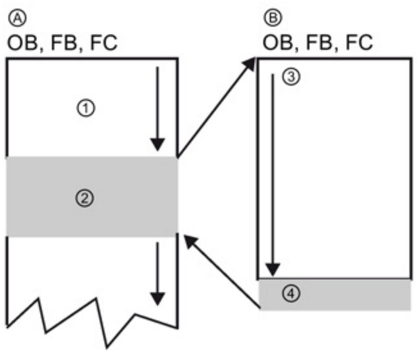
By creating generic code blocks that can be reused within the user program, you can simplify the design and implementation of the user program. Using generic code blocks has a number of benefits:

- You can create reusable blocks of code for standard tasks, such as for controlling a pump or a motor. You can also store these generic code blocks in a library that can be used by different applications or solutions.
- When you structure the user program into modular components that relate to functional tasks, the design of your program can be easier to understand and to manage. The modular components not only help to standardize the program design, but can also help to make updating or modifying the program code quicker and easier.
- Creating modular components simplifies the debugging of your program. By structuring the complete program as a set of modular program segments, you can test the functionality of each code block as it is developed.
- Creating modular components that relate to specific technological functions can help to simplify and reduce the time involved with commissioning the completed application.

7.3 Using blocks to structure your program

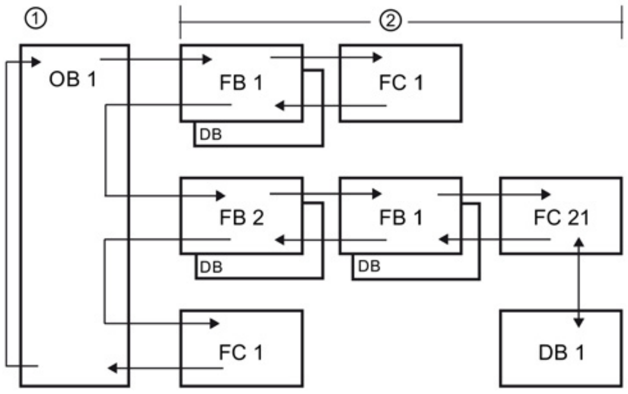
By designing FBs and FCs to perform generic tasks, you create modular code blocks. You then structure your program by having other code blocks call these reusable modules. The calling block passes device-specific parameters to the called block.

When a code block calls another code block, the CPU executes the program code in the called block. After execution of the called block is complete, the CPU resumes the execution of the calling block. Processing continues with execution of the instruction that follows after the block call.



A	Calling block
B	Called (or interrupting) block
①	Program execution
②	Instruction or event that initiates the execution of another block
③	Program execution
④	Block end (returns to calling block)

You can nest the block calls for a more modular structure. In the following example, the nesting depth is 3: the program cycle OB plus 3 layers of calls to code blocks.



①	Start of cycle
②	Nesting depth

7.3.1 Organization block (OB)

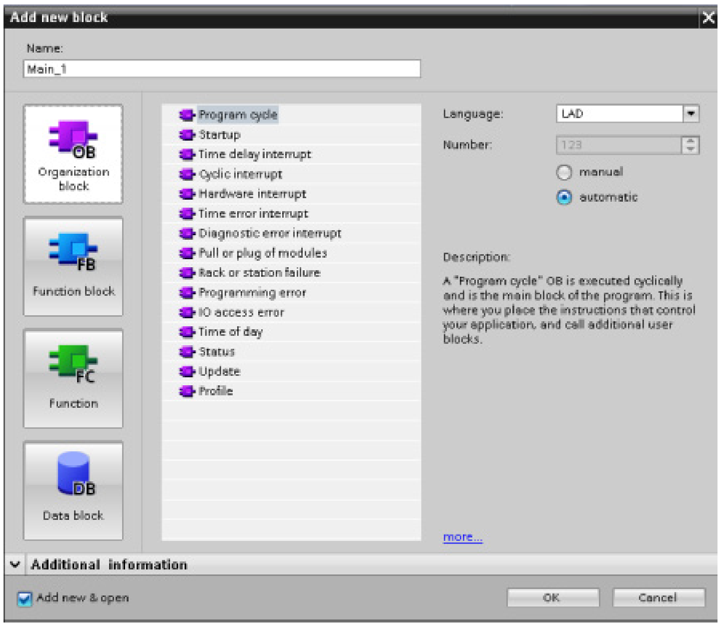
Organization blocks provide structure for your program. They serve as the interface between the operating system and the user program. OBs are event driven. An event, such as a diagnostic interrupt or a time interval, causes the CPU to execute an OB. Some OBs have predefined start events and behavior.

The program cycle OB contains your main program. You can include more than one program cycle OB in your user program. During RUN mode, the program cycle OBs execute at the lowest priority level and can be interrupted by all other event types. The startup OB does not interrupt the program cycle OB because the CPU executes the startup OB before going to RUN mode.

After finishing the processing of the program cycle OBs, the CPU immediately executes the program cycle OBs again. This cyclic processing is the "normal" type of processing used for programmable logic controllers. For many applications, the entire user program is located in a single program cycle OB.

You can create other OBs to perform specific functions, such as for handling interrupts and errors, or for executing specific program code at specific time intervals. These OBs interrupt the execution of the program cycle OBs.

Use the "Add new block" dialog to create new OBs in your user program.



Name: Main_1

Language: LAD

Number: 123

( ) manual [✓ automatic]

Description:

A "Program cycle" OB is executed cyclically and is the main block of the program. This is where you place the instructions that control your application, and call additional user blocks.

[✓] Add new & open

Interrupt handling is always event-driven. When such an event occurs, the CPU interrupts the execution of the user program and calls the OB that was configured to handle that event. After finishing the execution of the interrupting OB, the CPU resumes the execution of the user program at the point of interruption.

The CPU determines the order for handling interrupt events by priority. You can assign multiple interrupt events to the same priority class. For more information, refer to the topics on organization blocks (Page 88) and execution of the user program (Page 79).

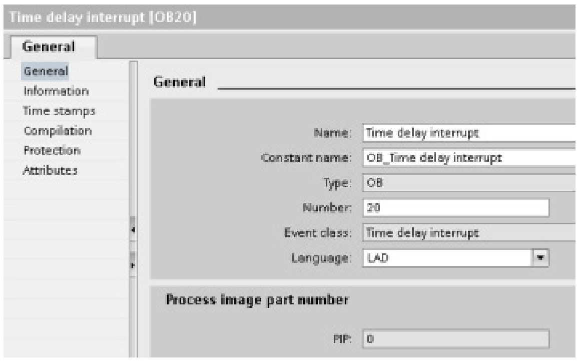
Creating additional OBs

You can create multiple OBs for your user program, even for the program cycle and startup OB events. Use the "Add new block" dialog to create an OB and enter a name for your OB.

If you create multiple program cycle OBs for your user program, the CPU executes each program cycle OB in numerical sequence, starting with the program cycle OB with the lowest number (such as OB 1). For example: after the first program cycle OB (such as OB 1) finishes, the CPU executes the program cycle OB with the next higher number.

Configuring the properties of an OB

You can modify the properties of an OB. For example, you can configure the OB number or programming language.



General

Name: Time delay interrupt
Constant name: OB_Time delay interrupt
Type: OB
Number: 20
Event class: Time delay interrupt
Language: LAD

Process image part number

PIP: 0

Note

Note that you can assign a process image part number to an OB that corresponds to PIP0, PIP1, PIP2, PIP3, or PIP4. If you enter a number for the process image part number, the CPU creates that process image partition. See the topic "Execution of the user program (Page 79)" for an explanation of the process image partitions.

7.3.2 Function (FC)

A function (FC) is a code block that typically performs a specific operation on a set of input values. The FC stores the results of this operation in memory locations. For example, use FCs to perform standard and reusable operations (such as for mathematical calculations) or technological functions (such as for individual controls using bit logic operations). An FC can also be called several times at different points in a program. This reuse simplifies the programming of frequently recurring tasks.

An FC does not have an associated instance data block (DB). The FC uses the local data stack for the temporary data used to calculate the operation. The temporary data is not saved. To store data permanently, assign the output value to a global memory location, such as M memory or to a global DB.

7.3.3 Function block (FB)

A function block (FB) is a code block that uses an instance data block for its parameters and static data. FBs have variable memory that is located in a data block (DB), or "instance" DB. The instance DB provides a block of memory that is associated with that instance (or call) of the FB and stores data after the FB finishes. You can associate different instance DBs with different calls of the FB. The instance DBs allow you to use one generic FB to control multiple devices. You structure your program by having one code block make a call to an FB and an instance DB. The CPU then executes the program code in that FB, and stores the block parameters and the static local data in the instance DB. When the execution of the FB finishes, the CPU returns to the code block that called the FB. The instance DB retains the values for that instance of the FB. These values are available to subsequent calls to the function block either in the same scan cycle or other scan cycles.

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 instance DBs. When you call an FB, you also specify an instance DB that contains the block parameters and the static local data for that call or "instance" of the FB. The instance DB maintains these values after the FB finishes execution.

By designing the FB for generic control tasks, you can reuse the FB for multiple devices by selecting different instance DBs for different calls of the FB.

An FB stores the Input, Output, and InOut, and Static parameters in an instance DB.

You can also modify and download the function block interface in RUN mode (Page 1092).

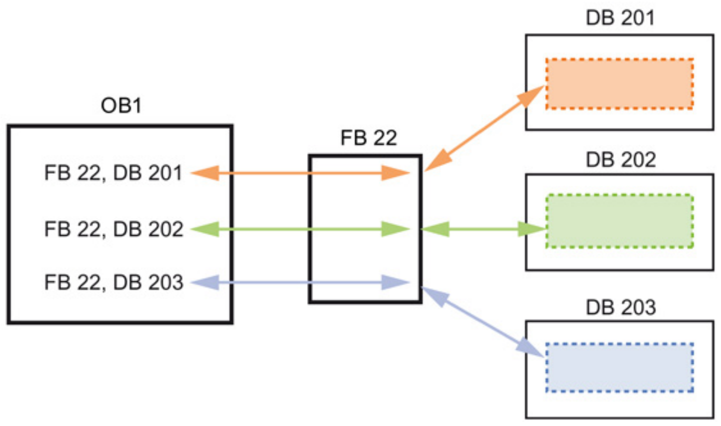
Assigning the start value in the instance DB

The instance DB stores both a default value and a start value for each parameter. The start value provides the value to be used when the FB is executed. The start value can then be modified during the execution of your user program.

The FB interface also provides a "Default value" column that allows you to assign a new start value for the parameter as you are writing the program code. This default value in the FB is then transferred to the start value in the associated instance DB. If you do not assign a new start value for a parameter in the FB interface, the default value from instance DB is copied to start value.

Using a single FB with DBs

The following figure shows an OB that calls one FB three times, using a different data block for each call. This structure allows one generic FB to control several similar devices, such as motors, by assigning a different instance data block for each call for the different devices. Each instance DB stores the data (such as speed, ramp-up time, and total operating time) for an individual device.



In this example, FB 22 controls three separate devices, with DB 201 storing the operational data for the first device, DB 202 storing the operational data for the second device, and DB 203 storing the operational data for the third device.

7.3.4 Data block (DB)

You create data blocks (DB) in your user program to store data for the code blocks. All of the program blocks in the user program can access the data in a global DB, but an instance DB stores data for a specific function block (FB).

The data stored in a DB is not deleted when the execution of the associated code block comes to an end. There are two types of DBs:

- A global DB stores data for the code blocks in your program. Any OB, FB, or FC can access the data in a global DB.
- An instance DB stores the data for a specific FB. The structure of the data in an instance DB reflects the parameters (Input, Output, and InOut) and the static data for the FB. (The Temp memory for the FB is not stored in the instance DB.)

Note

Although the instance DB reflects the data for a specific FB, any code block can access the data in an instance DB.

You can also modify and download data blocks in RUN mode (Page 1092).

Read-only data blocks

You can configure a DB as being read-only:

1. Right-click the DB in the project navigator and select "Properties" from the context menu.
2. In the "Properties" dialog, select "Attributes".
3. Select the "Data block write-protected in the device" option and click "OK".

Optimized and standard data blocks

You can also configure a data block to be either standard or optimized. A standard DB is compatible with STEP 7 Classic programming tools and the classic S7-300 and S7-400 CPUs. Data blocks with optimized access have no fixed defined structure. The data elements contain only a symbolic name in the declaration and no fixed address within the block. The CPU stores the elements automatically in the available memory area of the block so that there are no gaps in the memory. This makes for optimal use of the memory capacity.

To set optimized access for a data block, follow these steps:

1. Expand the program blocks folder in the STEP 7 project tree.
2. Right-click the data block and select "Properties" from the context menu.
3. For the attributes, select "Optimized block access".

Note that optimized block access is the default for new data blocks. If you deselect "Optimized block access", the block uses standard access.

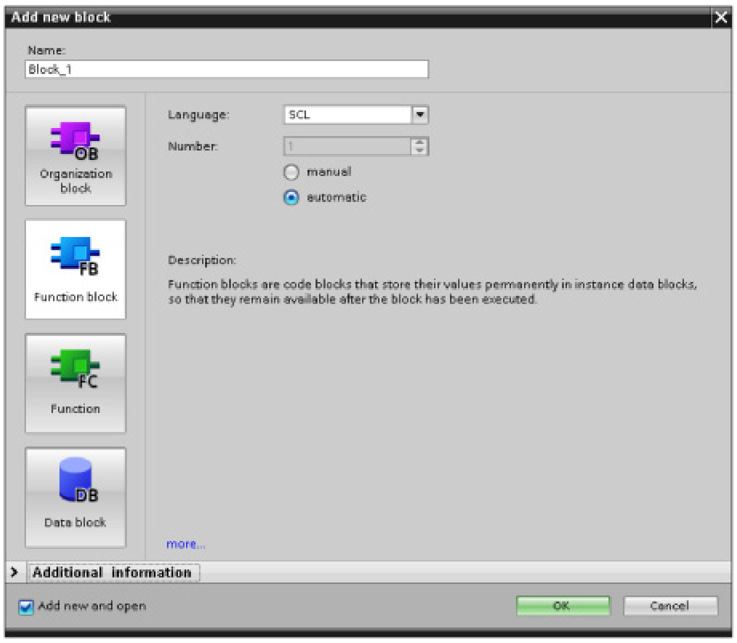
Note

Block access type for an FB and its instance DB

Be sure that if your FB setting is "Optimized block access" then the setting of the instance DB for that FB is also "Optimized block access". Similarly if you have not selected "Optimized block access" for the FB such that the FB is of type standard access, then be sure that the instance DB is also standard, or not optimized block access.

If you do not have compatible block access types, then changes to the InOut parameter values of the FB from an HMI during execution of the FB could be lost.

7.3.5 Creating reusable code blocks



Name: Block_1

Language: SCL

Number: 1

manual [✓ automatic]

Description:

Function blocks are code blocks that store their values permanently in instance data blocks, so that they remain available after the block has been executed.

[✓] Add new and open

Use the "Add new block" dialog under "Program blocks" in the Project navigator to create OBs, FBs, FCs, and global DBs.

When you create a code block, you select the programming language for the block. You do not select a language for a DB because it only stores data.

Selecting the "Add new and open" check box (default) opens the code block in the Project view.

You can store objects you want to reuse in libraries. For each project, there is a project library that is connected to the project. In addition to the project library, you can create any number of global libraries that can be used over several projects. Since the libraries are compatible with each other, library elements can be copied and moved from one library to another.

Libraries are used, for example, to create templates for blocks that you first paste into the project library and then further develop there. Finally, you copy the blocks from the project library to a global library. You make the global library available to other colleagues working on your project. They use the blocks and further adapt them to their individual requirements, where necessary.

For details about library operations, refer to the STEP 7 online Help library topics.

7.3.6 Passing parameters to blocks

Function Blocks (FB) and Functions (FC) have three different interface types:

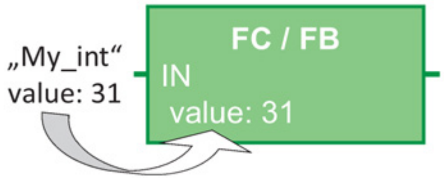
- IN
- IN/OUT
- OUT

FBs and FCs receive parameters through the IN and IN/OUT interface types. The blocks process the parameters and return values to the caller through the IN/OUT and OUT interface types.

The user program transfers parameters using one of two methods.

Call-by-value

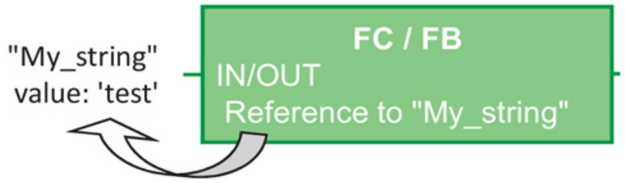
When the user program passes a parameter to a function as "call-by-value", the user program copies the actual parameter value into the input parameter of the block for the IN interface type. This operation requires additional memory for the copied value.



When the user program calls the block, it copies the values.

Call-by-reference

When the user program passes a parameter to a function as "call-by-reference", the user program references the address of the actual parameter for the IN/OUT interface type and does not copy the value. This operation does not require additional memory.



When the user program calls the block, it references the address of the actual parameters.

Note

Generally, use the IN/OUT interface type for structured tags (for example, ARRAY, STRUCT, and STRING) in order to avoid increasing the required data memory unnecessarily.

Block optimization and passing parameters

The user program passes FC parameters as "call-by-value" for simple data types (for example, INT, DINT, and REAL). It passes complex data types (for example, STRUCT, ARRAY, and STRING) as "call-by-reference".

The user program normally passes FB parameters in the instance Data block (DB) associated with the FB:

- The user program passes simple data types (for example, INT, DINT, and REAL) as "call-by-value" by copying the parameters to/from the instance DB.
- The user program copies complex data types (for example, STRUCT, ARRAY, and STRING) to and from the instance DB for IN and OUT parameter types.
- The user program passes complex data types as "call-by-reference" for the IN/OUT interface type.

DBs can be created as either "Optimized" or "Standard" (non-optimized). The optimized data blocks are more compact than the non-optimized data blocks. Also, the ordering of the data elements within the DB is different for optimized versus non-optimized DBs. Refer to the "Optimized blocks" section of the S7-Programming Guideline for S7-1200/1500, STEP 7 (TIA Portal), 03/2014 (http://support.automation.siemens.com/WW/view/en/81318674) for a discussion of optimized blocks.

You create FBs and FCs to process either optimized or non-optimized data. You can select the "Optimized block access" check box as one of the attributes for the block. The user program optimizes program blocks by default, and the program blocks expect data passed to the block to be in the optimized format.

When the user program passes a complex parameter (for example, a STRUCT) to a function, the system checks the optimization setting of the data block containing the structure and the optimization setting of the program block. If you optimize both the data block and the function, then the user program passes the STRUCT as a "call-by-reference". The same is true if you select non-optimized for both the data block and the function.

However, if you make the function and data block optimization different (meaning that you optimized one block and not the other block), the STRUCT must be converted to the format expected by the function. For example, if you select non-optimized for the data block and optimized for the function, then a STRUCT in the data block must be converted to an optimized format before the function can process the STRUCT. The system does this conversion by making a "copy" of the STRUCT and converting it to the optimized format that the function expects.

In summary, when the user program passes a complex data type (for example, a STRUCT) to a function as an IN/OUT parameter, the function expects the user program to pass the STRUCT as a "call-by-reference":

- If you select optimized or non-optimized for both the data block containing the STRUCT and the function, the user program passes the data as "call-by-reference".
- If you do not configure the data block and the function with the same optimization settings (one is optimized and the other is non-optimized), the system must make a copy of the STRUCT before passing it to the function. Because the system has to make this copy of the structure, this converts the "call-by-reference", effectively, into a "call-by-value".

Effect of optimization settings on user programs

The copying of the parameter can cause an issue in a user program if an HMI or interrupt OB modifies elements of the structure. For example, there is an IN/OUT parameter of a function (normally passed as "call-by-reference"), but the optimization settings of the data block and function are different:

1. When the user program is ready to call the function, the system must make a "copy" of the structure to change the format of the data to match the function.
2. The user program calls the function with a reference to the "copy" of the structure.
3. An interrupt OB occurs while the function is executing, and the interrupt OB changes a value in the original structure.
4. The function completes and, since the structure is an IN/OUT parameter, the system copies the values back to the original structure in the original format.

The effect of making the copy of the structure to change the format is that the data written by the interrupt OB is lost. The same can happen when writing a value with an HMI. The HMI can interrupt the user program and write a value in the same manner as an interrupt OB.

There are multiple ways to correct this issue:

- The best solution for this this issue is to match the optimization settings of the program block and the data block when using complex data types (for example, a STRUCT). This ensures that the user program always passes the parameters as "call-by-reference".
- Another solution is that an interrupt OB or HMI does not directly modify an element in the structure. The OB or HMI can modify another variable, and then you can copy this variable into the structure at a specific point in the user program.

7.4 Understanding data consistency

The CPU maintains the data consistency for all of the elementary data types (such as Words or DWords) and all of the system-defined structures (for example, IEC_TIMERS or DTL). The reading or writing of the value cannot be interrupted. (For example, the CPU protects the access to a DWord value until the four bytes of the DWord have been read or written.) To ensure that the program cycle OBs and the interrupt OBs cannot write to the same memory location at the same time, the CPU does not execute an interrupt OB until the read or write operation in the program cycle OB has been completed.

If your user program shares multiple values in memory between a program cycle OB and an interrupt OB, your user program must also ensure that these values are modified or read consistently. You can use the DIS_AIRT (disable alarm interrupt) and EN_AIRT (enable alarm interrupt) instructions in your program cycle OB to protect any access to the shared values.

- Insert a DIS_AIRT instruction in the code block to ensure that an interrupt OB cannot be executed during the read or write operation.
- Insert the instructions that read or write the values that could be altered by an interrupt OB.
- Insert an EN_AIRT instruction at the end of the sequence to cancel the DIS_AIRT and allow the execution of the interrupt OB.

A communication request from an HMI device or another CPU can also interrupt execution of the program cycle OB. The communication requests can also cause problems with data consistency. The CPU ensures that the elementary data types are always read and written consistently by the user program instructions. Because the user program is interrupted periodically by communications, it is not possible to guarantee that multiple values in the CPU will all be updated at the same time by the HMI. For example, the values displayed on a given HMI screen could be from different scan cycles of the CPU.

The PtP (Point-to-Point) instructions, PROFINET instructions (such as TSEND_C and TRCV_C), PROFINET Distributed I/O instructions (Page 350), and PROFIBUS Distributed I/O Instructions (Page 350) transfer buffers of data that could be interrupted. Ensure the data consistency for the buffers of data by avoiding any read or write operation to the buffers in both the program cycle OB and an interrupt OB. If it is necessary to modify the buffer values for these instructions in an interrupt OB, use a DIS_AIRT instruction to delay any interruption (an interrupt OB or a communication interrupt from an HMI or another CPU) until an EN_AIRT instruction is executed.

Note

The use of the DIS_AIRT instruction delays the processing of interrupt OBs until the EN_AIRT instruction is executed, affecting the interrupt latency (time from an event to the time when the interrupt OB is executed) of your user program.

7.5 Programming language

STEP 7 provides the following standard programming languages for S7-1200:

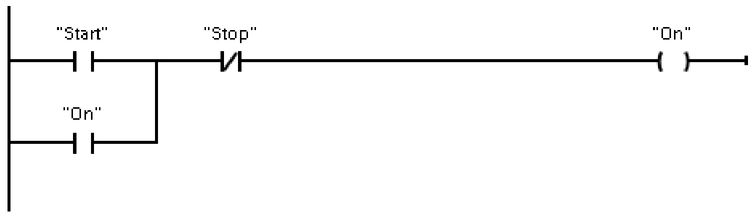
- LAD (ladder logic) is a graphical programming language. The representation is based on circuit diagrams (Page 186).
- FBD (Function Block Diagram) is a programming language that is based on the graphical logic symbols used in Boolean algebra (Page 187).
- SCL (structured control language) is a text-based, high-level programming language (Page 188).

When you create a code block, you select the programming language to be used by that block.

Your user program can utilize code blocks created in any or all of the programming languages.

7.5.1 Ladder logic (LAD)

The elements of a circuit diagram, such as normally closed and normally open contacts, and coils are linked to form networks.



To create the logic for complex operations, you can insert branches to create the logic for parallel circuits. Parallel branches are opened downwards or are connected directly to the power rail. You terminate the branches upwards.

LAD provides "box" instructions for a variety of functions, such as math, timer, counter, and move.

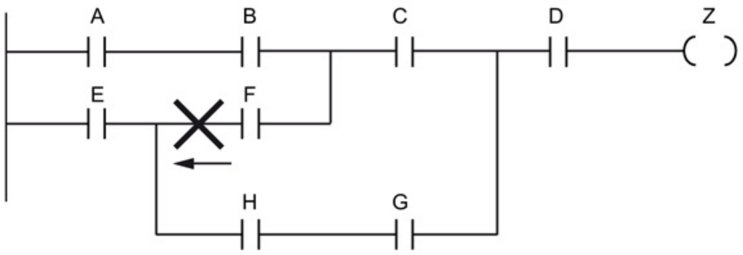
STEP 7 does not limit the number of instructions (rows and columns) in a LAD network.

Note

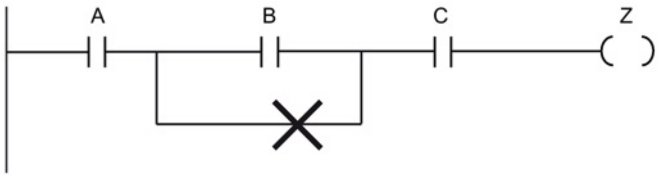
Every LAD network must terminate with a coil or a box instruction.

Consider the following rules when creating a LAD network:

- You cannot create a branch that could result in a power flow in the reverse direction.

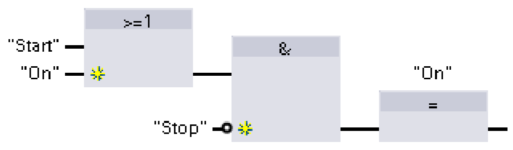


- You cannot create a branch that would cause a short circuit.



7.5.2 Function Block Diagram (FBD)

Like LAD, FBD is also a graphical programming language. The representation of the logic is based on the graphical logic symbols used in Boolean algebra.



To create the logic for complex operations, insert parallel branches between the boxes.

Mathematical functions and other complex functions can be represented directly in conjunction with the logic boxes.

STEP 7 does not limit the number of instructions (rows and columns) in an FBD network.

7.5.3 SCL

Structured Control Language (SCL) is a high-level, PASCAL-based programming language for the SIMATIC S7 CPUs. SCL supports the block structure of STEP 7 (Page 175). Your project can include program blocks in any of the three programming languages: SCL, LAD, and FBD.

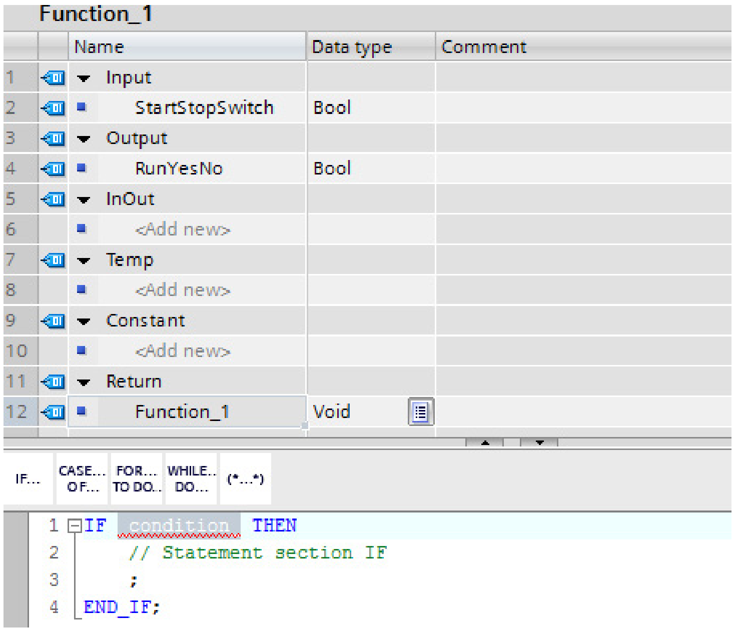
SCL instructions use standard programming operators, such as for assignment (:=), mathematical functions (+ for addition, - for subtraction, * for multiplication, and / for division). SCL also uses standard PASCAL program control operations, such as IF-THEN-ELSE, CASE, REPEAT-UNTIL, GOTO and RETURN. You can use any PASCAL reference for syntactical elements of the SCL programming language. Many of the other instructions for SCL, such as timers and counters, match the LAD and FBD instructions. For more information about specific instructions, refer to the specific instructions in the chapters for Basic instructions (Page 209) and Extended instructions (Page 315).

7.5.3.1 SCL program editor

You can designate any type of block (OB, FB, or FC) to use the SCL programming language at the time you create the block. STEP 7 provides an SCL program editor that includes the following elements:

- Interface section for defining the parameters of the code block
- Code section for the program code
- Instruction tree that contains the SCL instructions supported by the CPU

You enter the SCL code for your instruction directly in the code section. The editor includes buttons for common code constructs and comments. For more complex instructions, simply drag the SCL instructions from the instruction tree and drop them into your program. You can also use any text editor to create an SCL program and then import that file into STEP 7.


Function_1

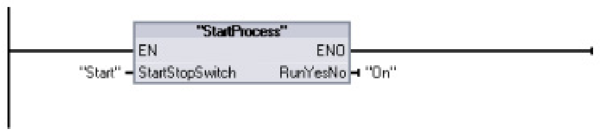
	Name	Data type	Comment
1	▼ Input		
2	StartStopSwitch	Bool	
3	▼ Output		
4	RunYesNo	Bool	
5	▼ InOut		
6	<Add new>		
7	▼ Temp		
8	<Add new>		
9	▼ Constant		
10	<Add new>		
11	▼ Return		
12	Function_1	Void	

```
IF condition THEN
    // Statement section IF
    ;
END_IF;
```

In the Interface section of the SCL code block you can declare the following types of parameters:

- Input, Output, InOut, and Ret_Val: These parameters define the input tags, output tags, and return value for the code block. The tag name that you enter here is used locally during the execution of the code block. You typically would not use the global tag name in the tag table.
- Static (FBs only; the illustration above is for an FC): The code block uses static tags for storage of static intermediate results in the instance data block. The block retains static data until overwritten, which can be after several cycles. The names of the blocks, which this block calls as multi-instance, are also stored in the static local data.
- Temp: These parameters are the temporary tags that are used during the execution of the code block.
- Constant: These are named constant values for your code block.

If you call the SCL code block from another code block, the parameters of the SCL code block appear as inputs or outputs.



In this example, the tags for "Start" and "On" (from the project tag table) correspond to "StartStopSwitch" and "RunYesNo" in the declaration table of the SCL program.

7.5.3.2 SCL expressions and operations

Constructing an SCL expression

An SCL expression is a formula for calculating a value. The expression consists of operands and operators (such as *, /, + or -). The operands can be tags, constants, or expressions.

The evaluation of the expression occurs in a certain order, which is defined by the following factors:

- Every operator has a pre-defined priority, with the highest-priority operation performed first.
- For operators with equal priority, the operators are processed in a left-to-right sequence.
- You use parentheses to designate a series of operators to be evaluated together.

The result of an expression can be used either for assigning a value to a tag used by your program, as a condition to be used by a control statement, or as parameters for another SCL instruction or for calling a code block.

Table 7- 2 Operators in SCL

Type	Operation	Operator	Priority
Parentheses	(*Expression*)	(,)	1
Math	Power	**	2
	Sign (unary plus)	+	3
	Sign (unary minus)	-	3
	Multiplication	*	4
	Division	/	4
	Modulo	MOD	4
	Addition	+	5
	Subtraction	-	5
Comparison	Less than	<	6
	Less than or equal to	<=	6
	Greater than	>	6
	Greater than or equal to	>=	6
	Equal to	=	7
	Not equal to	<>	7
Bit logic	Negation (unary)	NOT	3
	AND logic operation	AND or &	8
	Exclusive OR logic operation	XOR	9
	OR logic operation	OR	10
Assignment	Assignment	:=	11

As a high-level programming language, SCL uses standard statements for basic tasks:

- Assignment statement: :=
- Mathematical functions: +, -, *, and /
- Addressing of global variables (tags): "<tag name>" (Tag name or data block name enclosed in double quotes)
- Addressing of local variables: #<variable name> (Variable name preceded by "#" symbol)

The following examples show different expressions for different uses.

`"C" := #A+#B;`	Assigns the sum of two local variables to a tag
`"Data_block_1".Tag := #A;`	Assignment to a data block tag
`IF #A > #B THEN "C" := #A;`	Condition for the IF-THEN statement
`"C" := SQRT (SQR (#A) + SQR (#B));`	Parameters for the SQRT instruction

Arithmetic operators can process various numeric data types. The data type of the result is determined by the data type of the most-significant operands. For example, a multiplication operation that uses an INT operand and a REAL operand yields a REAL value for the result.

Control statements

A control statement is a specialized type of SCL expression that performs the following tasks:

- Program branching
- Repeating sections of the SCL program code
- Jumping to other parts of the SCL program
- Conditional execution

The SCL control statements include IF-THEN, CASE-OF, FOR-TO-DO, WHILE-DO, REPEAT-UNTIL, CONTINUE, GOTO, and RETURN.

A single statement typically occupies one line of code. You can enter multiple statements on one line, or you can break a statement into several lines of code to make the code easier to read. Separators (such as tabs, line breaks and extra spaces) are ignored during the syntax check. An END statement terminates the control statement.

The following examples show a FOR-TO-DO control statement. (Both forms of coding are syntactically valid.)

```
FOR x := 0 TO max DO sum := sum + value(x); END_FOR;
FOR x := 0 TO max DO
     sum := sum + value(x);
END_FOR;
```

A control statement can also be provided with a label. A label is set off by a colon at the beginning of the statement:

```
Label: <Statement>;
```

The STEP 7 online help provides a complete SCL programming language reference.

Conditions

A condition is a comparison expression or a logical expression whose result is of type BOOL (with the value of either TRUE or FALSE). The following example shows conditions of various types.

`#Temperature > 50`	Relational expression
`#Counter <= 100`	
`#CHAR1 < 'S'`	
`(#Alpha <> 12) AND NOT #Beta`	Comparison and logical expression
`5 + #Alpha`	Arithmetic expression

A condition can use arithmetic expressions:

- The condition of the expression is TRUE if the result is any value other than zero.
- The condition of the expression is FALSE if the result equals zero.

Calling other code blocks from your SCL program

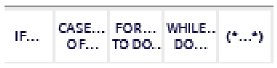
To call another code block in your user program, simply enter the name (or absolute address) of the FB or FC with the parameters. For an FB, you must provide the instance DB to be called with the FB.

`<DB name> (Parameter list)`	Call as a single instance
`<#Instance name> (Parameter list)`	Call as multi-instance
`"MyDB"(MyInput:=10, MyInOut:="Tag1");`	
`<FC name> (Parameter list)`	Standard call
`<Operand>:=<FC name> (Parameter list)`	Call in an expression
`"MyFC"(MyInput:=10, MyInOut:="Tag1");`	

You can also drag blocks from the navigation tree to the SCL program editor, and complete the parameter assignment.

Adding block comments to SCL code

You can include a block comment in your SCL code by including the comment text between (* and *). You can have any number of comment lines between the (* and the *). Your SCL program block can include many block comments. For programming convenience, the SCL editor includes a block comment button along with common control statements:



Addressing

As with LAD and FBD, SCL allows you to use either tags (symbolic addressing) or absolute addresses in your user program. SCL also allows you to use a variable as an array index.

Absolute addressing

`%I0.0` `%MB100`	Precede absolute addresses with the "%" symbol. Without the "%", STEP 7 generates an undefined tag error at compile time.

Symbolic addressing

`"PLC_Tag_1"`	Tag in PLC tag table
`"Data_block_1".Tag_1`	Tag in a data block
`"Data_block_1".MyArray[#i]`	Array element in a data block array

7.5.3.3 Indexed addressing with PEEK and POKE instructions

SCL provides PEEK and POKE instructions that allow you to read from or write to data blocks, I/O, or memory. You provide parameters for specific byte offsets or bit offsets for the operation.

Note

To use the PEEK and POKE instructions with data blocks, you must use standard (not optimized) data blocks. Also note that the PEEK and POKE instructions merely transfer data. They have no knowledge of data types at the addresses.

`PEEK(area:=_in_,`
`dbNumber:=_in_,`
`byteOffset:=_in_);`
	Reads the byte referenced by byteOffset of the referenced data block, I/O or memory area.
	Example referencing data block:
	`%MB100 := PEEK(area:=16#84, dbNumber:=1, byteOffset:=#i);`
	Example referencing IB3 input:
	`%MB100 := PEEK(area:=16#81, dbNumber:=0, byteOffset:=#i); // when #i = 3`

`PEEK_WORD(area:=_in_,`
`dbNumber:=_in_,`
`byteOffset:=_in_);`
	Reads the word referenced by byteOffset of the referenced data block, I/O or memory area.
	Example:
	`%MW200 := PEEK_WORD(area:=16#84, dbNumber:=1, byteOffset:=#i);`

`PEEK_DWORD(area:=_in_,`
`dbNumber:=_in_,`
`byteOffset:=_in_);`
	Reads the double word referenced by byteOffset of the referenced data block, I/O or memory area.
	Example:
	`%MD300 := PEEK_DWORD(area:=16#84, dbNumber:=1, byteOffset:=#i);`

`PEEK_BOOL(area:=_in_,`
`dbNumber:=_in_,`
`byteOffset:=_in_,`
`bitOffset:=_in_);`
	Reads a Boolean referenced by the bitOffset and byteOffset of the referenced data block, I/O or memory area
	Example:
	`%MB100.0 := PEEK_BOOL(area:=16#84, dbNumber:=1, byteOffset:=#ii, bitOffset:=#j);`

```
POKE(area:=_in_,
          dbNumber:=_in_,
          byteOffset:=_in_,
          value:=_in_);
```

Writes the value (Byte, Word, or DWord) to the referenced byteOffset of the referenced data block, I/O or memory area

Example referencing data block:

```
POKE(area:=16#84, dbNumber:=2,
byteOffset:=3, value:="Tag_1");
```

Example referencing QB3 output:

```
POKE(area:=16#82, dbNumber:=0,
byteOffset:=3, value:="Tag_1");
```

```
POKE_BOOL(area:=_in_,
          dbNumber:=_in_,
          byteOffset:=_in_,
          bitOffset:=_in_,
          value:=_in_);
```

Writes the Boolean value to the referenced bitOffset and byteOffset of the referenced data block, I/O or memory area

Example:

```
POKE_BOOL(area:=16#84, dbNumber:=2,
byteOffset:=3, bitOffset:=5, val-
ue:=0);
```

```
POKE_BLK(area_src:=_in_,
          dbNumber_src:=_in_,
          byteOffset_src:=_in_,
          area_dest:=_in_,
          dbNumber_dest:=_in_,
          byteOffset_dest:=_in_,
          count:=_in_);
```

Writes "count" number of bytes starting at the referenced byte Offset of the referenced source data block, I/O or memory area to the referenced byteOffset of the referenced destination data block, I/O or memory area

Example:

```
POKE_BLK(area_src:=16#84,
dbNumber_src:=#src_db, byteOff-
set_src:=#src_byte, area_dest:=16#84,
dbNumber_dest:=#src_db, byteOff-
set_dest:=#src_byte, count:=10);
```

For PEEK and POKE instructions, the following values for the "area", "area_src" and "area_dest" parameters are applicable. For areas other than data blocks, the dbNumber parameter must be 0.

16#81	I
16#82	Q
16#83	M
16#84	DB

7.5.4 EN and ENO for LAD, FBD and SCL

Determining "power flow" (EN and ENO) for an instruction

Certain instructions (such as the Math and the Move instructions) provide parameters for EN and ENO. These parameters relate to power flow in LAD or FBD and determine whether the instruction is executed during that scan. SCL also allows you to set the ENO parameter for a code block.

- EN (Enable In) is a Boolean input. Power flow (EN = 1) must be present at this input for the box instruction to be executed. If the EN input of a LAD box is connected directly to the left power rail, the instruction will always be executed.
- ENO (Enable Out) is a Boolean output. If the box has power flow at the EN input and the box executes its function without error, then the ENO output passes power flow (ENO = 1) to the next element. If an error is detected in the execution of the box instruction, then power flow is terminated (ENO = 0) at the box instruction that generated the error.

Table 7- 3 Operands for EN and ENO

Program editor	Inputs/outputs	Operands	Data type
LAD	EN, ENO	Power flow	Bool
FBD	EN	I, I:P, Q, M, DB, Temp, Power Flow	Bool
	ENO	Power Flow	Bool
SCL	EN[1]	TRUE, FALSE	Bool
	ENO[2]	TRUE, FALSE	Bool

[1] The use of EN is only available for FBs.

[2] The use of ENO with the SCL code block is optional. You must configure the SCL compiler to set ENO when the code block finishes.

Configuring SCL to set ENO

To configure the SCL compiler for setting ENO, follow these steps:

1. Select the "Settings" command from the "Options" menu.
2. Expand the "PLC programming" properties and select "SCL (Structured Control Language)".
3. Select the "Set ENO automatically" option.

Using ENO in program code

You can also use ENO in your program code, for example by assigning ENO to a PLC tag, or by evaluating ENO in a local block.

Examples:

```
"MyFunction"
  ( IN1 := … ,
    IN2 := … ,
    OUT1 => #myOut,
    ENO => #statusFlag ); // PLC tag statusFlag holds the value of E
NO

"MyFunction"
  ( IN1 := …
    IN2 := … ,
    OUT1 => #myOut,
    ENO => ENO );  // block status flag of "MyFunction"
                   // is stored in the local block

IF ENO = TRUE THEN
    // execute code only if MyFunction returns true ENO
```

Effect of Ret_Val or Status parameters on ENO

Some instructions, such as the communication instructions or the string conversion instructions, provide an output parameter that contains information about the processing of the instruction. For example, some instructions provide a Ret_Val (return value) parameter, which is typically an Int data type that contains status information in a range from -32768 to +32767. Other instructions provide a Status parameter, which is typically a Word data type that stores status information in a range of hexadecimal values from 16#0000 to 16#FFFF. The numerical value stored in a Ret_Val or a Status parameter determines the state of ENO for that instruction.

- Ret_Val: A value from 0 to 32767 typically sets ENO = 1 (or TRUE). A value from -32768 to -1 typically sets ENO = 0 (or FALSE). To evaluate Ret_Val, change the representation to hexadecimal.
- Status: A value from 16#0000 16#7FFF typically sets ENO = 1 (or TRUE). A value from 16#8000 to 16#FFFF typically sets ENO = 0 (or FALSE).

Instructions that take more than one scan to execute often provide a Busy parameter (Bool) to signal that the instruction is active but has not completed execution. These instructions often also provide a Done parameter (Bool) and an Error parameter (Bool). Done signals that the instruction was completed without error, and Error signals that the instruction was completed with an error condition.

- When Busy = 1 (or TRUE), ENO = 1 (or TRUE).
- When Done = 1 (or TRUE), ENO = 1 (or TRUE).
- When Error = 1 (or TRUE), ENO = 0 (or FALSE).

See also

OK (Check validity) and NOT_OK (Check invalidity) instructions (Page 234)

7.6 Protection

7.6.1 Access protection for the CPU

The CPU provides four levels of security for restricting access to specific functions. When you configure the security level and password for a CPU, you limit the functions and memory areas that can be accessed without entering a password.

Each level allows certain functions to be accessible without a password. The default condition for the CPU is to have no restriction and no password-protection. To restrict access to a CPU, you configure the properties of the CPU and enter the password.

Entering the password over a network does not compromise the password protection for the CPU. Password protection does not apply to the execution of user program instructions including communication functions. Entering the correct password provides access to all of the functions at that level.

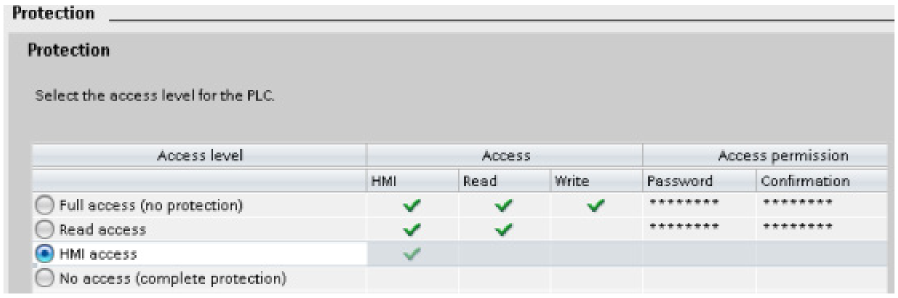
PLC-to-PLC communications (using communication instructions in the code blocks) are not restricted by the security level in the CPU.

Table 7- 4 Security levels for the CPU

Security level	Access restrictions
Full access (no protection)	Allows full access without password protection.
Read access	Allows HMI access and all forms of PLC-to-PLC communications without password protection. Password is required for modifying (writing to) the CPU and for changing the CPU mode (RUN/STOP).
HMI access	Allows HMI access and all forms of PLC-to-PLC communications without password protection. Password is required for reading the data in the CPU, for modifying (writing to) the CPU, and for changing the CPU mode (RUN/STOP).
No access (complete protection)	Allows no access without password protection. Password is required for HMI access, reading the data in the CPU, and for modifying (writing to) the CPU.

Passwords are case-sensitive. To configure the protection level and passwords, follow these steps:

1. In the "Device configuration", select the CPU.
2. In the inspector window, select the "Properties" tab.
3. Select the "Protection" property to select the protection level and to enter passwords.



Protection

Protection

Select the access level for the PLC.

Access level	Access			Access permission	
	HMI	Read	Write	Password	Confirmation
○ Full access (no protection)	✓	✓	✓	********	********
○ Read access	✓	✓		********	********
● HMI access	✓				
○ No access (complete protection)					

When you download this configuration to the CPU, the user has HMI access and can access HMI functions without a password. To read data, the user must enter the configured password for "Read access" or the password for "Full access (no protection)". To write data, the user must enter the configured password for "Full access (no protection)".

WARNING

Unauthorized access to a protected CPU

Users with CPU full access privileges have privileges to read and write PLC variables. Regardless of the access level for the CPU, Web server users can have privileges to read and write PLC variables. Unauthorized access to the CPU or changing PLC variables to invalid values could disrupt process operation and could result in death, severe personal injury and/or property damage.

Authorized users can perform operating mode changes, writes to PLC data, and firmware updates. Siemens recommends that you observe the following security practices:

- Password protect CPU access levels and Web server user IDs (Page 789) with strong passwords. Strong passwords are at least ten characters in length, mix letters, numbers, and special characters, are not words that can be found in a dictionary, and are not names or identifiers that can be derived from personal information. Keep the password secret and change it frequently.
- Enable access to the Web server only with the HTTPS protocol.
- Do not extend the default minimum privileges of the Web server "Everybody" user.
- Perform error-checking and range-checking on your variables in your program logic because Web page users can change PLC variables to invalid values.

Connection mechanisms

To access remote connection partners with PUT/GET instructions, the user must also have permission.

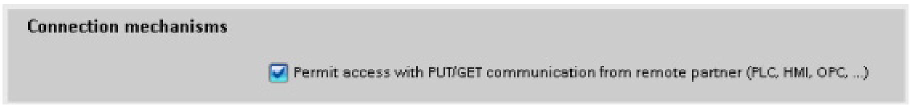
By default, the "Permit access with PUT/GET communication" option is not enabled. In this case, read and write access to CPU data is only possible for communication connections that require configuration or programming both for the local CPU and for the communication partner. Access through BSEND/BRCV instructions is possible, for example.

Connections for which the local CPU is only a server (meaning that no configuration/programming of the communication with the communication partner exists at the local CPU), are therefore not possible during operation of the CPU, for example:

- PUT/GET, FETCH/WRITE or FTP access through communication modules
- PUT/GET access from other S7 CPUs
- HMI access through PUT/GET communication

If you want to allow access to CPU data from the client side, that is, you do not want to restrict the communication services of the CPU, follow these steps:

1. Configure the protection access level to be any level other than "No access (complete protection)".
2. Select the "Permit access with PUT/GET communication" check box.


Connection mechanisms

✓ Permit access with PUT/GET communication from remote partner (PLC, HMI, OPC, ...)

When you download this configuration to the CPU, the CPU permits PUT/GET communication from remote partners

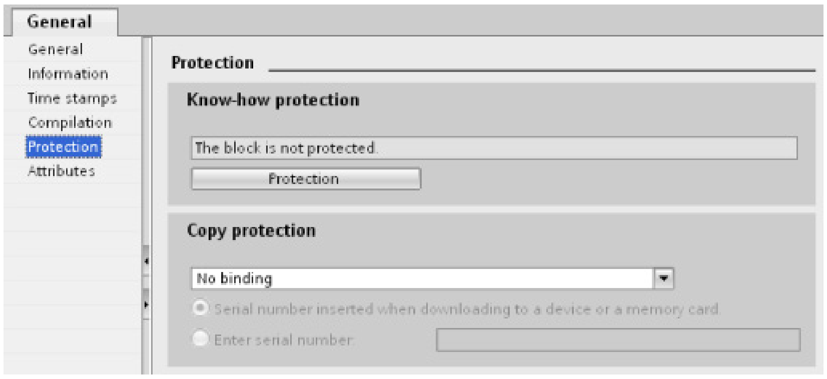
7.6.2 Know-how protection

Know-how protection allows you to prevent one or more code blocks (OB, FB, FC, or DB) in your program from unauthorized access. You create a password to limit access to the code block. The password-protection prevents unauthorized reading or modification of the code block. Without the password, you can read only the following information about the code block:

- Block title, block comment, and block properties
- Transfer parameters (IN, OUT, IN_OUT, Return)
- Call structure of the program
- Global tags in the cross references (without information on the point of use), but local tags are hidden

When you configure a block for "know-how" protection, the code within the block cannot be accessed except after entering the password.

Use the "Properties" task card of the code block to configure the know-how protection for that block. After opening the code block, select "Protection" from Properties.



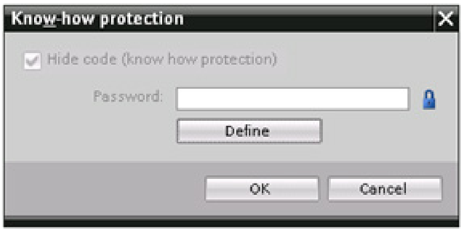
The block is not protected.

No binding

Serial number inserted when downloading to a device or a memory card

Enter serial number:

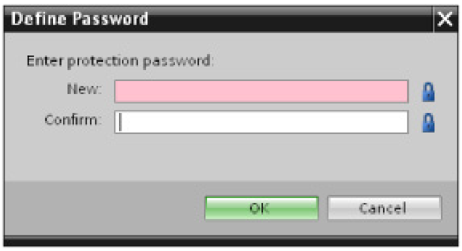
1. In the Properties for the code block, click the "Protection" button to display the "Know-how protection" dialog.
2. Click the "Define" button to enter the password.



Hide code (know how protection)

Password:

After entering and confirming the password, click "OK".



Enter protection password:

New:

Confirm:

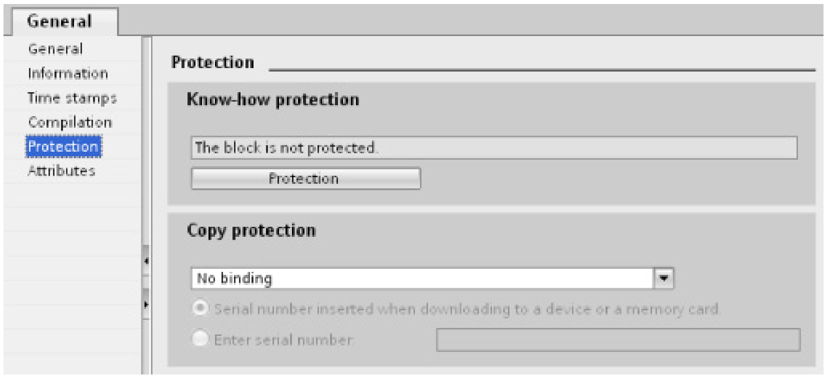
7.6.3 Copy protection

An additional security feature allows you to bind program blocks for use with a specific memory card or CPU. This feature is especially useful for protecting your intellectual property. When you bind a program block to a specific device, you restrict the program or code block for use only with a specific memory card or CPU. This feature allows you to distribute a program or code block electronically (such as over the Internet or through email) or by sending a memory cartridge. Copy protection is available for OBs (Page 176), FBs (Page 178), and FCs (Page 178). The S7-1200 CPU supports three types of block protection:

- Binding to the serial number of a CPU
- Binding to the serial number of a memory card
- Dynamic binding with mandatory password

Use the "Properties" task card of the code block to bind the block to a specific CPU or memory card.

1. After opening the code block, select "Protection".



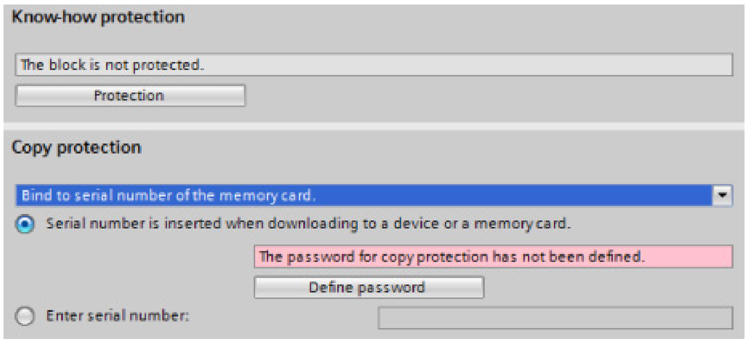
The block is not protected.

No binding

Serial number inserted when downloading to a device or a memory card.

Enter serial number:

2. From the drop-down list under "Copy protection" task, select the type of copy protection that you want to use.



The block is not protected.

Bind to serial number of the memory card.

Serial number is inserted when downloading to a device or a memory card.

The password for copy protection has not been defined.

Enter serial number:

3. For binding to the serial number of a CPU or memory card, select either to insert the serial number when downloading, or enter the serial number for the memory card or CPU.

Note

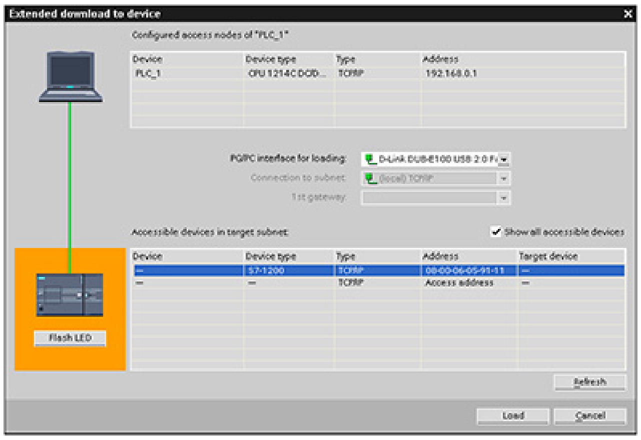
The serial number is case-sensitive.

For dynamic binding with mandatory password, define the password that you must use to download or copy the block.

When you subsequently download (Page 203) a block with dynamic binding, you must enter the password to be able to download the block. Note that the copy protection password and the know-how protection (Page 200) password are two separate passwords.

7.7 Downloading the elements of your program

You can download the elements of your project from the programming device to the CPU. When you download a project, the CPU stores the user program (OBs, FCs, FBs and DBs) in permanent memory.


Extended download to device

Configured access nodes of "PLC_1"

Device	Device type	Type	Address
PLC_1	CPU 1214C DC/D...	TCP/IP	192.168.0.1

PG/PC interface for loading: D-Link DUB-E100 USB 2.0 F...
Connection to subnet: (local) TCP/IP
1st gateway:

Accessible devices in target subnet: ✓ Show all accessible devices

Device	Device type	Type	Address	Target device
—	S7-1200	TCP/IP	00-00-06-05-91-11	—
—	—	TCP/IP	Access address	—

You can download your project from the programming device to your CPU from any of the following locations:

- "Project tree": Right-click the program element, and then click the context-sensitive "Download" selection.
- "Online" menu: Click the "Download to device" selection.
- Toolbar: Click the "Download to device" icon.

Note that if you have applied dynamic binding with mandatory password (Page 201) to any of the program blocks, you must enter the password for the protected blocks in order to download them. If you have configured this type of copy protection for multiple blocks, you must enter the password for each of the protected blocks in order to download them.

Note

Downloading a program does not clear or make any changes to existing values in retentive memory. If you want to clear retentive memory before a download, then reset your CPU to factory settings prior to downloading the program.

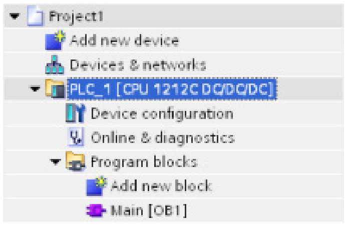
You can also download a panel project for the Basic HMI panels (Page 30) from the TIA Portal to a memory card in the S7-1200 CPU.

7.8 Uploading from the online CPU

You can also copy the program blocks from an online CPU or a memory card attached to your programming device.

Prepare the offline project for the copied program blocks:

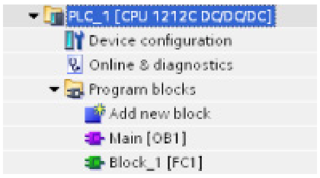
1. Add a CPU device that matches the online CPU.
2. Expand the CPU node once so that the "Program blocks" folder is visible.


Project1
Add new device
Devices & networks
PLC_1 [CPU 1212C DC/DC/DC]
Device configuration
Online & diagnostics
Program blocks
Add new block
Main [OB1]

To upload the program blocks from the online CPU to the offline project, follow these steps:

1. Click the "Program blocks" folder in the offline project.
2. Click the "Go online" button.
3. Click the "Upload" button.
4. Confirm your decision from the Upload dialog (Page 1069).

When the upload is complete, STEP 7 displays all of the uploaded program blocks in the project.


PLC_1 [CPU 1212C DC/DC/DC]
Device configuration
Online & diagnostics
Program blocks
Add new block
Main [OB1]
Block_1 [FC1]

7.8.1 Comparing the online CPU to the offline CPU

You can use the "Compare" editor (Page 1077) in STEP 7 to find differences between the online and offline projects. You might find this useful prior to uploading from the CPU.

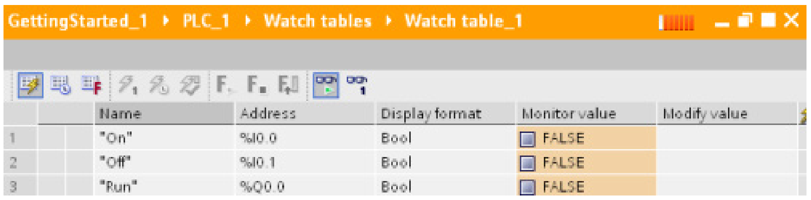
7.9 Debugging and testing the program

7.9.1 Monitor and modify data in the CPU

As shown in the following table, you can monitor and modify values in the online CPU.

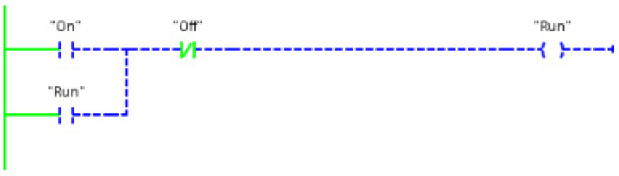
Table 7- 5 Monitoring and modifying data with STEP 7

Editor	Monitor	Modify	Force
Watch table	Yes	Yes	No
Force table	Yes	No	Yes
Program editor	Yes	Yes	No
Tag table	Yes	No	No
DB editor	Yes	No	No


GettingStarted_1 ▸ PLC_1 ▸ Watch tables ▸ Watch table_1

		Name	Address	Display format	Monitor value	Modify value
1		"On"	%I0.0	Bool	FALSE	
2		"Off"	%I0.1	Bool	FALSE	
3		"Run"	%Q0.0	Bool	FALSE	

Monitoring with a watch table



Monitoring with the LAD editor

Refer to the "Online and diagnostics" chapter for more information about monitoring and modifying data in the CPU (Page 1078).

7.9.2 Watch tables and force tables

You use "watch tables" for monitoring and modifying the values of a user program being executed by the online CPU. You can create and save different watch tables in your project to support a variety of test environments. This allows you to reproduce tests during commissioning or for service and maintenance purposes.

With a watch table, you can monitor and interact with the CPU as it executes the user program. You can display or change values not only for the tags of the code blocks and data blocks, but also for the memory areas of the CPU, including the inputs and outputs (I and Q), peripheral inputs (I:P), bit memory (M), and data blocks (DB).

With the watch table, you can enable the physical outputs (Q:P) of a CPU in STOP mode. For example, you can assign specific values to the outputs when testing the wiring for the CPU.

STEP 7 also provides a force table for "forcing" a tag to a specific value. For more information about forcing, see the section on forcing values in the CPU (Page 1085) in the "Online and Diagnostics" chapter.

Note

The force values are stored in the CPU and not in the watch table.

You cannot force an input (or "I" address). However, you can force a peripheral input. To force a peripheral input, append a ":P" to the address (for example: "On:P").

STEP 7 also provides the capability of tracing and recording program variables based on trigger conditions (Page 1097).

7.9.3 Cross reference to show usage

The Inspector window displays cross-reference information about how a selected object is used throughout the complete project, such as the user program, the CPU and any HMI devices. The "Cross-reference" tab displays the instances where a selected object is being used and the other objects using it. The Inspector window also includes blocks which are only available online in the cross-references. To display the cross-references, select the "Show cross-references" command. (In the Project view, find the cross references in the "Tools" menu.)

Note

You do not have to close the editor to see the cross-reference information.

You can sort the entries in the cross-reference. The cross-reference list provides an overview of the use of memory addresses and tags within the user program.

- When creating and changing a program, you retain an overview of the operands, tags and block calls you have used.
- From the cross-references, you can jump directly to the point of use of operands and tags.
- During a program test or when troubleshooting, you are notified about which memory location is being processed by which command in which block, which tag is being used in which screen, and which block is called by which other block.

Table 7- 6 Elements of the cross reference

Column	Description
Object	Name of the object that uses the lower-level objects or that is being used by the lower-level objects
Number	Number of uses
Point of use	Each location of use, for example, network
Property	Special properties of referenced objects, for example, the tag names in multi-instance declarations
as	Shows additional information about the object, such as whether an instance DB is used as template or as a multiple instance
Access	Type of access, whether access to the operand is read access (R) and/or write access (W)
Address	Address of the operand
Type	Information on the type and language used to create the object
Path	Path of object in project tree

Depending on the installed products, the cross-reference table displays additional or different columns.

7.9.4 Call structure to examine the calling hierarchy

The call structure describes the call hierarchy of the block within your user program. It provides an overview of the blocks used, calls to other blocks, the relationships between blocks, the data requirements for each block, and the status of the blocks. You can open the program editor and edit blocks from the call structure.

Displaying the call structure provides you with a list of the blocks used in the user program. STEP 7 highlights the first level of the call structure and displays any blocks that are not called by any other block in the program. The first level of the call structure displays the OBs and any FCs, FBs, and DBs that are not called by an OB. If a code block calls another block, the called block is shown as an indentation under the calling block. The call structure only displays those blocks that are called by a code block.

You can selectively display only the blocks causing conflicts within the call structure. The following conditions cause conflicts:

- Blocks that execute any calls with older or newer code time stamps
- Blocks that call a block with modified interface
- Blocks that use a tag with modified address and/or data type
- Blocks that are called neither directly nor indirectly by an OB
- Blocks that call a non-existent or missing block

You can group several block calls and data blocks as a group. You use a drop-down list to see the links to the various call locations.

You can also perform a consistency check to show time stamp conflicts. Changing the time stamp of a block during or after the program is generated can lead to time stamp conflicts, which in turn cause inconsistencies among the blocks that are calling and being called.

- Most time stamp and interface conflicts can be corrected by recompiling the code blocks.
- If compilation fails to clear up inconsistencies, use the link in the "Details" column to go to the source of the problem in the program editor. You can then manually eliminate any inconsistencies.
- Any blocks marked in red must be recompiled.

Basic instructions 8

8.1 Bit logic operations

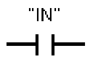
8.1.1 Bit logic instructions

LAD and FBD are very effective for handling Boolean logic. While SCL is especially effective for complex mathematical computation and for project control structures, you can use SCL for Boolean logic.

LAD contacts

Table 8- 1 Normally open and normally closed contacts

LAD	SCL	Description

	`IF in THEN`
	`     Statement;`
	`ELSE`
	`     Statement;`
	`END_IF;`
		Normally open and normally closed contacts: You can connect contacts to other contacts and create your own combination logic. If the input bit you specify uses memory identifier I (input) or Q (output), then the bit value is read from the process-image register. The physical contact signals in your control process are wired to I terminals on the PLC. The CPU scans the wired input signals and continuously updates the corresponding state values in the process-image input register.
		You can perform an immediate read of a physical input using ":P" following the I offset (example: "%I3.4:P"). For an immediate read, the bit data values are read directly from the physical input instead of the process image. An immediate read does not update the process image.

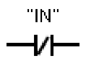
	`IF NOT (in) THEN`
	`     Statement;`
	`ELSE`
	`     Statement;`
	`END_IF;`	

Table 8- 2 Data types for the parameters

Parameter	Data type	Description
IN	Bool	Assigned bit

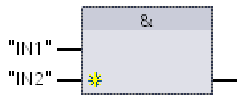
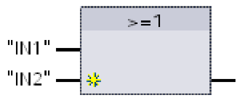
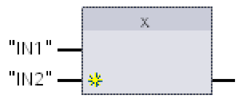
- The Normally Open contact is closed (ON) when the assigned bit value is equal to 1.
- The Normally Closed contact is closed (ON) when the assigned bit value is equal to 0.
- Contacts connected in series create AND logic networks.
- Contacts connected in parallel create OR logic networks.

FBD AND, OR, and XOR boxes

In FBD programming, LAD contact networks are transformed into AND (&), OR (>=1), and EXCLUSIVE OR (x) box networks where you can specify bit values for the box inputs and outputs. You may also connect to other logic boxes and create your own logic combinations. After the box is placed in your network, you can drag the "Insert input" tool from the "Favorites" toolbar or instruction tree and then drop it onto the input side of the box to add more inputs. You can also right-click on the box input connector and select "Insert input".

Box inputs and outputs can be connected to another logic box, or you can enter a bit address or bit symbol name for an unconnected input. When the box instruction is executed, the current input states are applied to the binary box logic and, if true, the box output will be true.

Table 8- 3 AND, OR, and XOR boxes

FBD	SCL[1]	Description
	`out := in1 AND in2;`	All inputs of an AND box must be TRUE for the output to be TRUE.
	`out := in1 OR in2;`	Any input of an OR box must be TRUE for the output to be TRUE.
	`out := in1 XOR in2;`	An odd number of the inputs of an XOR box must be TRUE for the output to be TRUE.

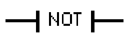
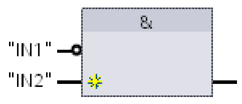
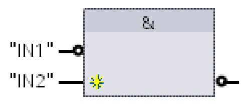
[1] For SCL: You must assign the result of the operation to a variable to be used for another statement.

Table 8- 4 Data types for the parameters

Parameter	Data type	Description
IN1, IN2	Bool	Input bit

NOT logic inverter

Table 8- 5 Invert RLO (Result of Logic Operation)

LAD	FBD	SCL	Description
	 	`NOT`	For FBD programming, you can drag the "Invert RLO" tool from the "Favorites" toolbar or instruction tree and then drop it on an input or output to create a logic inverter on that box connector. The LAD NOT contact inverts the logical state of power flow input. • If there is no power flow into the NOT contact, then there is power flow out. • If there is power flow into the NOT contact, then there is no power flow out.

Output coil and assignment box

The coil output instruction writes a value for an output bit. If the output bit you specify uses memory identifier Q, then the CPU turns the output bit in the process-image register on or off, setting the specified bit equal to power flow status. The output signals for your control actuators are wired to the Q terminals of the CPU. In RUN mode, the CPU system continuously scans your input signals, processes the input states according to your program logic, and then reacts by setting new output state values in the process-image output register. The CPU system transfers the new output state reaction that is stored in the process-image register, to the wired output terminals.

Table 8- 6 Assignment and negate assignment

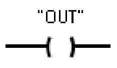
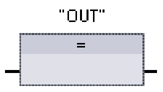
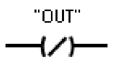
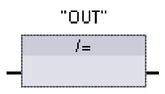
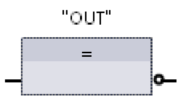
LAD	FBD	SCL	Description
		`out := <Boolean expression>;`	In FBD programming, LAD coils are transformed into assignment (= and /=) boxes where you specify a bit address for the box output. Box inputs and outputs can be connected to other box logic or you can enter a bit address. You can specify an immediate write of a physical output using ":P" following the Q offset (example: "%Q3.4:P"). For an immediate write, the bit data values are written to the process image output and directly to physical output.
	 	`out := NOT <Boolean expression>;`	

Table 8- 7 Data types for the parameters

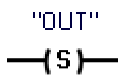
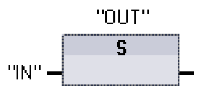
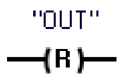
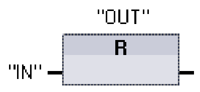
Parameter	Data type	Description
OUT	Bool	Assigned bit

- If there is power flow through an output coil or an FBD "=" box is enabled, then the output bit is set to 1.
- If there is no power flow through an output coil or an FBD "=" assignment box is not enabled, then the output bit is set to 0.
- If there is power flow through an inverted output coil or an FBD "/=" box is enabled, then the output bit is set to 0.
- If there is no power flow through an inverted output coil or an FBD "/=" box is not enabled, then the output bit is set to 1.

8.1.2 Set and reset instructions

Set and Reset 1 bit

Table 8- 8 S and R instructions

LAD	FBD	SCL	Description
		Not available	Set output: When S (Set) is activated, then the data value at the OUT address is set to 1. When S is not activated, OUT is not changed.
		Not available	Reset output: When R (Reset) is activated, then the data value at the OUT address is set to 0. When R is not activated, OUT is not changed.

[1] For LAD and FBD: These instructions can be placed anywhere in the network.

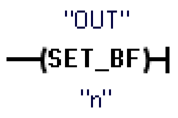
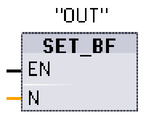
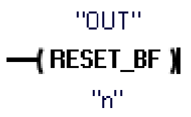
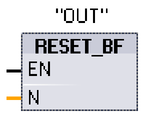
[2] For SCL: You must write code to replicate this function within your application.

Table 8- 9 Data types for the parameters

Parameter	Data type	Description
IN (or connect to contact/gate logic)	Bool	Bit tag of location to be monitored
OUT	Bool	Bit tag of location to be set or reset

Set and Reset Bit Field

Table 8- 10 SET_BF and RESET_BF instructions

LAD[1]	FBD	SCL	Description
		Not available	Set bit field: When SET_BF is activated, a data value of 1 is assigned to "n" bits starting at address tag OUT. When SET_BF is not activated, OUT is not changed.
		Not available	Reset bit field: RESET_BF writes a data value of 0 to "n" bits starting at address tag OUT. When RESET_BF is not activated, OUT is not changed.

1 For LAD and FBD: These instructions must be the right-most instruction in a branch.

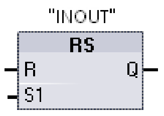
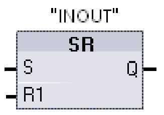
2 For SCL: You must write code to replicate this function within your application.

Table 8- 11 Data types for the parameters

Parameter	Data type	Description
OUT	Bool	Starting element of a bit field to be set or reset (Example: #MyArray[3])
n	Constant (UInt)	Number of bits to write

Set-dominant and Reset-dominant flip-flops

Table 8- 12 RS and SR instructions

LAD / FBD	SCL	Description
	Not available	Reset/set flip-flop: RS is a set dominant latch where the set dominates. If the set (S1) and reset (R) signals are both true, the value at address INOUT will be 1.
	Not available	Set/reset flip-flop: SR is a reset dominant latch where the reset dominates. If the set (S) and reset (R1) signals are both true, the value at address INOUT will be 0.

1 For LAD and FBD: These instructions must be the right-most instruction in a branch.

2 For SCL: You must write code to replicate this function within your application.

Table 8- 13 Data types for the parameters

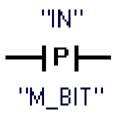
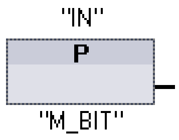
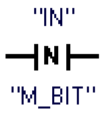
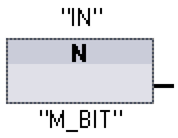
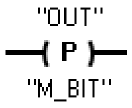
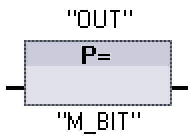
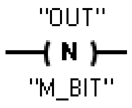
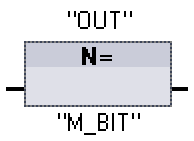
Parameter	Data type	Description
S, S1	Bool	Set input; 1 indicates dominance
R, R1	Bool	Reset input; 1 indicates dominance
INOUT	Bool	Assigned bit tag "INOUT"
Q	Bool	Follows state of "INOUT" bit

The "INOUT" tag assigns the bit address that is set or reset. The optional output Q follows the signal state of the "INOUT" address.

Instruction	**S1**	**R**	**"INOUT" bit**
RS	0	0	Previous state
	0	1	0
	1	0	1
	1	1	1
	S	**R1**	
SR	0	0	Previous state
	0	1	0
	1	0	1
	1	1	0

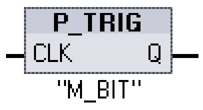
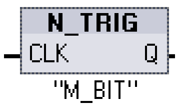
8.1.3 Positive and negative edge instructions

Table 8- 14 Positive and negative transition detection

LAD	FBD	SCL	Description
		Not available [1]	Scan operand for positive signal edge. LAD: The state of this contact is TRUE when a positive transition (OFF-to-ON) is detected on the assigned "IN" bit. The contact logic state is then combined with the power flow in state to set the power flow out state. The P contact can be located anywhere in the network except the end of a branch. FBD: The output logic state is TRUE when a positive transition (OFF-to-ON) is detected on the assigned input bit. The P box can only be located at the beginning of a branch.
		Not available [1]	Scan operand for negative signal edge. LAD: The state of this contact is TRUE when a negative transition (ON-to-OFF) is detected on the assigned input bit. The contact logic state is then combined with the power flow in state to set the power flow out state. The N contact can be located anywhere in the network except the end of a branch. FBD: The output logic state is TRUE when a negative transition (ON-to-OFF) is detected on the assigned input bit. The N box can only be located at the beginning of a branch.
		Not available [1]	Set operand on positve signal edge. LAD: The assigned bit "OUT" is TRUE when a positive transition (OFF-to-ON) is detected on the power flow entering the coil. The power flow in state always passes through the coil as the power flow out state. The P coil can be located anywhere in the network. FBD: The assigned bit "OUT" is TRUE when a positive transition (OFF-to-ON) is detected on the logic state at the box input connection or on the input bit assignment if the box is located at the start of a branch. The input logic state always passes through the box as the output logic state. The P= box can be located anywhere in the branch.
		Not available [1]	Set operand on negative signal edge. LAD: The assigned bit "OUT" is TRUE when a negative transition (ON-to-OFF) is detected on the power flow entering the coil. The power flow in state always passes through the coil as the power flow out state. The N coil can be located anywhere in the network. FBD: The assigned bit "OUT" is TRUE when a negative transition (ON-to-OFF) is detected on the logic state at the box input connection or on the input bit assignment if the box is located at the start of a branch. The input logic state always passes through the box as the output logic state. The N= box can be located anywhere in the branch.

[1] For SCL: You must write code to replicate this function within your application.

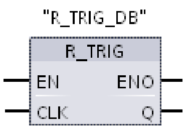
Table 8- 15 P_TRIG and N_TRIG

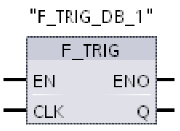
LAD / FBD	SCL	Description
	Not available [1]	Scan RLO (result of logic operation) for positve signal edge. The Q output power flow or logic state is TRUE when a positive transition (OFF-to-ON) is detected on the CLK input state (FBD) or CLK power flow in (LAD). In LAD, the P_TRIG instruction cannot be located at the beginning or end of a network. In FBD, the P_TRIG instruction can be located anywhere except the end of a branch.
	Not available [1]	Scan RLO for negative signal edge. The Q output power flow or logic state is TRUE when a negative transition (ON-to-OFF) is detected on the CLK input state (FBD) or CLK power flow in (LAD). In LAD, the N_TRIG instruction cannot be located at the beginning or end of a network. In FBD, the N_TRIG instruction can be located anywhere except the end of a branch.

[1] For SCL: You must write code to replicate this function within your application.

Table 8- 16 R_TRIG and F_TRIG instructions

LAD / FBD	SCL	Description

	**`"R_TRIG_DB"(`**
	`  CLK:=_in_,`
	`  Q=> _bool_out_);`
		Set tag on positive signal edge.
		The assigned instance DB is used to store the previous state of the CLK input. The Q output power flow or logic state is TRUE when a positive transition (OFF-to-ON) is detected on the CLK input state (FBD) or CLK power flow in (LAD).
		In LAD, the R_TRIG instruction cannot be located at the beginning or end of a network. In FBD, the R_TRIG instruction can be located anywhere except the end of a branch.

	`"F_TRIG_DB"(`
	`  CLK:=_in_,`
	`  Q=> _bool_out_);`
		Set tag on negative signal edge.
		The assigned instance DB is used to store the previous state of the CLK input. The Q output power flow or logic state is TRUE when a negative transition (ON-to-OFF) is detected on the CLK input state (FBD) or CLK power flow in (LAD).
		In LAD, the F_TRIG instruction cannot be located at the beginning or end of a network. In FBD, the F_TRIG instruction can be located anywhere except the end of a branch.

For R_TRIG and F_TRIG, when you insert the instruction in the program, the "Call options" dialog opens automatically. In this dialog you can assign
whether the edge memory bit is stored in its own data block (single instance) or as a local tag (multiple instance) in the
block interface. If you create a separate data block, you will find it in the project tree in the "Program resources" folder
under "Program blocks > System blocks".

Table 8- 17 Data types for the parameters (P and N contacts/coils, P=, N=, P_TRIG and N_TRIG)

Parameter	Data type	Description
M_BIT	Bool	Memory bit in which the previous state of the input is saved
IN	Bool	Input bit whose transition edge is detected
OUT	Bool	Output bit which indicates a transition edge was detected
CLK	Bool	Power flow or input bit whose transition edge is detected
Q	Bool	Output which indicates an edge was detected

All edge instructions use a memory bit (M_BIT: P/N contacts/coils, P_TRIG/N_TRIG) or (instance DB bit: R_TRIG, F_TRIG) to store the previous state of the monitored input signal. An edge is detected by comparing the state of the input with the previous state. If the states indicate a change of the input in the direction of interest, then an edge is reported by writing the output TRUE. Otherwise, the output is written to FALSE.

Note

Edge instructions evaluate the input and memory-bit values each time they are executed, including the first execution. You must account for the initial states of the input and memory bit in your program design either to allow or to avoid edge detection on the first scan.

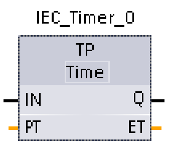
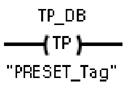
Because the memory bit must be maintained from one execution to the next, you should use a unique bit for each edge instruction, and you should not use this bit any other place in your program. You should also avoid temporary memory and memory that can be affected by other system functions, such as an I/O update. Use only M, global DB, or Static memory (in an instance DB) for M_BIT memory assignments.

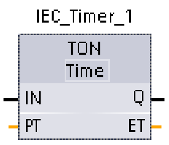
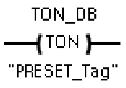
8.2 Timer operations

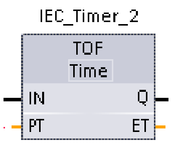
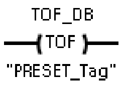
You use the timer instructions to create programmed time delays. The number of timers that you can use in your user program is limited only by the amount of memory in the CPU. Each timer uses a 16 byte IEC_Timer data type DB structure to store timer data that is specified at the top of the box or coil instruction. STEP 7 automatically creates the DB when you insert the instruction.

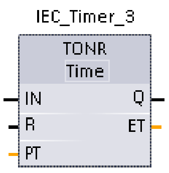
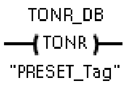
Table 8- 18 Timer instructions

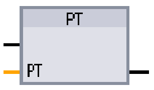
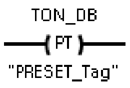
LAD / FBD boxes	LAD coils	SCL	Description

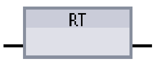
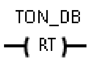
		`"IEC_Timer_0_DB".TP(`
		`IN:=_bool_in_,`
		`PT:=_time_in_,`
		`Q=>_bool_out_,`
		`ET=>_time_out_);`
			The TP timer generates a pulse with a preset width time.

		`"IEC_Timer_0_DB".TON (`
		`IN:=_bool_in_,`
		`PT:=_time_in_,`
		`Q=>_bool_out_,`
		`ET=>_time_out_);`
			The TON timer sets output Q to ON after a preset time delay.

		`"IEC_Timer_0_DB".TOF (`
		`IN:=_bool_in_,`
		`PT:=_time_in_,`
		`Q=>_bool_out_,`
		`ET=>_time_out_);`
			The TOF timer resets output Q to OFF after a preset time delay.

		`"IEC_Timer_0_DB".TONR (`
		`IN:=_bool_in_,`
		`R:=_bool_in_,`
		`PT:=_time_in_,`
		`Q=>_bool_out_,`
		`ET=>_time_out_);`
			The TONR timer sets output Q to ON after a pre-set time delay. Elapsed time is accumulated over multiple timing periods until the R input is used to reset the elapsed time.

FBD only: 		`PRESET_TIMER(`
		`PT:=_time_in_,`
		
		`TIMER:=_iec_timer_in_);`
			The PT (Preset timer) coil loads a new PRESET time value in the specified IEC_Timer.

FBD only: 		`RESET_TIMER(`
		`_iec_timer_in_);`
			The RT (Reset timer) coil resets the specified IEC_Timer.

1 STEP 7 automatically creates the DB when you insert the instruction.

2 In the SCL examples, "IEC_Timer_0_DB" is the name of the instance DB.

Table 8- 19 Data types for the parameters

Parameter	Data type	Description
Box: IN Coil: Power flow	Bool	TP, TON, and TONR: Box: 0=Disable timer, 1=Enable timer Coil: No power flow=Disable timer, Power flow=Enable timer TOF: Box: 0=Enable timer, 1=Disable timer Coil: No power flow=Enable timer, Power flow=Disable timer
R	Bool	TONR box only: 0=No reset 1= Reset elapsed time and Q bit to 0
Box: PT Coil: "PRESET_Tag"	Time	Timer box or coil: Preset time input
Box: Q Coil: DBdata.Q	Bool	Timer box: Q box output or Q bit in the timer DB data Timer coil: you can only address the Q bit in the timer DB data
Box: ET Coil: DBdata.ET	Time	Timer box: ET (elapsed time) box output or ET time value in the timer DB data Timer coil: you can only address the ET time value in the timer DB data.

Table 8- 20 Effect of value changes in the PT and IN parameters

Timer	Changes in the PT and IN box parameters and the corresponding coil parameters
TP	• Changing PT has no effect while the timer runs. • Changing IN has no effect while the timer runs.
TON	• Changing PT has no effect while the timer runs. • Changing IN to FALSE, while the timer runs, resets and stops the timer.
TOF	• Changing PT has no effect while the timer runs. • Changing IN to TRUE, while the timer runs, resets and stops the timer.
TONR	• Changing PT has no effect while the timer runs, but has an effect when the timer resumes. • Changing IN to FALSE, while the timer runs, stops the timer but does not reset the timer. Changing IN back to TRUE will cause the timer to start timing from the accumulated time value.

PT (preset time) and ET (elapsed time) values are stored in the specified IEC_TIMER DB data as signed double integers that represent milliseconds of time. TIME data uses the T# identifier and can be entered as a simple time unit (T#200ms or 200) and as compound time units like T#2s_200ms.

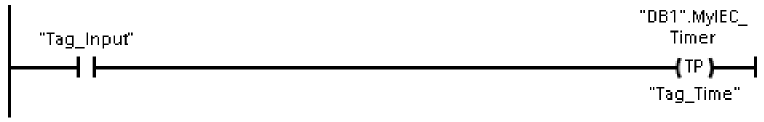
Table 8- 21 Size and range of the TIME data type

Data type	Size	Valid number ranges[1]
TIME	32 bits, stored as DInt data	T#-24d_20h_31m_23s_648ms to T#24d_20h_31m_23s_647ms Stored as -2,147,483,648 ms to +2,147,483,647 ms

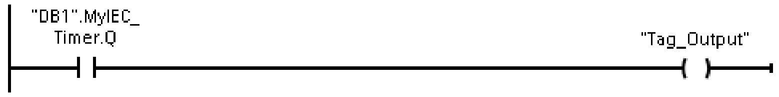
[1] The negative range of the TIME data type shown above cannot be used with the timer instructions. Negative PT (preset time) values are set to zero when the timer instruction is executed. ET (elapsed time) is always a positive value.

Timer coil example

The -(TP)-, -(TON)-, -(TOF)-, and -(TONR)- timer coils must be the last instruction in a LAD network. As shown in the timer example, a contact instruction in a subsequent network evaluates the Q bit in a timer coil's IEC_Timer DB data. Likewise, you must address the ELAPSED element in the IEC_timer DB data if you want to use the elapsed time value in your program.



The pulse timer is started on a 0 to 1 transition of the Tag_Input bit value. The timer runs for the time specified by Tag_Time time value.



As long as the timer runs, the state of DB1.MyIEC_Timer.Q=1 and the Tag_Output value=1. When the Tag_Time value has elapsed, then DB1.MyIEC_Timer.Q=0 and the Tag_Output value=0.

Reset timer -(RT)- and Preset timer -(PT)- coils

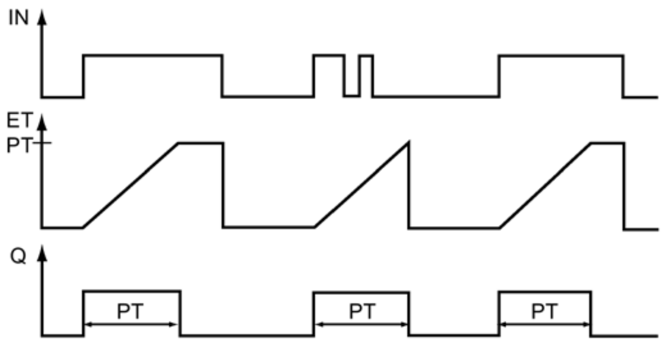
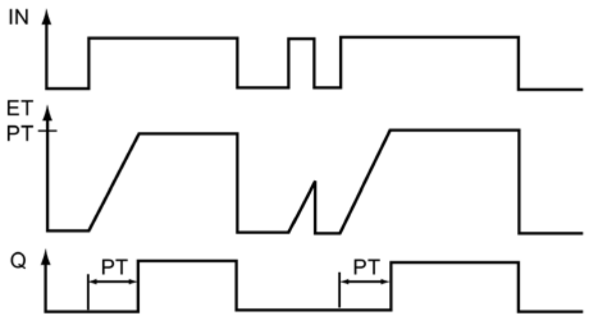
These coil instructions can be used with box or coil timers and can be placed in a mid-line position. The coil output power flow status is always the same as the coil input status. When the -(RT)- coil is activated, the ELAPSED time element of the specified IEC_Timer DB data is reset to 0. When the -(PT)- coil is activated, the PRESET time element of the specified IEC_Timer DB data is loaded with the assigned time-duration value..

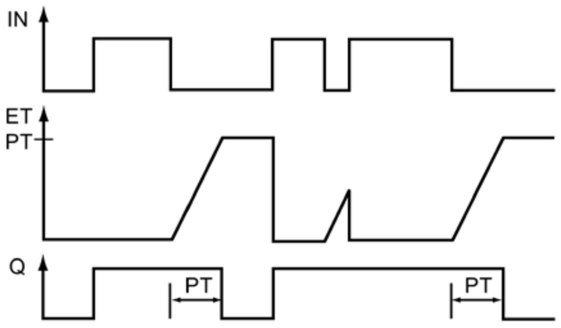
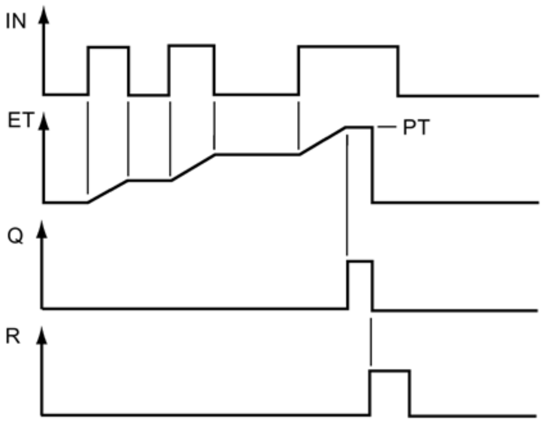
Note

When you place timer instructions in an FB, you can select the "Multi-instance data block" option. The timer structure names can be different with separate data structures, but the timer data is contained in a single data block and does not require a separate data block for each timer. This reduces the processing time and data storage necessary for handling the timers. There is no interaction between the timer data structures in the shared multi-instance DB.

Operation of the timers

Table 8- 22 Types of IEC timers

Timer	Timing diagram
TP: Generate pulse The TP timer generates a pulse with a preset width time.	
TON: Generate ON-delay The TON timer sets output Q to ON after a preset time delay.	

Timer	Timing diagram
TOF: Generate OFF-delay The TOF timer resets output Q to OFF after a preset time delay.	
TONR: Time accumulator The TONR timer sets output Q to ON after a preset time delay. Elapsed time is accumulated over multiple timing periods until the R input is used to reset the elapsed time.	

Note

In the CPU, no dedicated resource is allocated to any specific timer instruction. Instead, each timer utilizes its own timer structure in DB memory and a continuously-running internal CPU timer to perform timing.

When a timer is started due to an edge change on the input of a TP, TON, TOF, or TONR instruction, the value of the continuously-running internal CPU timer is copied into the START member of the DB structure allocated for this timer instruction. This start value remains unchanged while the timer continues to run, and is used later each time the timer is updated. Each time the timer is started, a new start value is loaded into the timer structure from the internal CPU timer.

When a timer is updated, the start value described above is subtracted from the current value of the internal CPU timer to determine the elapsed time. The elapsed time is then compared with the preset to determine the state of the timer Q bit. The ELAPSED and Q members are then updated in the DB structure allocated for this timer. Note that the elapsed time is clamped at the preset value (the timer does not continue to accumulate elapsed time after the preset is reached).

A timer update is performed when and only when:

- A timer instruction (TP, TON, TOF, or TONR) is executed
- The "ELAPSED" member of the timer structure in DB is referenced directly by an instruction
- The "Q" member of the timer structure in DB is referenced directly by an instruction

Timer programming

The following consequences of timer operation should be considered when planning and creating your user program:

- You can have multiple updates of a timer in the same scan. The timer is updated each time the timer instruction (TP, TON, TOF, TONR) is executed and each time the ELAPSED or Q member of the timer structure is used as a parameter of another executed instruction. This is an advantage if you want the latest time data (essentially an immediate read of the timer). However, if you desire to have consistent values throughout a program scan, then place your timer instruction prior to all other instructions that need these values, and use tags from the Q and ET outputs of the timer instruction instead of the ELAPSED and Q members of the timer DB structure.
- You can have scans during which no update of a timer occurs. It is possible to start your timer in a function, and then cease to call that function again for one or more scans. If no other instructions are executed which reference the ELAPSED or Q members of the timer structure, then the timer will not be updated. A new update will not occur until either the timer instruction is executed again or some other instruction is executed using ELAPSED or Q from the timer structure as a parameter.
- Although not typical, you can assign the same DB timer structure to multiple timer instructions. In general, to avoid unexpected interaction, you should only use one timer instruction (TP, TON, TOF, TONR) per DB timer structure.
- Self-resetting timers are useful to trigger actions that need to occur periodically. Typically, self-resetting timers are created by placing a normally-closed contact which references the timer bit in front of the timer instruction. This timer network is typically located above one or more dependent networks that use the timer bit to trigger actions. When the timer expires (elapsed time reaches preset value), the timer bit is ON for one scan, allowing the dependent network logic controlled by the timer bit to execute. Upon the next execution of the timer network, the normally closed contact is OFF, thus resetting the timer and clearing the timer bit. The next scan, the normally closed contact is ON, thus restarting the timer. When creating self-resetting timers such as this, do not use the "Q" member of the timer DB structure as the parameter for the normally-closed contact in front of the timer instruction. Instead, use the tag connected to the "Q" output of the timer instruction for this purpose. The reason to avoid accessing the Q member of the timer DB structure is because this causes an update to the timer and if the timer is updated due to the normally closed contact, then the contact will reset the timer instruction immediately. The Q output of the timer instruction will not be ON for the one scan and the dependent networks will not execute.

Time data retention after a RUN-STOP-RUN transition or a CPU power cycle

If a run mode session is ended with stop mode or a CPU power cycle and a new run mode session is started, then the timer data stored in the previous run mode session is lost, unless the timer data structure is specified as retentive (TP, TON, TOF, and TONR timers).

When you accept the defaults in the call options dialog after you place a timer instruction in the program editor, you are automatically assigned an instance DB which **cannot be made retentive**. To make your timer data retentive, you must either use a global DB or a Multi-instance DB.

Assign a global DB to store timer data as retentive data

This option works regardless of where the timer is placed (OB, FC, or FB).

1. Create a global DB:
   - Double-click "Add new block" from the Project tree
   - Click the data block (DB) icon
   - For the Type, choose global DB
   - If you want to be able to select individual data elements in this DB as retentive, be sure the DB type "Optimized" box is checked. The other DB type option "Standard - compatible with S7-300/400" only allows setting all DB data elements retentive or none retentive.
   - Click OK
2. Add timer structure(s) to the DB:
   - In the new global DB, add a new static tag using data type IEC_Timer.
   - In the "Retain" column, check the box so that this structure will be retentive.
   - Repeat this process to create structures for all the timers that you want to store in this DB. You can either place each timer structure in a unique global DB, or you can place multiple timer structures into the same global DB. You can also place other static tags besides timers in this global DB. Placing multiple timer structures into the same global DB allows you to reduce your overall number of blocks.
   - Rename the timer structures if desired.
3. Open the program block for editing where you want to place a retentive timer (OB, FC, or FB).
4. Place the timer instruction at the desired location.
5. When the call options dialog appears, click the cancel button.
6. On the top of the new timer instruction, type the name (do not use the helper to browse) of the global DB and timer structure that you created above (example: "Data_block_3.Static_1").

Assign a multi-instance DB to store timer data as retentive data

This option only works if you place the timer in an FB.

This option depends upon whether the FB properties specify "Optimized block access" (allows symbolic access only). To verify how the access attribute is configured for an existing FB, right-click on the FB in the Project tree, choose properties, and then choose Attributes.

If the FB specifies "Optimized block access" (allows symbolic access only):

1. Open the FB for edit.
2. Place the timer instruction at the desired location in the FB.
3. When the Call options dialog appears, click the Multi instance icon. The Multi Instance option is only available if the instruction is being placed into an FB.
4. In the Call options dialog, rename the timer if desired.
5. Click OK. The timer instruction appears in the editor, and the IEC_TIMER structure appears in the FB Interface under Static.
6. If necessary, open the FB interface editor (may have to click on the small arrow to expand the view).
7. Under Static, locate the timer structure that was just created for you.
8. In the Retain column for this timer structure, change the selection to "Retain". Whenever this FB is called later from another program block, an instance DB will be created with this interface definition which contains the timer structure marked as retentive.

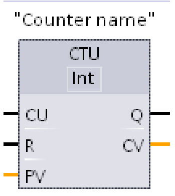
If the FB does not specify "Optimized block access", then the block access type is standard, which is compatible with S7-300/400 classic configurations and allows symbolic and direct access. To assign a multi-instance to a standard block access FB, follow these steps:

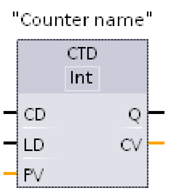
1. Open the FB for edit.
2. Place the timer instruction at the desired location in the FB.
3. When the Call options dialog appears, click on the multi instance icon. The multi instance option is only available if the instruction is being placed into an FB.
4. In the Call options dialog, rename the timer if desired.
5. Click OK. The timer instruction appears in the editor, and the IEC_TIMER structure appears in the FB Interface under Static.
6. Open the block that will use this FB.
7. Place this FB at the desired location. Doing so results in the creation of an instance data block for this FB.
8. Open the instance data block created when you placed the FB in the editor.
9. Under Static, locate the timer structure of interest. In the Retain column for this timer structure, check the box to make this structure retentive.

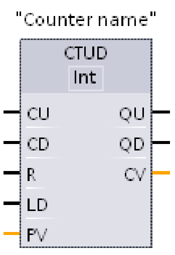
8.3 Counter operations

Table 8- 23 Counter instructions

LAD / FBD	SCL	Description

	`"IEC_Counter_0_DB".CTU (`
	`CU:=_bool_in,`
	`R:=_bool_in,`
	`PV:=_in,`
	`Q=>_bool_out,`
	`CV=>_out);`
		Use the counter instructions to count internal program events and external process events. Each counter uses a structure stored in a data block to maintain counter data. You assign the data block when the counter instruction is placed in the editor.
		• CTU is a count-up counter
		• CTD is a count-down counter
		• CTUD is a count-up-and-down counter

	`"IEC_Counter_0_DB".CTD (`
	`CD:=_bool_in,`
	`LD:=_bool_in,`
	`PV:=_in,`
	`Q=>_bool_out,`
	`CV=>_out);`	

 "Counter name"
CTUD
Int
CU
QU
CD
QD
R
CV
LD
PV
	`"IEC_Counter_0_DB".CTUD(`
	`CU:=_bool_in,`
	`CD:=_bool_in,`
	`R:=_bool_in,`
	`LD:=_bool_in,`
	`PV:=_in_,`
	`QU=>_bool_out,`
	`QD=>_bool_out,`
	`CV=>_out_);`	

1 For LAD and FBD: Select the count value data type from the drop-down list below the instruction name.

2 STEP 7 automatically creates the DB when you insert the instruction.

3 In the SCL examples, "IEC_Counter_0_DB" is the name of the instance DB.

Table 8- 24 Data types for the parameters

Parameter	Data type[1]	Description
CU, CD	Bool	Count up or count down, by one count
R (CTU, CTUD)	Bool	Reset count value to zero
LD (CTD, CTUD)	Bool	Load control for preset value
PV	SInt, Int, DInt, USInt, UInt, UDInt	Preset count value
Q, QU	Bool	True if CV >= PV
QD	Bool	True if CV <= 0
CV	SInt, Int, DInt, USInt, UInt, UDInt	Current count value

1 The numerical range of count values depends on the data type you select. If the count value is an unsigned integer type, you can count down to zero or count up to the range limit. If the count value is a signed integer, you can count down to the negative integer limit and count up to the positive integer limit.

The number of counters that you can use in your user program is limited only by the amount of memory in the CPU. Counters use the following amount of memory:

- For SInt or USInt data types, the counter instruction uses 3 bytes.
- For Int or UInt data types, the counter instruction uses 6 bytes.
- For DInt or UDInt data types, the counter instruction uses 12 bytes.

These instructions use software counters whose maximum counting rate is limited by the execution rate of the OB in which they are placed. The OB that the instructions are placed in must be executed often enough to detect all transitions of the CU or CD inputs. For faster counting operations, see the CTRL_HSC instruction (Page 457).

Note

When you place counter instructions in an FB, you can select the multi-instance DB option, the counter structure names can be different with separate data structures, but the counter data is contained in a single DB and does not require a separate DB for each counter. This reduces the processing time and data storage necessary for the counters. There is no interaction between the counter data structures in the shared multi-instance DB.

Operation of the counters

Table 8- 25 Operation of CTU (count up)

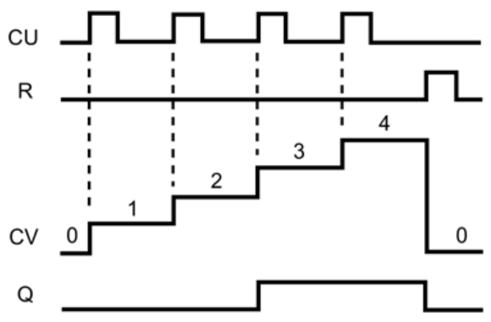
Counter	Operation
The CTU counter counts up by 1 when the value of parameter CU changes from 0 to 1. The CTU timing diagram shows the operation for an unsigned integer count value (where PV = 3). • If the value of parameter CV (current count value) is greater than or equal to the value of parameter PV (preset count value), then the counter output parameter Q = 1. • If the value of the reset parameter R changes from 0 to 1, then the current count value is reset to 0.	

Table 8- 26 Operation of CTD (count down)

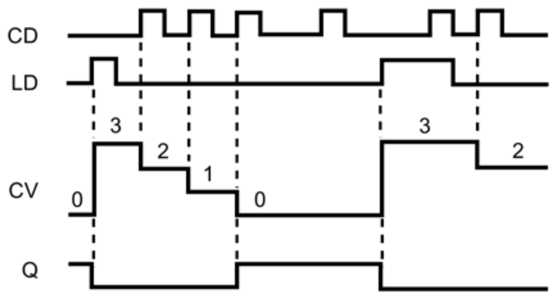
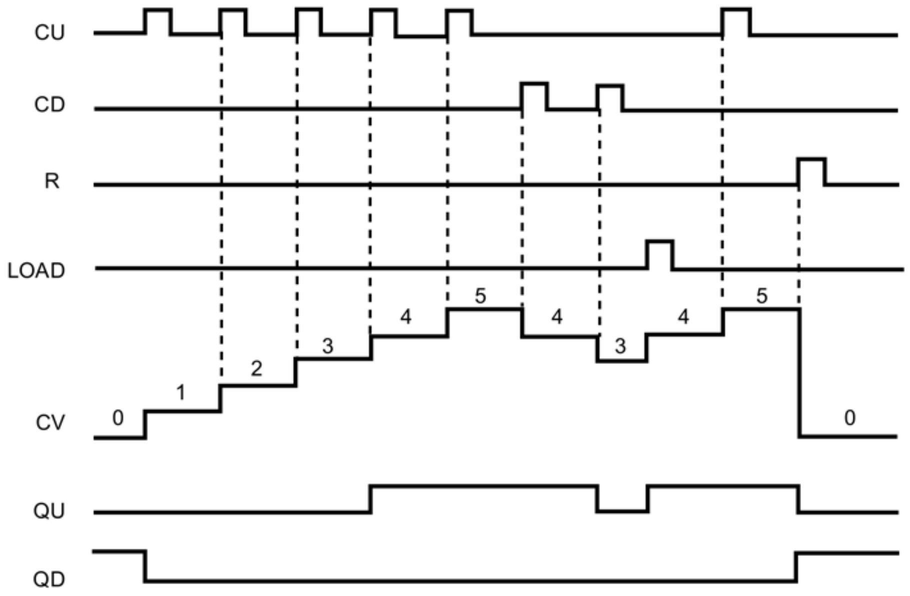
Counter	Operation
The CTD counter counts down by 1 when the value of parameter CD changes from 0 to 1. The CTD timing diagram shows the operation for an unsigned integer count value (where PV = 3). • If the value of parameter CV (current count value) is equal to or less than 0, the counter output parameter Q = 1. • If the value of parameter LOAD changes from 0 to 1, the value at parameter PV (preset value) is loaded to the counter as the new CV (current count value).	

Table 8- 27 Operation of CTUD (count up and down)

Counter	Operation
The CTUD counter counts up or down by 1 on the 0 to 1 transition of the count up (CU) or count down (CD) inputs. The CTUD timing diagram shows the operation for an unsigned integer count value (where PV = 4). • If the value of parameter CV is equal to or greater than the value of parameter PV, then the counter output parameter QU = 1. • If the value of parameter CV is less than or equal to zero, then the counter output parameter QD = 1. • If the value of parameter LOAD changes from 0 to 1, then the value at parameter PV is loaded to the counter as the new CV. • If the value of the reset parameter R is changes from 0 to 1, the current count value is reset to 0.	 CU CD R LOAD CV QU QD 0 1 2 3 4 5 4 3 4 5 0

Counter data retention after a RUN-STOP-RUN transition or a CPU power cycle

If a run mode session is ended with stop mode or a CPU power cycle and a new run mode session is started, then the counter data stored in the previous run mode session is lost, unless the counter data structure is specified as retentive (CTU, CTD, and CTUD counters).

When you accept the defaults in the call options dialog after you place a counter instruction in the program editor, you are automatically assigned an instance DB which **cannot be made retentive.** To make your counter data retentive, you must either use a global DB or a Multi-instance DB.

Assign a global DB to store counter data as retentive data

This option works regardless of where the counter is placed (OB, FC, or FB).

1. Create a global DB:
   – Double-click "Add new block" from the Project tree
   – Click the data block (DB) icon
   – For the Type, choose global DB
   – If you want to be able to select individual items in this DB as retentive, be sure the symbolic-access-only box is checked.
   – Click OK
2. Add counter structure(s) to the DB:
   – In the new global DB, add a new static tag using one of the counter data types. Be sure to consider the Type you want to use for your Preset and Count values.
   – In the "Retain" column, check the box so that this structure will be retentive.
   – Repeat this process to create structures for all the counters that you want to store in this DB. You can either place each counter structure in a unique global DB, or you can place multiple counter structures into the same global DB. You can also place other static tags besides counters in this global DB. Placing multiple counter structures into the same global DB allows you to reduce your overall number of blocks.
   – Rename the counter structures if desired.
3. Open the program block for editing where you want to place a retentive counter (OB, FC, or FB).
4. Place the counter instruction at the desired location.
5. When the call options dialog appears, click the cancel button. You should now see a new counter instruction which has "???" both just above and just below the instruction name.
6. On the top of the new counter instruction, type the name (do not use the helper to browse) of the global DB and counter structure that you created above (example: "Data_block_3.Static_1"). This causes the corresponding preset and count value type to be filled in (example: UInt for an IEC_UCounter structure).

Counter Data Type	Corresponding Type for the Preset and Count Values
IEC_Counter	INT
IEC_SCounter	SINT

IEC_DCounter	DINT
IEC_UCounter	UINT
IEC_USCounter	USINT
IEC_UDCounter	UDINT

Assign a multi-instance DB to store counter data as retentive data

This option only works if you place the counter in an FB.

This option depends upon whether the FB properties specify "Optimized block access" (allows symbolic access only). To verify how the access attribute is configured for an existing FB, right-click on the FB in the Project tree, choose properties, and then choose Attributes.

If the FB specifies "Optimized block access" (allows symbolic access only):

1. Open the FB for edit.
2. Place the counter instruction at the desired location in the FB.
3. When the Call options dialog appears, click on the Multi instance icon. The Multi Instance option is only available if the instruction is being placed into an FB.
4. In the Call options dialog, rename the counter if desired.
5. Click OK. The counter instruction appears in the editor with type INT for the preset and count values, and the IEC_COUNTER structure appears in the FB Interface under Static.
6. If desired, change the type in the counter instruction from INT to one of the other types. The counter structure will change correspondingly.
7. If necessary, open the FB interface editor (may have to click on the small arrow to expand the view).
8. Under Static, locate the counter structure that was just created for you.
9. In the Retain column for this counter structure, change the selection to "Retain". Whenever this FB is called later from another program block, an instance DB will be created with this interface definition which contains the counter structure marked as retentive.

If the FB does not specify "Optimized block access", then the block access type is standard, which is compatible with S7-300/400 classic configurations and allows symbolic and direct access. To assign a multi-instance to a standard block access FB, follow these steps:

1. Open the FB for edit.
2. Place the counter instruction at the desired location in the FB.
3. When the Call options dialog appears, click on the multi instance icon. The multi instance option is only available if the instruction is being placed into an FB.
4. In the Call options dialog, rename the counter if desired.
5. Click OK. The counter instruction appears in the editor with type INT for the preset and count value, and the IEC_COUNTER structure appears in the FB Interface under Static.
6. If desired, change the type in the counter instruction from INT to one of the other types. The counter structure will change correspondingly.
7. Open the block that will use this FB.

8. Place this FB at the desired location. Doing so results in the creation of an instance data block for this FB.
9. Open the instance data block created when you placed the FB in the editor.
10. Under Static, locate the counter structure of interest. In the Retain column for this counter structure, check the box to make this structure retentive.

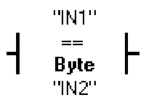
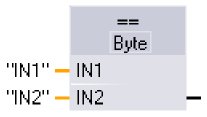
Type shown in counter instruction (for preset and count values)	Corresponding structure Type shown in FB interface
INT	IEC_Counter
SINT	IEC_SCounter
DINT	IEC_DCounter
UINT	IEC_UCounter
USINT	IEC_USCounter
UDINT	IEC_UDCounter

8.4 Comparator operations

8.4.1 Compare values instructions

Table 8- 28 Compare instructions

LAD	FBD	SCL	Description

		`out := in1 = in2;`
		or
		`IF in1 = in2`
		`    THEN out := 1;`
		`    ELSE out := 0;`
		`    END_IF;`
			Compares two values of the same data type. When the LAD contact comparison is TRUE, then the contact is activated. When the FBD box comparison is TRUE, then the box output is TRUE.

[1] For LAD and FBD: Click the instruction name (such as "==") to change the comparison type from the drop-down list. Click the "???" and select data type from the drop-down list.

Table 8- 29 Data types for the parameters

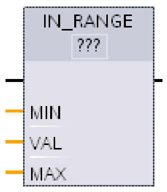
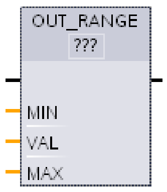
Parameter	Data type	Description
IN1, IN2	Byte, Word, DWord, SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, String, ,WString, Char, Char, Time, Date, TOD, DTL, Constant	Values to compare

Table 8- 30 Comparison descriptions

Relation type	The comparison is true if ...
=	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

8.4.2 IN_Range (Value within range) and OUT_Range (Value outside range) instructions

Table 8- 31 Value within Range and value outside range instructions

LAD / FBD	SCL	Description
	`out := IN_RANGE(min, val, max);`	Tests whether an input value is in or out of a specified value range. If the comparison is TRUE, then the box output is TRUE.
	`out := OUT_RANGE(min, val, max);`	

1 For LAD and FBD: Click the "???" and select the data type from the drop-down list.

Table 8- 32 Data types for the parameters

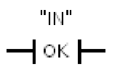
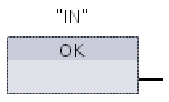
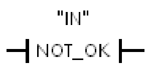
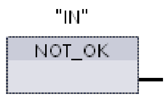
Parameter	Data type[1]	Description
MIN, VAL, MAX	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Constant	Comparator inputs

1 The input parameters MIN, VAL, and MAX must be the same data type.

- The IN_RANGE comparison is true if: MIN <= VAL <= MAX
- The OUT_RANGE comparison is true if: VAL < MIN or VAL > MAX

8.4.3 OK (Check validity) and NOT_OK (Check invalidity) instructions

Table 8- 33 OK (check validity) and Not OK (check invalidity) instructions

LAD	FBD	SCL	Description
		Not available	Tests whether an input data reference is a valid real number according to IEEE specification 754.
		Not available	

1 For LAD and FBD: When the LAD contact is TRUE, the contact is activated and passes power flow. When the FBD box is TRUE, then the box output is TRUE.

Table 8- 34 Data types for the parameter

Parameter	Data type	Description
IN	Real, LReal	Input data

Table 8- 35 Operation

Instruction	The Real number test is TRUE if:
OK	The input value is a valid real number [1]
NOT_OK	The input value is not a valid real number [1]

1 A Real or LReal value is invalid if it is +/- INF (infinity), NaN (Not a Number), or if it is a denormalized value. A denormalized value is a number very close to zero. The CPU substitutes a zero for a denormalized value in calculations.

8.4.4 Variant and array comparison instructions

8.4.4.1 Equality and non-equality comparison instructions

The S7-1200 CPU provides instructions for querying the data type of a tag to which a Variant operand points for either equality or non-equality to the data type of the other operand.

In addition, the S7-1200 CPU provides instructions for querying the data type of an array element for either equality or non-equality to the data type of the other operand.

In these instructions, you are comparing <Operand1> to <Operand2>. <Operand1> must have the Variant data type. <Operand2> can be an elementary data type of a PLC data type. In LAD and FBD, <Operand1> is the operand above the instruction. In LAD, <Operand2> is the operand below the instruction.

For all instructions, the result of logic operation (RLO) is 1 (true) if the equality or non-equality test passes, and is 0 (false) if not.

The equality and non-equality type comparison instructions are as follows:

Table 8- 36 EQ_Type (Compare data type for EQUAL with the data type of a tag)
NE_Type (Compare data type for UNEQUAL with the data type of a tag)
EQ_ElemType (Compare data type of an ARRAY element for EQUAL with the data type of a tag)
NE_ElemType (Compare data type of an ARRAY element for UNEQUAL with the data type of a tag) instructions

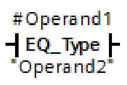
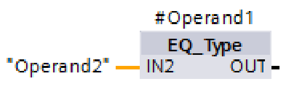
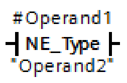
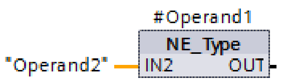
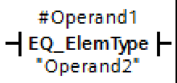
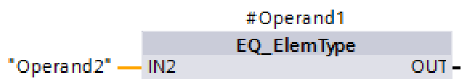
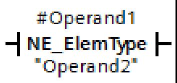
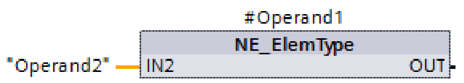
LAD	FBD	SCL	Description
		Not available	Tests whether the tag pointed to by the Variant at Operand1 is of the same data type as the tag at Operand2.
		Not available	Tests whether the tag pointed to by the Variant at Operand1 is of a different data type as the tag at Operand2.
		Not available	Tests whether the array element pointed to by the Variant at Operand1 is of the same data type as the tag at Operand2.
		Not available	Tests whether the array element pointed to by the Variant at Operand1 is of a different data type as the tag at Operand2.

Table 8- 37 Data types for the parameters

Parameter	Data type	Description
Operand1	Variant	First operand
Operand2	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY, PLC data types	Second operand

8.4.4.2 Null comparsion instructions

You can use the instructions IS_NULL and NOT_NULL to determine whether or not the input actually points to an object or not.

For both instructions, <Operand> must have the Variant data type.

The null comparison instructions are as follows:

Table 8- 38 IS_NULL (Query for EQUALS ZERO pointer)
NOT_NULL (Query for EQUALS ZERO pointer) instructions

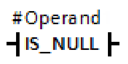
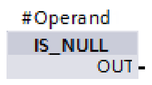
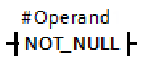
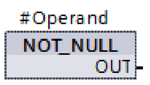
LAD	FBD	SCL	Description
		Not available	Tests whether the tag pointed to by the Variant at Operand is null and therefore not an object.
		Not available	Tests whether the tag pointed to by the Variant at Operand is not null and therefore does point to an object.

Table 8- 39 Data types for the parameters

Parameter	Data type	Description
Operand	Variant	Operand to evaluate for null or not null.

8.4.4.3 IS_ARRAY (Check for ARRAY)

You can use the "Check for ARRAY" instruction to query whether the Variant points to a tag of the Array data type.

The <Operand> must have the Variant data type.

The instructions returns 1 (true) if the operand is an array.

Table 8- 40 IS_ARRAY (Check for ARRAY)

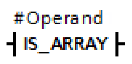
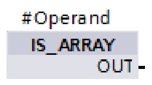
LAD	FBD	SCL	Description
		IS_ARRAY(_variant_in_)	Tests whether the tag pointed to by the Variant at Operand is an array.

Table 8- 41 Data types for the parameters

Parameter	Data type	Description
Operand	Variant	Operand to evaluate for whether it is an array.

8.5 Math functions

8.5.1 CALCULATE (Calculate) instruction

Table 8- 42 CALCULATE instruction

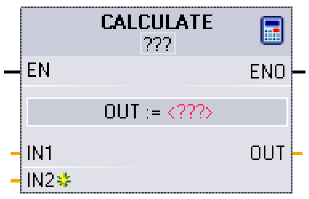
LAD / FBD	SCL	Description
 CALCULATE ??? EN ENO OUT := <???> IN1 OUT IN2	Use the standard SCL math expressions to create the equation.	The CALCULATE instruction lets you create a math function that operates on inputs (IN1, IN2, .. INn) and produces the result at OUT, according to the equation that you define. • Select a data type first. All inputs and the output must be the same data type. • To add another input, click the icon at the last input.

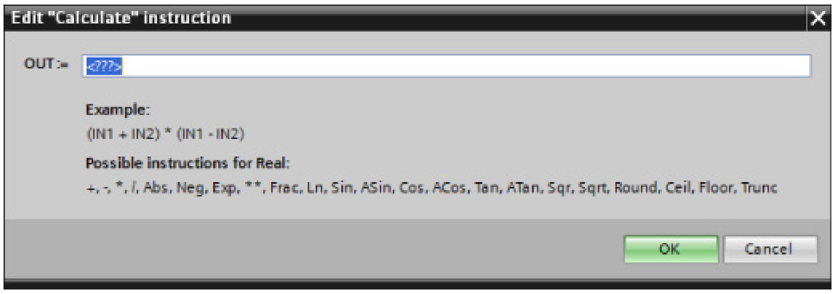
Table 8- 43 Data types for the parameters

Parameter	Data type[1]
IN1, IN2, ..INn	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord

[1] The IN and OUT parameters must be the same data type (with implicit conversions of the input parameters). For example: A SINT value for an input would be converted to an INT or a REAL value if OUT is an INT or REAL

Click the calculator icon to open the dialog and define your math function. You enter your equation as inputs (such as IN1 and IN2) and operations. When you click "OK" to save the function, the dialog automatically creates the inputs for the CALCULATE instruction.

The dialog shows an example and a list of possible instructions that you can include based on the data type of the OUT parameter:



Edit "Calculate" instruction

OUT := <???>

Example:
(IN1 + IN2) * (IN1 - IN2)

Possible instructions for Real:
+, -, *, /, Abs, Neg, Exp, **, Frac, Ln, Sin, ASin, Cos, ACos, Tan, ATan, Sqr, Sqrt, Round, Ceil, Floor, Trunc

Note

You also must create an input for any constants in your function. The constant value would then be entered in the associated input for the CALCULATE instruction.

By entering constants as inputs, you can copy the CALCULATE instruction to other locations in your user program without having to change the function. You then can change the values or tags of the inputs for the instruction without modifying the function.

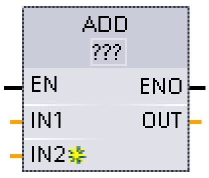
When CALCULATE is executed and all the individual operations in the calculation complete successfully, then the ENO = 1. Otherwise, ENO = 0.

For an example of the CALCULATE instruction, see "Creating a complex equation with a simple instruction (Page 39)".

8.5.2 Add, subtract, multiply and divide instructions

Table 8- 44 Add, subtract, multiply and divide instructions

LAD / FBD	SCL	Description

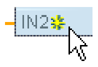
	`out := in1 + in2;`
	`out := in1 - in2;`
	`out := in1 * in2;`
	`out := in1 / in2;`
		• ADD: Addition (IN1 + IN2 = OUT)
		• SUB: Subtraction (IN1 - IN2 = OUT)
		• MUL: Multiplication (IN1 * IN2 = OUT)
		• DIV: Division (IN1 / IN2 = OUT)
		An Integer division operation truncates the fractional part of the quotient to produce an integer output.

[1] For LAD and FBD: Click the "???" and select a data type from the drop-down menu.

Table 8- 45 Data types for the parameters (LAD and FBD)

Parameter	Data type[1]	Description
IN1, IN2	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Constant	Math operation inputs
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	Math operation output

[1] Parameters IN1, IN2, and OUT must be the same data type.



To add an ADD or MUL input, click the "Create" icon or right-click on an input stub for one of the existing IN parameters and select the "Insert input" command.

To remove an input, right-click on an input stub for one of the existing IN parameters (when there are more than the original two inputs) and select the "Delete" command.

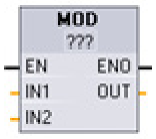
When enabled (EN = 1), the math instruction performs the specified operation on the input values (IN1 and IN2) and stores the result in the memory address specified by the output parameter (OUT). After the successful completion of the operation, the instruction sets ENO = 1.

Table 8- 46 ENO status

ENO	Description
1	No error
0	The Math operation result value would be outside the valid number range of the data type selected. The least significant part of the result that fits in the destination size is returned.
0	Division by 0 (IN2 = 0): The result is undefined and zero is returned.
0	Real/LReal: If one of the input values is NaN (not a number) then NaN is returned.
0	ADD Real/LReal: If both IN values are INF with different signs, this is an illegal operation and NaN is returned.
0	SUB Real/LReal: If both IN values are INF with the same sign, this is an illegal operation and NaN is returned.
0	MUL Real/LReal: If one IN value is zero and the other is INF, this is an illegal operation and NaN is returned.
0	DIV Real/LReal: If both IN values are zero or INF, this is an illegal operation and NaN is returned.

8.5.3 MOD (return remainder of division) instruction

Table 8- 47 Modulo (return remainder of division) instruction

LAD / FBD	SCL	Description
	`out := in1 MOD in2;`	You can use the MOD instruction to return the remainder of an integer division operation. The value at the IN1 input is divided by the value at the IN2 input and the remainder is returned at the OUT output.

[1] For LAD and FBD: Click the "???" and select a data type from the drop-down menu.

Table 8- 48 Data types for parameters

Parameter	Data type[1]	Description
IN1 and IN2	SInt, Int, DInt, USInt, UInt, UDInt, Constant	Modulo inputs
OUT	SInt, Int, DInt, USInt, UInt, UDInt	Modulo output

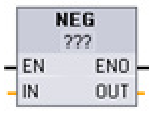
[1] The IN1, IN2, and OUT parameters must be the same data type.

Table 8- 49 ENO values

ENO	Description
1	No error
0	Value IN2 = 0, OUT is assigned the value zero

8.5.4 NEG (Create twos complement) instruction

Table 8- 50 NEG (create twos complement) instruction

LAD / FBD	SCL	Description
	`-(in);`	The NEG instruction inverts the arithmetic sign of the value at parameter IN and stores the result in parameter OUT.

1 For LAD and FBD: Click the "???" and select a data type from the drop-down menu.

Table 8- 51 Data types for parameters

Parameter	Data type[1]	Description
IN	SInt, Int, DInt, Real, LReal, Constant	Math operation input
OUT	SInt, Int, DInt, Real, LReal	Math operation output

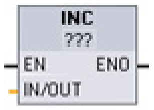
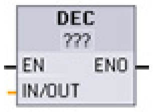
1 The IN and OUT parameters must be the same data type.

Table 8- 52 ENO status

ENO	Description
1	No error
0	The resulting value is outside the valid number range of the selected data type. Example for SInt: NEG (-128) results in +128 which exceeds the data type maximum.

8.5.5 INC (Increment) and DEC (Decrement) instructions

Table 8- 53 INC and DEC instructions

LAD / FBD	SCL	Description
	`in_out := in_out + 1;`	Increments a signed or unsigned integer number value: IN_OUT value +1 = IN_OUT value
	`in_out := in_out - 1;`	Decrements a signed or unsigned integer number value: IN_OUT value - 1 = IN_OUT value

1 For LAD and FBD: Click the "???" and select a data type from the drop-down menu.

Table 8- 54 Data types for parameters

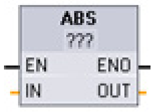
Parameter	Data type	Description
IN/OUT	SInt, Int, DInt, USInt, UInt, UDInt	Math operation input and output

Table 8- 55 ENO status

ENO	Description
1	No error
0	The resulting value is outside the valid number range of the selected data type. Example for SInt: INC (+127) results in +128, which exceeds the data type maximum.

8.5.6 ABS (Form absolute value) instruction

Table 8- 56 ABS (absolute value) instruction

LAD / FBD	SCL	Description
	`out := ABS(in);`	Calculates the absolute value of a signed integer or real number at parameter IN and stores the result in parameter OUT.

1 For LAD and FBD: Click the "???" and select a data type from the drop-down menu.

Table 8- 57 Data types for parameters

Parameter	Data type[1]	Description
IN	SInt, Int, DInt, Real, LReal	Math operation input
OUT	SInt, Int, DInt, Real, LReal	Math operation output

1 The IN and OUT parameters must be the same data type.

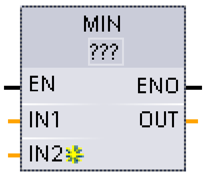
Table 8- 58 ENO status

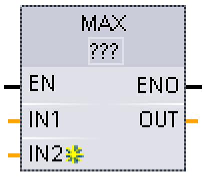
ENO	Description
1	No error
0	The math operation result value is outside the valid number range of the selected data type. Example for SInt: ABS (-128) results in +128 which exceeds the data type maximum.

8.5.7 MIN (Get minimum) and MAX (Get maximum) instructions

Table 8- 59 MIN (get minimum) and MAX (get maximum) instructions

LAD / FBD	SCL	Description

	`out:= MIN(`
	`in1:=_variant_in_,`
	`in2:=_variant_in_`
	`[,...in32]);`
		The MIN instruction compares the value of two parameters IN1 and IN2 and assigns the minimum (lesser) value to parameter OUT.

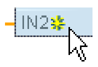
	`out:= MAX(`
	`in1:=_variant_in_,`
	`in2:=_variant_in_`
	`[,...in32]);`
		The MAX instruction compares the value of two parameters IN1 and IN2 and assigns the maximum (greater) value to parameter OUT.

1 For LAD and FBD: Click the "???" and select a data type from the drop-down menu.

Table 8- 60 Data types for the parameters

Parameter	Data type[1]	Description
IN1, IN2 [...IN32]	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Time, Date, TOD, Constant	Math operation inputs (up to 32 inputs)
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Time, Date, TOD	Math operation output

1 The IN1, IN2, and OUT parameters must be the same data type.



To add an input, click the "Create" icon or right-click on an input stub for one of the existing IN parameters and select the "Insert input" command.

To remove an input, right-click on an input stub for one of the existing IN parameters (when there are more than the original two inputs) and select the "Delete" command.

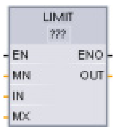
Table 8- 61 ENO status

ENO	Description
1	No error
0	For Real data type only: • At least one input is not a real number (NaN). • The resulting OUT is +/- INF (infinity).

8.5.8 LIMIT (Set limit value) instruction

Table 8- 62 LIMIT (set limit value) instruction

LAD / FBD	SCL	Description

	`LIMIT(MN:=_variant_in_,`
	`IN:=_variant_in_,`
	`MX:=_variant_in_,`
	`OUT:=_variant_out_);`
		The Limit instruction tests if the value of parameter IN is inside the value range specified by parameters MIN and MAX and if not, clamps the value at MIN or MAX.

1 For LAD and FBD: Click the "???" and select a data type from the drop-down menu.

Table 8- 63 Data types for the parameters

Parameter	Data type[1]	Description
MN, IN, and MX	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Time, Date, TOD·Constant	Math operation inputs
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Time, Date, TOD	Math operation output

1 The MN, IN, MX, and OUT parameters must be the same data type.

If the value of parameter IN is within the specified range, then the value of IN is stored in parameter OUT. If the value of parameter IN is outside of the specified range, then the OUT value is the value of parameter MIN (if the IN value is less than the MIN value) or the value of parameter MAX (if the IN value is greater than the MAX value).

Table 8- 64 ENO status

ENO	Description
1	No error
0	Real: If one or more of the values for MIN, IN and MAX is NaN (Not a Number), then NaN is returned.
0	If MIN is greater than MAX, the value IN is assigned to OUT.

SCL examples:

- MyVal := LIMIT(MN:=10,IN:=53, MX:=40); //Result: MyVal = 40
- MyVal := LIMIT(MN:=10,IN:=37, MX:=40); //Result: MyVal = 37
- MyVal := LIMIT(MN:=10,IN:=8, MX:=40); //Result: MyVal = 10

8.5.9 Exponent, logarithm, and trigonometry instructions

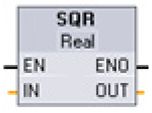
You use the floating point instructions to program mathematical operations using a Real or LReal data type:

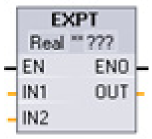
- SQR: Form square (IN^2 = OUT)
- SQRT: Form square root (√IN = OUT)
- LN: Form natural logarithm (LN(IN) = OUT)
- EXP: Form exponential value (e^{IN} =OUT), where base e = 2.71828182845904523536
- EXPT: exponentiate ($IN1^{IN2}$ = OUT)

  EXPT parameters IN1 and OUT are always the same data type, for which you must select Real or LReal. You can select the data type for the exponent parameter IN2 from among many data types.
- FRAC: Return fraction (fractional part of floating point number IN = OUT)
- SIN: Form sine value (sin(IN radians) = OUT)
- ASIN: Form arcsine value (arcsine(IN) = OUT radians), where the sin(OUT radians) = IN
- COS: Form cosine (cos(IN radians) = OUT)
- ACOS: Form arccosine value (arccos(IN) = OUT radians), where the cos(OUT radians) = IN
- TAN: Form tangent value (tan(IN radians) = OUT)
- ATAN: Form arctangent value (arctan(IN) = OUT radians), where the tan(OUT radians) = IN

Table 8- 65 Examples of floating-point math instructions

LAD / FBD	SCL	Description

	`out := SQR(in);`
	or
	`out := in * in;`
		Square: IN^2 = OUT
		For example: If IN = 9, then OUT = 81.

	`out := in1 ** in2;`	General exponential: $IN1^{IN2}$ = OUT For example: If IN1 = 3 and IN2 = 2, then OUT = 9.

[1] For LAD and FBD: Click the "???" (by the instruction name) and select a data type from the drop-down menu.

[2] For SCL: You can also use the basic SCL math operators to create the mathematical expressions.

Table 8- 66 Data types for parameters

Parameter	Data type	Description
IN, IN1	Real, LReal, Constant	Inputs
IN2	SInt, Int, DInt, USInt, UInt,UDInt, Real, LReal, Constant	EXPT exponent input
OUT	Real, LReal	Outputs

Table 8- 67 ENO status

ENO	Instruction	Condition	Result (OUT)
1	All	No error	Valid result
0	SQR	Result exceeds valid Real/LReal range	+INF
		IN is +/- NaN (not a number)	+NaN
	SQRT	IN is negative	-NaN
		IN is +/- INF (infinity) or +/- NaN	+/- INF or +/- NaN
	LN	IN is 0.0, negative, -INF, or -NaN	-NaN
		IN is +INF or +NaN	+INF or +NaN
	EXP	Result exceeds valid Real/LReal range	+INF
		IN is +/- NaN	+/- NaN
	SIN, COS, TAN	IN is +/- INF or +/- NaN	+/- INF or +/- NaN
	ASIN, ACOS	IN is outside valid range of -1.0 to +1.0	+NaN
		IN is +/- NaN	+/- NaN
	ATAN	IN is +/- NaN	+/- NaN
	FRAC	IN is +/- INF or +/- NaN	+NaN
	EXPT	IN1 is +INF and IN2 is not -INF	+INF
		IN1 is negative or -INF	+NaN if IN2 is Real/LReal, -INF otherwise
		IN1 or IN2 is +/- NaN	+NaN
		IN1 is 0.0 and IN2 is Real/LReal (only)	+NaN

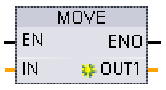
8.6 Move operations

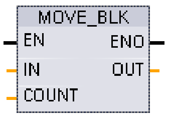
8.6.1 MOVE (Move value), MOVE_BLK (Move block), UMOVE_BLK (Move block uninterruptible), and MOVE_BLK_VARIANT (Move block) instructions

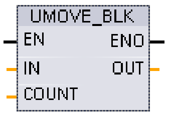
Use the Move instructions to copy data elements to a new memory address and convert from one data type to another. The source data is not changed by the move process.

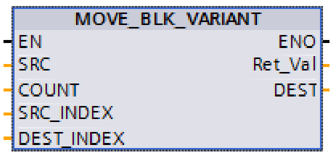
- The MOVE instruction copies a single data element from the source address specified by the IN parameter to the destination addresses specified by the OUT parameter.
- The MOVE_BLK and UMOVE_BLK instructions have an additional COUNT parameter. The COUNT specifies how many data elements are copied. The number of bytes per element copied depends on the data type assigned to the IN and OUT parameter tag names in the PLC tag table.

Table 8- 68 MOVE, MOVE_BLK, UMOVE_BLK, and MOVE_BLK_VARIANT instructions

LAD / FBD	SCL	Description
	`out1 := in;`	Copies a data element stored at a specified address to a new address or multiple addresses.[1]

	`MOVE_BLK(`
	`in:=_variant_in,`
	`count:=_uint_in,`
	`out=>_variant_out);`
		Interruptible move that copies a block of data elements to a new address.

	`UMOVE_BLK(`
	`in:=_variant_in,`
	`count:=_uint_in,`
	`out=>_variant_out);`
		Uninterruptible move that copies a block of data elements to a new address.

 MOVE_BLK_VARIANT
EN
ENO
SRC
Ret_Val
COUNT
DEST
SRC_INDEX
DEST_INDEX
	`MOVE_BLK(`
	`SRC:=_variant_in,`
	`COUNT:=_udint_in,`
	`SRC_INDEX:=_dint_in,`
	
	`DEST_INDEX:=_dint_in,`
	`DEST=>_variant_out);`
		Moves the contents of a source memory area to a destination memory area.
		You can copy a complete array or elements of an array to another array of the same data type. The size (number of elements) of source and destination array may be different. You can copy multiple or single elements within an array. You use Variant data types to point to both the source and destination arrays.

[1] MOVE instruction: To add another output in LAD or FBD, click the "Create" icon by the output parameter. For SCL, use multiple assignment statements. You might also use one of the loop constructions.

Table 8- 69 Data types for the MOVE instruction

Parameter	Data type	Description
IN	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Char, WChar, Array, Struct, DTL, Time, Date, TOD, IEC data types, PLC data types	Source address
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Char, WChar, Array, Struct, DTL, Time, Date, TOD, IEC data types, PLC data types	Destination address

To add MOVE outputs, click the "Create" icon or right-click on an output stub for one of the existing OUT parameters and select the "Insert output" command.

To remove an output, right-click on an output stub for one of the existing OUT parameters (when there are more than the original two outputs) and select the "Delete" command.

Table 8- 70 Data types for the MOVE_BLK and UMOVE_BLK instructions

Parameter	Data type	Description
IN	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal Byte, Word, DWord, Time, Date, TOD, WChar	Source start address
COUNT	UInt	Number of data elements to copy
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Time, Date, TOD, WChar	Destination start address

Table 8- 71 Data types for the MOVE_BLK_VARIANT instruction

Parameter	Data type	Description
SRC	Variant (which points to an array or individual array element)	Source block from which to copy
COUNT	UDInt	Number of data elements to copy
SRC_INDEX	DInt	Zero-based index into the SRC array
DEST_INDEX	DInt	Zero-based index into the DEST array
RET_VAL	Int	Error information
DEST	Variant (which points to an array or individualt array element)	Destination area into which to copy the contents of the source block

---

Note

Rules for data copy operations

- To copy the Bool data type, use SET_BF, RESET_BF, R, S, or output coil (LAD) (Page 212)
- To copy a single elementary data type, use MOVE
- To copy an array of an elementary data type, use MOVE_BLK or UMOVE_BLK
- To copy a structure, use MOVE
- To copy a string, use S_MOVE (Page 326)
- To copy a single character in a string, use MOVE
- The MOVE_BLK and UMOVE_BLK instructions cannot be used to copy arrays or structures to the I, Q, or M memory areas.

MOVE_BLK and UMOVE_BLK instructions differ in how interrupts are handled:

- Interrupt events are **queued and processed** during MOVE_BLK execution. Use the MOVE_BLK instruction when the data at the move destination address is not used within an interrupt OB subprogram or, if used, the destination data does not have to be consistent. If a MOVE_BLK operation is interrupted, then the last data element moved is complete and consistent at the destination address. The MOVE_BLK operation is resumed after the interrupt OB execution is complete.
- Interrupt events are **queued but not processed** until UMOVE_BLK execution is complete. Use the UMOVE_BLK instruction when the move operation must be completed and the destination data consistent, before the execution of an interrupt OB subprogram. For more information, see the section on data consistency (Page 185).

ENO is always true following execution of the MOVE instruction.

Table 8- 72 ENO status

ENO	Condition	Result
1	No error	All COUNT elements were successfully copied.
0	Either the source (IN) range or the destination (OUT) range exceeds the available memory area.	Elements that fit are copied. No partial elements are copied.

Table 8- 73 Condition codes for the MOVE_BLK_VARIANT instruction

RET_VAL (W#16#...)	Description
0000	No error
80B4	Data types do not correspond.
8151	Access to the SRC parameter is not possible.
8152	The operand at the SRC parameter is an invalid type.
8153	Code generation error at the SRC parameter
8154	The operand at the SRC parameter has the data type Bool.
8281	The COUNT parameter has an invalid value.

RET_VAL (W#16#...)	Description
8382	The value at the SRC_INDEX parameter is outside the limits of the Variant.
8383	The value at parameter SRC_INDEX is outside the high limit of the array.
8482	The value at the DEST_INDEX parameter is outside the limits of the Variant.
8483	The value at parameter DEST_INDEX is outside the high limit of the array.
8534	The DEST parameter is write-protected.
8551	Access to the DEST parameter is not possible.
8552	The operand at the DEST parameter is an invalid type.
8553	Code generation error at the DEST parameter
8554	The operand at the DEST parameter has the data type Bool.
*You can display error codes in the program editor as integer or hexadecimal values.	

8.6.2 Deserialize

You can use the "Deserialize" instruction to convert the sequential representation of a PLC data type (UDT) back to a PLC data type and to fill its entire contents. If the comparison is TRUE, then the box output is TRUE.

The memory area which holds the sequential representation of a PLC data type must have the Array of Byte data type and you must declare the data block to have standard (not optimized) access. Make sure that there is enough memory space prior to the conversion.

The instruction enables you to convert multiple sequential representations of converted PLC data types back to their original data types.

Note

If you only want to convert back a single sequential representation of a PLC data type (UDT), you can also use the instruction "TRCV: Receive data via communication connection".

Table 8- 74 DESERIALIZE instruction

LAD / FBD	SCL	Description

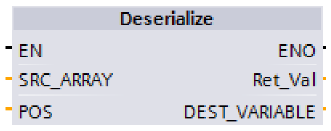
 Deserialize
EN
ENO
SRC_ARRAY
Ret_Val
POS
DEST_VARIABLE
	`ret_val := Deserialize(`
	`SRC_ARRAY:=_variant_in_,`
	`DEST_VARIABLE=>_variant_out`
	`_,`
	`POS:=_dint_inout_);`
		Converts the sequential representation of a PLC data type (UDT) back to a PLC data type and fills its entire contents

Table 8- 75 Parameters for the DESERIALIZE instruction

Parameter	Type	Data type	Description
SRC_ARRAY	IN	Variant	Global data blockwhich contains the data stream
DEST_VARIABLE	INOUT	Variant	Tag in which to store the converted PLC data type (UDT)
POS	INOUT	DInt	Number of bytes that the converted PLC data type uses
RET_VAL	OUT	Int	Error information

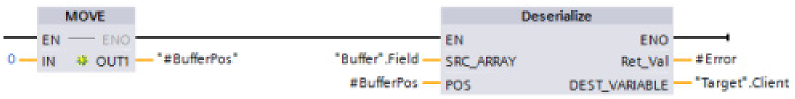
Table 8- 76 RET_VAL parameter

RET_VAL* (W#16#...)	Description
0000	No error
80B0	The memory areas for the SRC_ARRAY and DEST_VARIABLE parameters overlap.
8136	The data block at the DEST_VARIABLE parameter is not a block with standard access.
8150	The Variant data type at the SRC_ARRAY parameter contains no value.
8151	Code generation error at the SRC_ARRAY parameter.
8153	There is not enough free memory available at the SRC_ARRAY parameter.
8250	The Variant data type at the DEST_VARIABLE parameter contains no value.
8251	Code generation error at the DEST_VARIABLE parameter.
8254	Invalid data type at the DEST_VARIABLE parameter.
8382	The value at parameter POS is outside the limits of the array.
*You can view the error codes as either integer or hexadecimal in the program editor.	

Example: Deserialize instruction

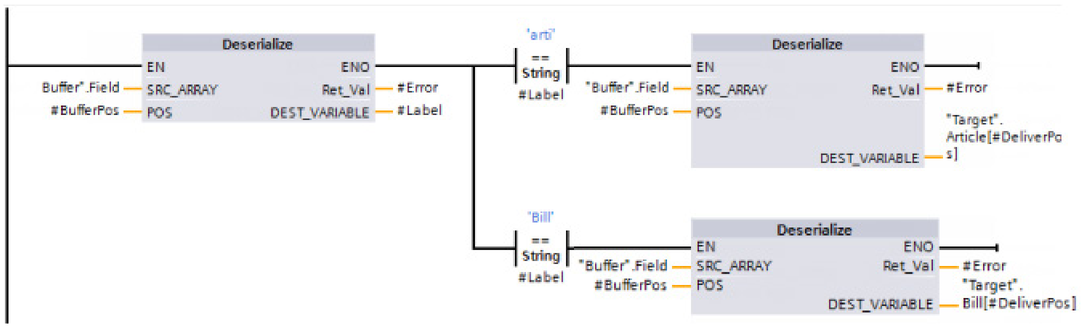
The following example shows how the instruction works:

Network 1:



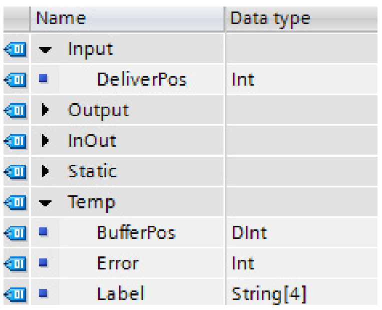
The "MOVE" instruction moves the value "0" to the "#BufferPos" data block tag. The Deserialize instruction then deserializes the sequential representation of the customer data from the "Buffer" data block and writes it to the "Target" data block. The Deserialize instruction calculates the number of bytes that the converted data uses and stores it in the "#BufferPos" data block tag.

Network 2:



The "Deserialize" instruction deserializes the sequential representation of the data stream pointed to by "Buffer" and writes the characters to the "#Label" operand. The logic compares the characters using the comparison instructions "arti" and "Bill". If the comparison for "arti" = TRUE, the data is article data that is to be deserialized and written to the "Article" data structure of the "Target" data block. If the comparison for "Bill" = TRUE, the data is billing data that is to be deserialized and written to the "Bill" data structure of the "Target" data block.

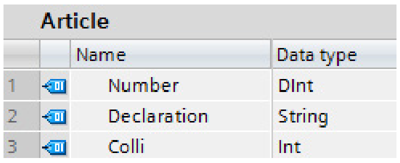
Function block (or Function) interface:



Name	Data type
▼ Input	
DeliverPos	Int
▶ Output	
▶ InOut	
▶ Static	
▼ Temp	
BufferPos	DInt
Error	Int
Label	String[4]

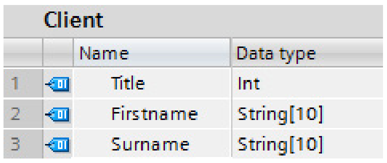
Custom PLC data types:

The structure of the two PLC data types (UDTs) for this example are as follows:



Article

	Name	Data type
1	Number	DInt
2	Declaration	String
3	Colli	Int

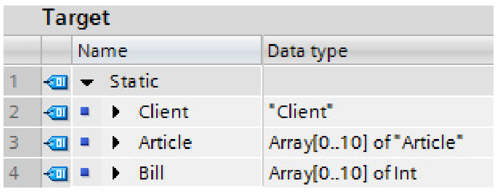


Client

	Name	Data type
1	Title	Int
2	Firstname	String[10]
3	Surname	String[10]

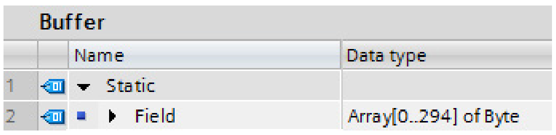
Data blocks:

The two data blocks for this example are as follows:



Target

	Name	Data type
1	Static	
2	Client	"Client"
3	Article	Array[0..10] of "Article"
4	Bill	Array[0..10] of Int



Buffer

	Name	Data type
1	Static	
2	Field	Array[0..294] of Byte

8.6.3 Serialize

You can use the "Serialize" instruction to convert several PLC data types (UDTs) to a sequential representation without any loss of structure.

You can use the instruction to temporarily save multiple structured data items from your program to a buffer, for example to a global data block, and send them to another CPU. The memory area in which the converted PLC data types are stored must have the ARRAY of BYTE data type and be declared with standard access. Make sure that there is enough memory space prior to the conversion.

The POS parameter contains information about the number of bytes that the converted PLC data types use.

Note

If you only want to send a single PLC data type (UDT), you can use the instruction "TSEND: Send data via communication connection".

Table 8- 77 SERIALIZE instruction

LAD / FBD	SCL	Description

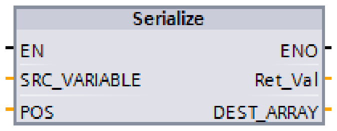
 Serialize
EN
ENO
SRC_VARIABLE
Ret_Val
POS
DEST_ARRAY
	`ret_val := Serialize(`
	`SRC_VARIABLE=>_variant_in_,`
	`DEST_ARRAY:=_variant_out_,`
	`POS:=_dint_inout_);`
		Converts a PLC data type (UDT) to a sequential representation.

Table 8- 78 Parameters for the SERIALIZE instruction

Parameter	Type	Data type	Description
SRC_VARIABLE	IN	Variant	PLC data type (UDT) that is to be converted to a serial representation
DEST_ARRAY	INOUT	Variant	Data block in which the generated data stream is to be stored
POS	INOUT	DInt	Number of bytes that the converted PLC data types use. The calculated POS parameter is zero-based.
RET_VAL	OUT	Int	Error information

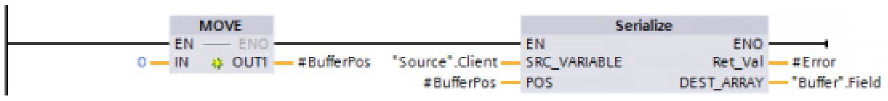
Table 8- 79 RET_VAL parameter

RET_VAL* (W#16#...)	Description
0000	No error
80B0	The memory areas for the SRC_VARIABLE and DEST_ARRAY parameters overlap.
8150	The Variant data type at the SRC_VARIABLE parameter contains no value.
8152	Code generation error at the SRC_VARIABLE parameter.
8236	The data block at the DEST_ARRAY parameter is not a block with standard access.
8250	The Variant data type at the DEST_ARRAY parameter contains no value.
8252	Code generation error at the DEST_ARRAY parameter.
8253	There is not enough free memory available at the DEST_ARRAY parameter.
8254	Invalid data type at the DEST_VARIABLE parameter.
8382	The value at parameter POS is outside the limits of the array.
*You can view the error codes as either integer or hexadecimal in the program editor.	

Example: Serialize instruction

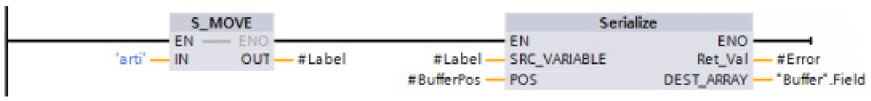
The following example shows how the instruction works:

Network 1:



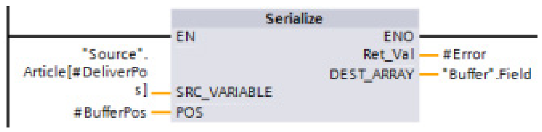
The "MOVE" instruction moves the value "0" to the "#BufferPos" parameter. The "Serialize" instruction serializes the customer data from the "Source" data block and writes it in sequential representation to the "Buffer" data block. The instruction stores the number of bytes used by the sequential representation in the "#BufferPos" parameter.

Network 2:



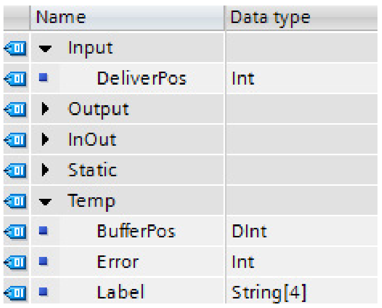
The logic now inserts some separator text to make it easier to deserialize the sequential representation later. The "S_MOVE" instruction moves the text string "arti" to the "#Label" parameter. The "Serialize" instruction writes these characters after the source client data to the "Buffer" data block. The instruction adds the number of bytes in the text string "arti" to the number already stored in the "#BufferPos" parameter.

Network 3:



The "Serialize" instruction serializes the data of a specific article, which is calculated in runtime, from the "Source" data block and writes it in sequential representation to the "Buffer" data block after the "arti" characters

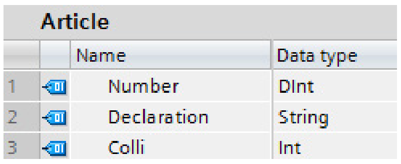
Block Interface:



Name	Data type
▼ Input	
DeliverPos	Int
▶ Output	
▶ InOut	
▶ Static	
▼ Temp	
BufferPos	DInt
Error	Int
Label	String[4]

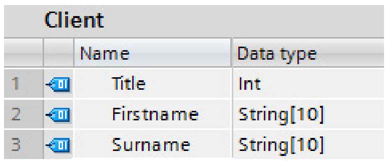
Custom PLC data types:

The structure of the two PLC data types (UDTs) for this example are as follows:



Article

	Name	Data type
1	Number	DInt
2	Declaration	String
3	Colli	Int

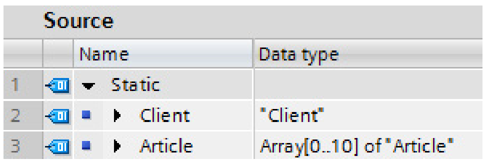


Client

	Name	Data type
1	Title	Int
2	Firstname	String[10]
3	Surname	String[10]

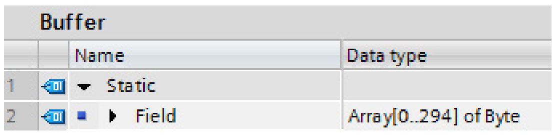
Data blocks:

The two data blocks for this example are as follows:



Source

	Name	Data type
1	▼ Static	
2	▶ Client	"Client"
3	▶ Article	Array[0..10] of "Article"



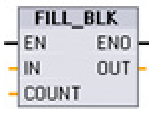
Buffer

	Name	Data type
1	▼ Static	
2	▶ Field	Array[0..294] of Byte

8.6.4 FILL_BLK (Fill block) and UFILL_BLK (Fill block uninterruptible) instructions

Table 8- 80 FILL_BLK and UFILL_BLK instructions

LAD / FBD	SCL	Description

	`FILL_BLK(`
	`in:=_variant_in,`
	`count:=int,`
	`out=>_variant_out);`
		Interruptible fill instruction: Fills an address range with copies of a specified data element

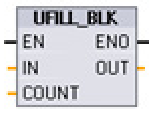
	`UFILL_BLK(`
	`in:=_variant_in,`
	`count:=int`
	`out=>_variant_out);`
		Uninterruptible fill instruction: Fills an address range with copies of a specified data element

Table 8- 81 Data types for parameters

Parameter	Data type	Description
IN	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Time, Date, TOD, Char, WChar	Data source address
COUNT	UDint, USInt, UInt	Number of data elements to copy
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Time, Date, TOD, Char, WChar	Data destination address

Note

Rules for data fill operations

- To fill with the BOOL data type, use SET_BF, RESET_BF, R, S, or output coil (LAD)
- To fill with a single elementary data type, use MOVE
- To fill an array with an elementary data type, use FILL_BLK or UFILL_BLK
- To fill a single character in a string, use MOVE
- The FILL_BLK and UFILL_BLK instructions cannot be used to fill arrays in the I, Q, or M memory areas.

The FILL_BLK and UFILL_BLK instructions copy the source data element IN to the destination where the initial address is specified by the parameter OUT. The copy process repeats and a block of adjacent addresses is filled until the number of copies is equal to the COUNT parameter.

FILL_BLK and UFILL_BLK instructions differ in how interrupts are handled:

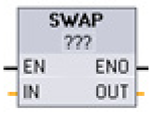
- Interrupt events are **queued and processed** during FILL_BLK execution. Use the FILL_BLK instruction when the data at the move destination address is not used within an interrupt OB subprogram or, if used, the destination data does not have to be consistent.
- Interrupt events are **queued but not processed** until UFILL_BLK execution is complete. Use the UFILL_BLK instruction when the move operation must be completed and the destination data consistent, before the execution of an interrupt OB subprogram.

Table 8- 82 ENO status

ENO	Condition	Result
1	No error	The IN element was successfully copied to all COUNT destinations.
0	The destination (OUT) range exceeds the available memory area	Elements that fit are copied. No partial elements are copied.

8.6.5 SWAP (Swap bytes) instruction

Table 8- 83 SWAP instruction

LAD / FBD	SCL	Description
	`out := SWAP(in);`	Reverses the byte order for two-byte and four-byte data elements. No change is made to the bit order within each byte. ENO is always TRUE following execution of the SWAP instruction.

¹ For LAD and FBD: Click the "???" and select a data type from the drop-down menu.

Table 8- 84 Data types for the parameters

Parameter	Data type	Description
IN	Word, DWord	Ordered data bytes IN
OUT	Word, DWord	Reverse ordered data bytes OUT

Example 1	Parameter IN = MB0 (before execution)		Parameter OUT = MB4, (after execution)	
Address	MW0	MB1	MW4	MB5
W#16#1234	12	34	34	12
WORD	MSB	LSB	MSB	LSB

Example 2	Parameter IN = MB0 (before execution)				Parameter OUT = MB4, (after execution)			
Address	MD0	MB1	MB2	MB3	MD4	MB5	MB6	MB7
DW#16# 12345678	12	34	56	78	78	56	34	12
DWORD	MSB			LSB	MSB			LSB

8.6.6 Read / Write memory instructions

8.6.6.1 PEEK and POKE instructions (SCL only)

SCL provides PEEK and POKE instructions that allow you to read from or write to data blocks, I/O, or memory. You provide parameters for specific byte offsets or bit offsets for the operation.

Note

To use the PEEK and POKE instructions with data blocks, you must use standard (not optimized) data blocks. Also note that the PEEK and POKE instructions merely transfer data. They have no knowledge of data types at the addresses.

`PEEK(area:=_in_,`
`          dbNumber:=_in_,`
`          byteOffset:=_in_);`
	Reads the byte referenced by byteOffset of the referenced data block, I/O or memory area.
	Example referencing data block:
	`%MB100 := PEEK(area:=16#84, dbNumber:=1, byteOffset:=#i);`
	Example referencing IB3 input:
	`%MB100 := PEEK(area:=16#81, dbNumber:=0, byteOffset:=#i); // when #i = 3`

`PEEK_WORD(area:=_in_,`
`          dbNumber:=_in_,`
`          byteOffset:=_in_);`
	Reads the word referenced by byteOffset of the referenced data block, I/O or memory area.
	Example:
	`%MW200 := PEEK_WORD(area:=16#84, dbNumber:=1, byteOffset:=#i);`

`PEEK_DWORD(area:=_in_,`
`          dbNumber:=_in_,`
`          byteOffset:=_in_);`
	Reads the double word referenced by byteOffset of the referenced data block, I/O or memory area.
	Example:
	`%MD300 := PEEK_DWORD(area:=16#84, dbNumber:=1, byteOffset:=#i);`

`PEEK_BOOL(area:=_in_,`
`          dbNumber:=_in_,`
`          byteOffset:=_in_,`
`          bitOffset:=_in_);`
	Reads a Boolean referenced by the bitOffset and byteOffset of the referenced data block, I/O or memory area
	Example:
	`%MB100.0 := PEEK_BOOL(area:=16#84, dbNumber:=1, byteOffset:=#ii, bitOffset:=#j);`

`POKE(area:=_in_,`
`dbNumber:=_in_,`
`byteOffset:=_in_,`
`value:=_in_);`
	Writes the value (Byte, Word, or DWord) to the referenced byteOffset of the referenced data block, I/O or memory area
	Example referencing data block:
	`POKE(area:=16#84, dbNumber:=2, byteOffset:=3, value:="Tag_1");`
	Example referencing QB3 output:
	`POKE(area:=16#82, dbNumber:=0, byteOffset:=3, value:="Tag_1");`

`POKE_BOOL(area:=_in_,`
`dbNumber:=_in_,`
`byteOffset:=_in_,`
`bitOffset:=_in_,`
`value:=_in_);`
	Writes the Boolean value to the referenced bitOffset and byteOffset of the referenced data block, I/O or memory area
	Example:
	`POKE_BOOL(area:=16#84, dbNumber:=2, byteOffset:=3, bitOffset:=5, val-ue:=0);`

`POKE_BLK(area_src:=_in_,`
`dbNumber_src:=_in_,`
`byteOffset_src:=_in_,`
`area_dest:=_in_,`
`dbNumber_dest:=_in_,`
`byteOffset_dest:=_in_,`
`count:=_in_);`
	Writes "count" number of bytes starting at the referenced byte Offset of the referenced source data block, I/O or memory area to the referenced byteOffset of the referenced desti-nation data block, I/O or memory area
	Example:
	`POKE_BLK(area_src:=16#84, dbNumber_src:=#src_db, byteOff-set_src:=#src_byte, area_dest:=16#84, dbNumber_dest:=#src_db, byteOff-set_dest:=#src_byte, count:=10);`

For PEEK and POKE instructions, the following values for the "area", "area_src" and "area_dest" parameters are applicable. For areas other than data blocks, the dbNumber parameter must be 0.

16#81	I
16#82	Q
16#83	M
16#84	DB

8.6.6.2 Read and write big and little Endian instructions (SCL)

The S7-1200 CPU provides SCL instructions for reading and writing data in little endian format and in big endian format. Little endian format means that the byte with the least significant bit is in the lowest memory address. Big endian format means that the byte with the most significant bit is in the lowest memory address.

The four SCL instructions for reading and writing data in little endian and big endian format are as follows:

- READ_LITTLE (Read data in little endian format)
- WRITE_LITTLE (Write data in little endian format)
- READ_BIG (Read data in big endian format)
- WRITE_BIG (Write data in big endian format)

Table 8- 85 Read and write big and little endian instructions

LAD / FBD	SCL	Description

Not available	`READ_LITTLE(`
	`    src_array:=_variant_in_,`
	`    dest_Variable =>_out_,`
	`    pos:=_dint_inout)`
		Reads data from a memory area and writes it to a single tag in little endian byte format.

Not available	`WRITE_LITTLE(`
	`    src_variable:=_in_,`
	`    dest_array =>_variant_inout_,`
	`    pos:=_dint_inout)`
		Writes data from a single tag to a memory area in little endian byte format.

Not available	`READ_BIG(`
	`    src_array:=_variant_in_,`
	`    dest_Variable =>_out_,`
	`    pos:=_dint_inout)`
		Reads data from a memory area and writes it to a single tag in big endian byte format.

Not available	`WRITE_BIG(`
	`    src_variable:=_in_,`
	`    dest_array =>_variant_inout_,`
	`    pos:=_dint_inout)`
		Writes data from a single tag to a memory area in big endian byte format.

Table 8- 86 Parameters for the READ_LITTLE and READ_BIG instructions

Parameter	Data type	Description
src_array	Array of Byte	Memory area from which to read data
dest_Variable	Bit strings, integers, floating-point numbers, timers, date and time, character strings	Destination variable at which to write data
pos	DINT	Zero-based position from which to start reading data from the src_array input.

Table 8- 87 Parameters for the WRITE_LITTLE and WRITE_BIG instructions

Parameter	Data type	Description
src_variable	Bit strings, integers, floating-point numbers, LDT, TOD, LTOD, DATA, Char, WChar	Source data from tag
dest_array	Array of Byte	Memory area at which to write data
pos	DINT	Zero-based position at which to start writing data into the dest_array output.

Table 8- 88 RET_VAL parameter

RET_VAL* (W#16#...)	Description
0000	No error
80B4	The SRC_ARRAY or DEST_ARRAY is not an Array of Byte
8382	The value at parameter POS is outside the limits of the array.
8383	The value at parameter POS is within the limits of the Array but the size of the memory area exceeds the high limit of the array.
*You can view the error codes as either integer or hexadecimal in the program editor.	

8.6.7 Variant instructions

8.6.7.1 VariantGet (Read VARIANT tag value)

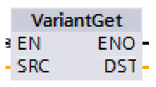
You can use the "Read out Variant tag value" instruction to read the value of the tag to which the Variant pointer at the SRC parameter points and write it in the tag at the DST parameter.

The SRC parameter has the Variant data type. Any data type except for Variant can be specified at the DST parameter.

The data type of the tag at the DST parameter must match the data type to which the Variant points.

Table 8- 89 VariantGet instruction

LAD / FBD	SCL	Description

	`VariantGet(`
	`    SRC:=_variant_in_,`
	`    DST=>_variant_out_);`
		Reads the tag pointed to by the SRC parameter and writes it to the tag at the DST parameter

Note

To copy structures and arrays, you can use the "MOVE_BLK_VARIANT: Move block" instruction.

Table 8- 90 Parameters for the VariantGet instruction

Parameter	Data type	Description
SRC	Variant	Pointer to source data
DST	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY elements, PLC data types	Destination at which to write data

Table 8- 91 ENO status

ENO	Condition	Result
1	No error	Instruction copied the tag data pointed to by SRC to the DST tag.
0	Enable input EN has the signal state "0" or the data types do not correspond.	Instruction copied no data.

8.6.7.2 VariantPut (Write VARIANT tag value) instruction

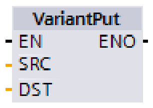
You can use the "Write VARIANT tag value" instruction to write the value of the tag at the SRC parameter to the tag at the DST parameter to which the VARIANT points.

The DST parameter has the VARIANT data type. Any data type except for VARIANT can be specified at the SRC parameter.

The data type of the tag at the SRC parameter must match the data type to which the VARIANT points.

Table 8- 92 VariantPut instruction

LAD / FBD	SCL	Description

	`VariantPut(`
	`    SRC:=_variant_in_,`
	`    DST=>_variant_in_);`
		Writes the tag referenced by the SRC parameter to the variant pointed to by the DST parameter

Note

To copy structures and ARRAYs, you can use the "MOVE_BLK_VARIANT: Move block" instruction.

Table 8- 93 Parameters for the VariantPut instruction

Parameter	Data type	Description
SRC	Bit strings, integers, floating-point numbers, timers, date and time, character strings, ARRAY elements, PLC data types	Pointer to source data
DST	Variant	Destination at which to write data

Table 8- 94 ENO status

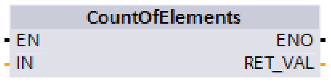
ENO	Condition	Result
1	No error	Instruction copied the SRC tag data to the DST tag.
0	Enable input EN has the signal state "0" or the data types do not correspond.	Instruction copied no data.

8.6.7.3 CountOfElements (Get number of ARRAY elements) instruction

You can use the "Get number of ARRAY elements" instruction to query how many Array elements are in a tag pointed to by a Variant.

If it is a one-dimensional ARRAY, the instruction returns the difference between the high and low limit +1 is output. If it is a multi-dimensional ARRAY, the instruction returns the product of all dimensions.

Table 8- 95 CountOfElements instruction

LAD / FBD	SCL	Description
	`Result := CountOfElements( _variant_in_);`	Counts the number of array elements at the array pointed to by the IN parameter.

Note

If the Variant points to an Array of Bool, the instruction counts the fill elements to the nearest byte boundary. For example, the instruction returns 8 as the count for an Array[0..1] of Bool.

Table 8- 96 Parameters for the CountOfElements instruction

Parameter	Data type	Description
IN	Variant	Tag with array elements to be counted
RET_VAL	UDint	Instruction result

Table 8- 97 ENO status

ENO	Condition	Result
1	No error	Instruction returns the number of array elements.
0	Enable input EN has the signal state "0" or the Variant does not point to an array.	Instruction returns 0.

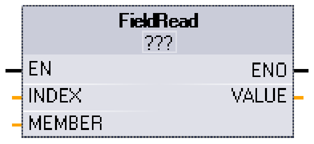
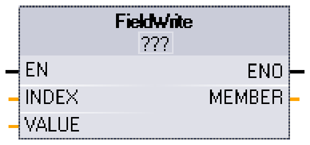
8.6.8 Legacy instructions

8.6.8.1 FieldRead (Read field) and FieldWrite (Write field) instructions

Note

STEP 7 V10.5 **did not support** a variable reference as an array index or multi-dimensional arrays. The FieldRead and FieldWrite instructions were used to provide variable array index operations for a one-dimensional array. STEP 7 V11 and greater **do support** a variable as an array index and multi-dimensional arrays. FieldRead and FieldWrite are included in STEP 7 V11 and greater for backward compatibility with programs that have used these instructions.

Table 8- 98 FieldRead and FieldWrite instructions

LAD / FBD	SCL	Description
	`value := mem-ber[index];`	FieldRead reads the array element with the index value INDEX from the array whose first element in specified by the MEMBER parameter. The value of the array element is transferred to the location speci-fied at the VALUE parameter.
	`member[index] := val-ue;`	WriteField transfers the value at the location specified by the VALUE parameter to the array whose first element is specified by the MEMBER parameter. The value is transferred to the array element whose array index is specified by the INDEX parameter.

1 For LAD and FBD: Click the "???" and select a data type from the drop-down menu.

Table 8- 99 Data types for parameters

Parameter and type		Data type	Description
Index	Input	DInt	The index number of the array element to be read or written to
Member 1	Input	Binary numbers, integers, float-ing-point numbers, timers, DATE, TOD, CHAR and WCHAR as components of an ARRAY tag	Location of the first element in a one- dimension array defined in a global data block or block inter-face. For example: If the array index is specified as [-2..4], then the index of the first element is -2 and not 0.
Value 1	Out	Binary numbers, integers, float-ing-point numbers, timers, DATE, TOD, CHAR, WCHAR	Location to which the specified array element is copied (FieldRead) Location of the value that is copied to the specified array element (FieldWrite)

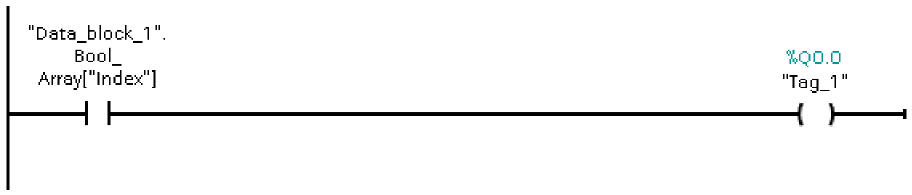
1 The data type of the array element specified by the MEMBER parameter and the VALUE parameter must have the same data type.

The enable output ENO = 0, if one of the following conditions applies:

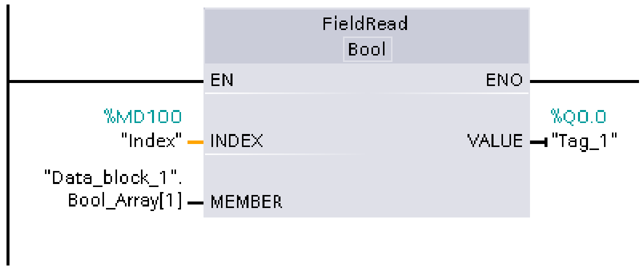
- The EN input has signal state "0"
- The array element specified at the INDEX parameter is not defined in the array referenced at MEMBER parameter
- Errors such as an overflow occur during processing

Example: Accessing data by array indexing

To access elements of an array with a variable, simply use the variable as an array index in your program logic. For example, the network below sets an output based on the Boolean value of an array of Booleans in "Data_block_1" referenced by the PLC tag "Index".



The logic with the variable array index is equivalent to the former method using the FieldRead instruction:



FieldWrite and FieldRead instructions can be replaced with variable array indexing logic.

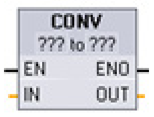
SCL has no FieldRead or FieldWrite instructions, but supports indirect addressing of an array with a variable:

```
#Tag_1 := "Data_block_1".Bool_Array[#Index];
```

8.7 Conversion operations

8.7.1 CONV (Convert value) instruction

Table 8- 100 Convert (CONV) instruction

LAD / FBD	SCL	Description
	`out := <data type in>_TO_<data type out>(in);`	Converts a data element from one data type to another data type.

1 For LAD and FBD: Click the "???" and select the data types from the drop-down menu.

2 For SCL: Construct the conversion instruction by identifying the data type for the input parameter (in) and output parameter (out). For example, DWORD_TO_REAL converts a DWord value to a Real value.

Table 8- 101 Data types for the parameters

Parameter	Data type	Description
IN	Bit string[1], SInt, USInt, Int, UInt, DInt, UDInt, Real, LReal, BCD16, BCD32, Char, WChar	Input value
OUT	Bit string[1], SInt, USInt, Int, UInt, DInt, UDInt, Real, LReal, BCD16, BCD32, Char, WChar	Input value converted to a new data type

1 The instruction does not allow you to select Bit strings (Byte, Word, DWord). To enter an operand of data type Byte, Word, or DWord for a parameter of the instruction, select an unsigned integer with the same bit length. For example, select USInt for a Byte, UInt for a Word, or UDInt for a DWord.

After you select the (convert from) data type, a list of possible conversions is shown in the (convert to) dropdown list. Conversions from and to BCD16 are restricted to the Int data type. Conversions from and to BCD32 are restricted to the DInt data type.

Table 8- 102 ENO status

ENO	Description	Result OUT
1	No error	Valid result
0	IN is +/- INF or +/- NaN	+/- INF or +/- NaN
0	Result exceeds valid range for OUT data type	OUT is set to the IN value

8.7.2 Conversion instructions for SCL

Conversion instructions for SCL

Table 8- 103 Conversion from a Bool, Byte, Word, or DWord

Data type	Instruction	Result
Bool	`BOOL_TO_BYTE, BOOL_TO_WORD, BOOL_TO_DWORD, BOOL_TO_INT, BOOL_TO_DINT`	The value is transferred to the least significant bit of the target data type.
Byte	`BYTE_TO_BOOL`	The least significant bit is transferred into the destination data type.
	`BYTE_TO_WORD, BYTE_TO_DWORD`	The value is transferred to the least significant byte of the target data type.
	`BYTE_TO_SINT, BYTE_TO_USINT`	The value is transferred to the target data type.
	`BYTE_TO_INT, BYTE_TO_UINT, BYTE_TO_DINT, BYTE_TO_UDINT`	The value is transferred to the least significant byte of the target data type.
Word	`WORD_TO_BOOL`	The least significant bit is transferred into the destination data type.
	`WORD_TO_BYTE`	The least significant byte of the source value is transferred to the target data type
	`WORD_TO_DWORD`	The value is transferred to the least significant word of the target data type.
	`WORD_TO_SINT, WORD_TO_USINT`	The least significant byte of the source value is transferred to the target data type.
	`WORD_TO_INT, WORD_TO_UINT`	The value is transferred to the target data type.
	`WORD_TO_DINT, WORD_TO_UDINT`	The value is transferred to the least significant word of the target data type.
DWord	`DWORD_TO_BOOL`	The least significant bit is transferred into the destination data type.
	`DWORD_TO_BYTE, DWORD_TO_WORD, DWORD_TO_SINT`	The least significant byte of the source value is transferred to the target data type.
	`DWORD_TO_USINT, DWORD_TO_INT, DWORD_TO_UINT`	The least significant word of the source value is transferred to the target data type.
	`DWORD_TO_DINT, DWORD_TO_UDINT, DWORD_TO_REAL`	The value is transferred to the target data type.

Table 8- 104 Conversion from a short integer (SInt or USInt)

Data type	Instruction	Result
SInt	`SINT_TO_BOOL`	The least significant bit is transferred into the destination data type.
	`SINT_TO_BYTE`	The value is transferred to the target data type
	`SINT_TO_WORD, SINT_TO_DWORD`	The value is transferred to the least significant byte of the target data type.
	`SINT_TO_INT, SINT_TO_DINT, SINT_TO_USINT, SINT_TO_UINT, SINT_TO_UDINT, SINT_TO_REAL, SINT_TO_LREAL, SINT_TO_CHAR, SINT_TO_STRING`	The value is converted.
USInt	`USINT_TO_BOOL`	The least significant bit is transferred into the destination data type.
	`USINT_TO_BYTE`	The value is transferred to the target data type
	`USINT_TO_WORD, USINT_TO_DWORD, USINT_TO_INT, USINT_TO_UINT, USINT_TO_DINT, USINT_TO_UDINT`	The value is transferred to the least significant byte of the target data type.
	`USINT_TO_SINT, USINT_TO_REAL, USINT_TO_LREAL, USINT_TO_CHAR, USINT_TO_STRING`	The value is converted.

Table 8- 105 Conversion from an integer (Int or UInt)

Data type	instruction	Result
Int	`INT_TO_BOOL`	The least significant bit is transferred into the destination data type.
	`INT_TO_BYTE, INT_TO_DWORD, INT_TO_SINT, INT_TO_USINT, INT_TO_UINT, INT_TO_UDINT, INT_TO_REAL, INT_TO_LREAL, INT_TO_CHAR, INT_TO_STRING`	The value is converted.
	`INT_TO_WORD`	The value is transferred to the target data type.
	`INT_TO_DINT`	The value is transferred to the least significant byte of the target data type.
UInt	`UINT_TO_BOOL`	The least significant bit is transferred into the destination data type.
	`UINT_TO_BYTE, UINT_TO_SINT, UINT_TO_USINT, UINT_TO_INT, UINT_TO_REAL, UINT_TO_LREAL, UINT_TO_CHAR, UINT_TO_STRING`	The value is converted.
	`UINT_TO_WORD, UINT_TO_DATE`	The value is transferred to the target data type.
	`UINT_TO_DWORD, UINT_TO_DINT, UINT_TO_UDINT`	The value is transferred to the least significant byte of the target data type.

Table 8- 106 Conversion from a double integer (Dint or UDInt)

Data type	Instruction	Result
DInt	`DINT_TO_BOOL`	The least significant bit is transferred into the destination data type.
	`DINT_TO_BYTE, DINT_TO_WORD, DINT_TO_SINT, DINT_TO_USINT, DINT_TO_INT, DINT_TO_UINT, DINT_TO_UDINT, DINT_TO_REAL, DINT_TO_LREAL, DINT_TO_CHAR, DINT_TO_STRING`	The value is converted.
	`DINT_TO_DWORD, DINT_TO_TIME`	The value is transferred to the target data type.
UDInt	`UDINT_TO_BOOL`	The least significant bit is transferred into the destination data type.
	`UDINT_TO_BYTE, UDINT_TO_WORD, UDINT_TO_SINT, UDINT_TO_USINT, UDINT_TO_INT, UDINT_TO_UINT, UDINT_TO_DINT, UDINT_TO_REAL, UDINT_TO_LREAL, UDINT_TO_CHAR, UDINT_TO_STRING`	The value is converted.
	`UDINT_TO_DWORD, UDINT_TO_TOD`	The value is transferred to the target data type.

Table 8- 107 Conversion from a Real number (Real or LReal)

Data type	Instruction	Result
Real	`REAL_TO_DWORD, REAL_TO_LREAL`	The value is transferred to the target data type.
	`REAL_TO_SINT, REAL_TO_USINT, REAL_TO_INT, REAL_TO_UINT, REAL_TO_DINT, REAL_TO_UDINT, REAL_TO_STRING`	The value is converted.
LReal	`LREAL_TO_SINT, LREAL_TO_USINT, LREAL_TO_INT, LREAL_TO_UINT, LREAL_TO_DINT, LREAL_TO_UDINT, LREAL_TO_REAL, LREAL_TO_STRING`	The value is converted.

Table 8- 108 Conversion from Time, DTL, TOD or Date

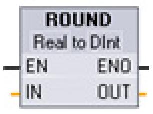
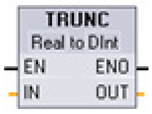
Data type	Instruction	Result
Time	`TIME_TO_DINT`	The value is transferred to the target data type.
DTL	`DTL_TO_DATE, DTL_TO_TOD`	The value is converted.
TOD	`TOD_TO_UDINT`	The value is converted.
Date	`DATE_TO_UINT`	The value is converted.

Table 8- 109 Conversion from a Char or String

Data type	Instruction	Result
Char	`CHAR_TO_SINT, CHAR_TO_USINT, CHAR_TO_INT, CHAR_TO_UINT, CHAR_TO_DINT, CHAR_TO_UDINT`	The value is converted.
	`CHAR_TO_STRING`	The value is transferred to the first character of the string.
String	`STRING_TO_SINT, STRING_TO_USINT, STRING_TO_INT, STRING_TO_UINT, STRING_TO_DINT, STRING_TO_UDINT, STRING_TO_REAL, STRING_TO_LREAL`	The value is converted.
	`STRING_TO_CHAR`	The first character of the string is copied to the Char.

8.7.3 ROUND (Round numerical value) and TRUNC (Truncate numerical value) instructions

Table 8- 110 ROUND and TRUNC instructions

LAD / FBD	SCL	Description
	`out := ROUND (in);`	Converts a real number to an integer. For LAD/FBD, you click the "???" in the instruction box to select the data type for the output, for example "DInt". For SCL, the default data type for the output of the ROUND instruction is DINT. To round to another output data type, enter the instruction name with the explicit name of the data type, for example, ROUND_REAL or ROUND_LREAL. The real number fraction is rounded to the nearest integer value (IEEE - round to nearest). If the number is exactly one-half the span between two integers (for example, 10.5), then the number is rounded to the even integer. For example: • ROUND (10.5) = 10 • ROUND (11.5) = 12
	`out := TRUNC(in);`	TRUNC converts a real number to an integer. The fractional part of the real number is truncated to zero (IEEE - round to zero).

[1] For LAD and FBD: Click the "???" (by the instruction name) and select a data type from the drop-down menu.

Table 8- 111 Data types for the parameters

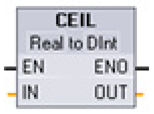
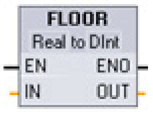
Parameter	Data type	Description
IN	Real, LReal	Floating point input
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	Rounded or truncated output

Table 8- 112 ENO status

ENO	Description	Result OUT
1	No error	Valid result
0	IN is +/- INF or +/- NaN	+/- INF or +/- NaN

8.7.4 CEIL and FLOOR (Generate next higher and lower integer from floating-point number) instructions

Table 8- 113 CEIL and FLOOR instructions

LAD / FBD	SCL	Description
	`out := CEIL(in);`	Converts a real number (Real or LReal) to the closest integer greater than or equal to the selected real number (IEEE "round to +infinity").
	`out := FLOOR(in);`	Converts a real number (Real or LReal) to the closest integer smaller than or equal to the selected real number (IEEE "round to -infinity").

[1] For LAD and FBD: Click the "???" (by the instruction name) and select a data type from the drop-down menu.

Table 8- 114 Data types for the parameters

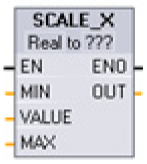
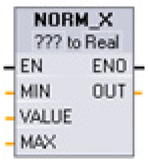
Parameter	Data type	Description
IN	Real, LReal	Floating point input
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	Converted output

Table 8- 115 ENO status

ENO	Description	Result OUT
1	No error	Valid result
0	IN is +/- INF or +/- NaN	+/- INF or +/- NaN

8.7.5 SCALE_X (Scale) and NORM_X (Normalize) instructions

Table 8- 116 SCALE_X and NORM_X instructions

LAD / FBD	SCL	Description
 SCALE_X Real to ??? EN ENO MIN OUT VALUE MAX	`out :=SCALE_X(min:=_in_, value:=_in_, max:=_in_);`	Scales the normalized real parameter VALUE where (0.0 <= VALUE <= 1.0) in the data type and value range specified by the MIN and MAX parameters: OUT = VALUE (MAX - MIN) + MIN
 NORM_X ??? to Real EN ENO MIN OUT VALUE MAX	`out :=NORM_X(min:=_in_, value:=_in_, max:=_in_);`	Normalizes the parameter VALUE inside the value range specified by the MIN and MAX parameters: OUT = (VALUE - MIN) / (MAX - MIN), where (0.0 <= OUT <= 1.0)

1 For LAD and FBD: Click the "???" and select a data type from the drop-down menu.

Table 8- 117 Data types for the parameters

Parameter	Data type[1]	Description
MIN	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	Input minimum value for range
VALUE	SCALE_X: Real, LReal NORM_X: SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	Input value to scale or normalize
MAX	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	Input maximum value for range
OUT	SCALE_X: SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal NORM_X: Real, LReal	Scaled or normalized output value

1 For SCALE_X: Parameters MIN, MAX, and OUT must be the same data type.
For NORM_X: Parameters MIN, VALUE, and MAX must be the same data type.

Note

SCALE_X parameter VALUE should be restricted to (0.0 <= VALUE <= 1.0)

If parameter VALUE is less than 0.0 or greater than 1.0:

- The linear scaling operation can produce OUT values that are less than the parameter MIN value or above the parameter MAX value for OUT values that fit within the value range of the OUT data type. SCALE_X execution sets ENO = TRUE for these cases.
- It is possible to generate scaled numbers that are not within the range of the OUT data type. For these cases, the parameter OUT value is set to an intermediate value equal to the least-significant portion of the scaled real number prior to final conversion to the OUT data type. SCALE_X execution sets ENO = FALSE in this case.

NORM_X parameter VALUE should be restricted to (MIN <= VALUE <= MAX)

If parameter VALUE is less than MIN or greater than MAX, the linear scaling operation can produce normalized OUT values that are less than 0.0 or greater than 1.0. NORM_X execution sets ENO = TRUE in this case.

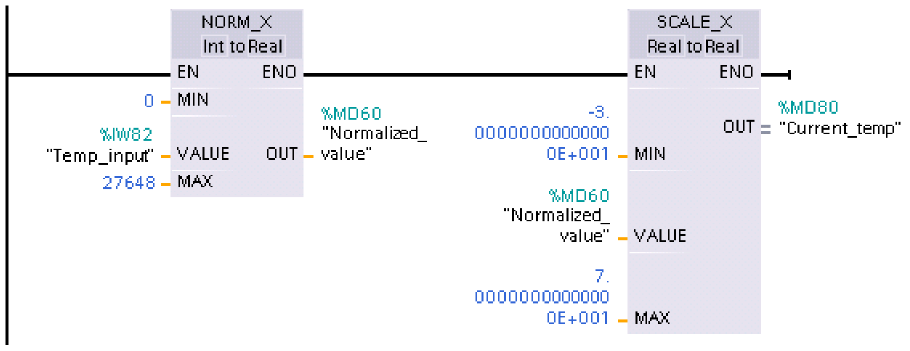
Table 8- 118 ENO status

ENO	Condition	Result OUT
1	No error	Valid result
0	Result exceeds valid range for the OUT data type	Intermediate result: The least-significant portion of a real number prior to final conversion to the OUT data type.
0	Parameters MAX <= MIN	SCALE_X: The least-significant portion of the Real number VALUE to fill up the OUT size. NORM_X: VALUE in VALUE data type extended to fill a double word size.
0	Parameter VALUE = +/- INF or +/- NaN	VALUE is written to OUT

Example (LAD): normalizing and scaling an analog input value

An analog input from an analog signal module or signal board using input in current is in the range 0 to 27648 for valid values. Suppose an analog input represents a temperature where the 0 value of the analog input represents -30.0 degrees C and 27648 represents 70.0 degrees C.

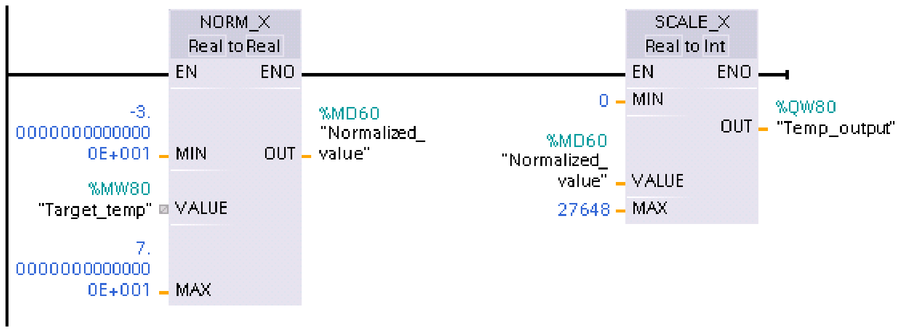
To transform the analog value to the corresponding engineering units, normalize the input to a value between 0.0 and 1.0, and then scale it between -30.0 and 70.0. The resulting value is the temperature represented by the analog input in degrees C:



Note that if the analog input was from an analog signal module or signal board using voltage, the MIN value for the NORM_X instruction would be -27648 instead of 0.

Example (LAD): normalizing and scaling an analog output value

An analog output to be set in an analog signal module or signal board using output in current must be in the range 0 to 27648 for valid values. Suppose an analog output represents a temperature setting where the 0 value of the analog input represents -30.0 degrees C and 27648 represents 70.0 degrees C. To convert a temperature value in memory that is between -30.0 and 70.0 to a value for the analog output in the range 0 to 27648, you must normalize the value in engineering units to a value between 0.0 and 1.0, and then scale it to the range of the analog output, 0 to 27648:



Note that if the analog output was for an analog signal module or signal board using voltage, the MIN value for the SCALE_X instruction would be -27648 instead of 0.

Additional information on analog input representations (Page 1194) and analog output representations (Page 1195) in both voltage and current can be found in the Technical Specifications.

8.7.6 Variant conversion instructions

8.7.6.1 VARIANT_TO_DB_ANY (Convert VARIANT to DB_ANY) instruction

You use the SCL "Convert VARIANT to DB_ANY" instruction to read the operand at the IN parameter and convert it to the data type DB_ANY. The IN parameter is of the Variant data type and represents either an instance data block or an ARRAY data block. When you create the program, you do not need to know which data block corresponds to the IN parameter. The instruction reads the data block number during runtime and writes it to the operand at the RET_VAL parameter.

Table 8- 119 VARIANT_TO_DB_ANY instruction

LAD / FBD	SCL	Description

Not available	`RET_VAL :=`
	`VARIANT_TO_DB_ANY(`
	`    in := _variant_in_,`
	`    err => _int_out_);`
		Reads the operand from the Variant IN parameter and stores it in the function result, which is of the type DB_ANY

Table 8- 120 Parameters for the VARIANT_TO_DB_ANY instruction

Parameter	Data type	Description
IN	Variant	Variant that represents and instance data block or an array data block
RET_VAL	DB_ANY	Output DB_ANY data type that contains the converted data block number
ERR	Int	Error information

Table 8- 121 ENO status

ENO	Condition	Result
1	No error	Instruction converts the input Variant and stores it in the DB_ANY function output
0	Enable input EN has the signal state "0" or the IN parameter is invalid.	Instruction does nothing.

Table 8- 122 Error output codes for the VARIANT_TO_DB_ANY instruction

Err (W#16#...)	Description
0000	No error
252C	The Variant data type at IN parameter has the value 0. The CPU changes to STOP mode.
8131	The data block does not exist or is too short (first access).
8132	The data block is too short and not an Array data block (second access).
8134	The data block is write-protected
8150	The data type Variant at parameter IN provides the value "0". To receive this error message, the "Handle errors within block" block property must be activated. Otherwise the CPU changes to STOP mode and sends the error code 16#252C
8154	The data block has the incorrect data type.
*You can display error codes in the program editor as integer or hexadecimal values.	

8.7.6.2 DB_ANY_TO_VARIANT (Convert DB_ANY to VARIANT) instruction

You use the SCL "Convert DB_ANY to VARIANT" to read the number of a data block which meets the requirements listed below. The operand at the IN parameter has the data type DB_ANY, which means you do not need to know during program creation which data block whose number is to be read will be specified. The data block number is read during runtime and written by means of a VARIANT pointer to the operand specified at the RET_VAL parameter.

Table 8- 123 DB_ANY_TO_VARIANT instruction

LAD / FBD	SCL	Description

Not available	`RET_VAL := DB_ANY_TO_VARIANT(`
	`    in := _db_any_in_,`
	`    err => _int_out_);`
		Reads the data block number from the Variant IN parameter and stores it in the function result, which is of the type Variant

Table 8- 124 Parameters for the DB_ANY_TO_VARIANT instruction

Parameter	Data type	Description
IN	DB_ANY	Variant that contains the data block number
RET_VAL	Variant	Output DB_ANY data type that contains the converted data block number
ERR	Int	Error information

Table 8- 125 ENO status

ENO	Condition	Result
1	No error	Instruction converts the data block number in the variant and stores it in the function DB_ANY output
0	Enable input EN has the signal state "0" or the IN parameter is invalid.	Instruction does nothing.

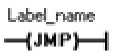
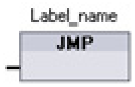
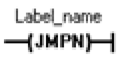
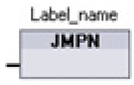
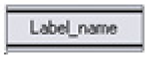
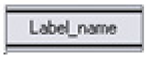
Table 8- 126 Error output codes for the DB_ANY_TO_VARIANT instruction

Err (W#16#...)	**Description**
0000	No error
8130	The number of the data block is 0.
8131	The data block does not exist or is too short.
8132	The data block is too short and not an Array data block.
8134	The data block is write-protected.
8154	The data block has the incorrect data type.
8155	Unknown type code
*You can display error codes in the program editor as integer or hexadecimal values.	

8.8 Program control operations

8.8.1 JMP (Jump if RLO = 1), JMPN (Jump if RLO = 0), and Label (Jump label) instructions

Table 8- 127 JMP, JMPN, and LABEL instruction

LAD	FBD	SCL	Description
		See the GOTO (Page 304) statement.	Jump if RLO (result of logic operation) = 1: If there is power flow to a JMP coil (LAD), or if the JMP box input is true (FBD), then program execution continues with the first instruction following the specified label.
			Jump if RLO = 0: If there is no power flow to a JMPN coil (LAD), or if the JMPN box input is false (FBD), then program execution continues with the first instruction following the specified label.
			Destination label for a JMP or JMPN jump instruction.

1 You create your label names by typing in the LABEL instruction directly. Use the parameter helper icon to select the available label names for the JMP and JMPN label name field. You can also type a label name directly into the JMP or JMPN instruction.

Table 8- 128 Data types for the parameters

Parameter	Data type	Description
Label_name	Label identifier	Identifier for Jump instructions and the corresponding jump destination program label

- Each label must be unique within a code block.
- You can jump within a code block, but you cannot jump from one code block to another code block.
- You can jump forward or backward.
- You can jump to the same label from more than one place in the same code block.

8.8.2 JMP_LIST (Define jump list) instruction

Table 8- 129 JMP_LIST instruction

LAD / FBD	SCL	Description

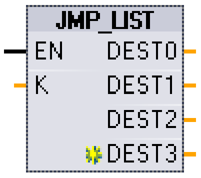
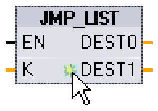
	`CASE k OF`
	`      0: GOTO dest0;`
	`      1: GOTO dest1;`
	`      2: GOTO dest2;`
	`     [n: GOTO destn;]`
	`END_CASE;`
		The JMP_LIST instruction acts as a program jump distributor to control the execution of program sections. Depending on the value of the K input, a jump occurs to the corresponding program label. Program execution continues with the program instructions that follow the destination jump label. If the value of the K input exceeds the number of labels - 1, then no jump occurs and processing continues with the next program network.

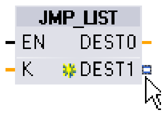
Table 8- 130 Data types for parameters

Parameter	Data type	Description
K	UInt	Jump distributor control value
DEST0, DEST1, .., DESTn.	Program Labels	Jump destination labels corresponding to specific K parameter values: If the value of K equals 0, then a jump occurs to the program label assigned to the DEST0 output. If the value of K equals 1, then a jump occurs to the program label assigned to the DEST1 output, and so on. If the value of the K input exceeds the (number of labels - 1), then no jump occurs and processing continues with the next program network.

For LAD and FBD: When the JMP_LIST box is first placed in your program, there are two jump label outputs. You can add or delete jump destinations.



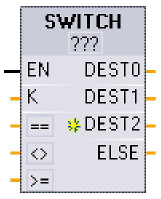
Click the create icon inside the box (on the left of the last DEST parameter) to add new outputs for jump labels.



- Right-click on an output stub and select the "Insert output" command.
- Right-click on an output stub and select the "Delete" command.

8.8.3 SWITCH (Jump distributor) instruction

Table 8- 131 SWITCH instruction

LAD / FBD	SCL	Description
 SWITCH ??? EN DEST0 K DEST1 == DEST2 <> ELSE >=	`Not available`	The SWITCH instruction acts as a program jump distributor to control the execution of program sections. Depending on the result of comparisons between the value of the K input and the values assigned to the specified comparison inputs, a jump occurs to the program label that corresponds to the first comparison test that is true. If none of the comparisons is true, then a jump to the label assigned to ELSE occurs. Program execution continues with the program instructions that follow the destination jump label.

[1] For LAD and FBD: Click below the box name and select a data type from the drop-down menu.

[2] For SCL: Use an IF-THEN set of comparisons.

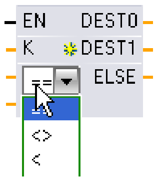
Table 8- 132 Data types for parameters

Parameter	Data type[1]	Description
K	UInt	Common comparison value input
==, <>, <, <=, >. >=	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Time, TOD, Date	Separate comparison value inputs for specific comparison types
DEST0, DEST1, .., DESTn, ELSE	Program Labels	Jump destination labels corresponding to specific comparisons: The comparison input below and next to the K input is processed first and causes a jump to the label assigned to DEST0, if the comparison between the K value and this input is true. The next comparison test uses the next input below and causes a jump to the label assigned to DEST1, if the comparison is true, The remaining comparisons are processed similarly and if none of the comparisons are true, then a jump to the label assigned to the ELSE output occurs.

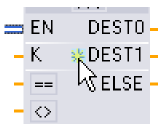
[1] The K input and comparison inputs (==, <>, <, <=, >, >=) must be the same data type.

Adding inputs, deleting inputs, and specifying comparison types

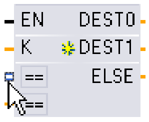
When the LAD or FBD SWITCH box is first placed in your program there are two comparison inputs. You can assign comparison types and add inputs/jump destinations, as shown below.



Click a comparison operator inside the box and select a new operator from the drop-down list.



Click the create icon inside the box (to the left of the last DEST parameter) to add new comparison-destination parameters.



- Right-click on an input stub and select the "Insert input" command.
- Right-click on an input stub and select the "Delete" command.

Table 8- 133 SWITCH box data type selection and allowed comparison operations

Data type	Comparison	Operator syntax
Byte, Word, DWord	Equal	==
	Not equal	<>
SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Time, TOD, Date	Equal	==
	Not equal	<>
	Greater than or equal	>=
	Less than or equal	<=
	Greater than	>
	Less than	<

SWITCH box placement rules

- No LAD/FBD instruction connection in front of the compare input is allowed.
- There is no ENO output, so only one SWITCH instruction is allowed in a network and the SWITCH instruction must be the last operation in a network.

8.8.4 RET (Return) instruction

The optional RET instruction is used to terminate the execution of the current block. If and only if there is power flow to the RET coil (LAD) or if the RET box input is true (FBD), then program execution of the current block will end at that point and instructions beyond the RET instruction will not be executed. If the current block is an OB, the "Return_Value" parameter is ignored. If the current block is a FC or FB, the value of the "Return_Value" parameter is passed back to the calling routine as the ENO value of the called box.

You are not required to use a RET instruction as the last instruction in a block; this is done automatically for you. You can have multiple RET instructions within a single block.

For SCL, see the RETURN (Page 304) statement.

Table 8- 134 Return_Value (RET) execution control instruction

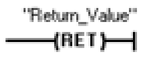
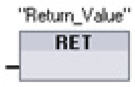
LAD	FBD	SCL	Description
		`RETURN;`	Terminates the execution of the current block

Table 8- 135 Data types for the parameters

Parameter	Data type	Description
Return_Value	Bool	The "Return_value" parameter of the RET instruction is assigned to the ENO output of the block call box in the calling block.

Sample steps for using the RET instruction inside an FC code block:

1. Create a new project and add an FC:
2. Edit the FC:
   - Add instructions from the instruction tree.
   - Add a RET instruction, including one of the following for the "Return_Value" parameter:

     TRUE, FALSE, or a memory location that specifies the required return value.
   - Add more instructions.
3. Call the FC from MAIN [OB1].

The EN input on the FC box in the MAIN code block must be true to begin execution of the FC.

The value specified by the RET instruction in the FC will be present on the ENO output of the FC box in the MAIN code block following execution of the FC for which power flow to the RET instruction is true.

8.8.5 ENDIS_PW (Enable/disable CPU passwords) instruction

Table 8- 136 ENDIS_PW instruction

LAD / FBD	SCL	Description

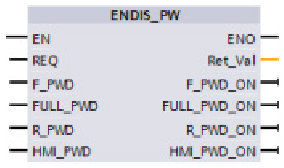
 ENDIS_PW
EN
ENO
REQ
Ret_Val
F_PWD
F_PWD_ON
FULL_PWD
FULL_PWD_ON
R_PWD
R_PWD_ON
HMI_PWD
HMI_PWD_ON
	`ENDIS_PW(`
	`req:=_bool_in_,`
	`f_pwd:=_bool_in_,`
	`full_pwd:=_bool_in_,`
	`r_pwd:=_bool_in_,`
	`hmi_pwd:=_bool_in_,`
	`f_pwd_on=>_bool_out_,`
	`full_pwd_on=>_bool_out_,`
	`r_pwd_on=>_bool_out_,`
	`hmi_pwd_on=>_bool_out_);`
		The ENDIS_PW instruction can allow and disallow client connections to a S7-1200 CPU, even when the client can provide the correct password.
		This instruction does not disallow Web server passwords.

Table 8- 137 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	Perform function if REQ=1
F_PWD	IN	Bool	Fail-safe password: Allow (=1) or disallow (=0)
FULL_PWD	IN	Bool	Full access password: Allow (=1) or disallow (=0) full access password
R_PWD	IN	Bool	Read access password: Allow (=1) or disallow (=0)
HMI_PWD	IN	Bool	HMI password: Allow (=1) or disallow (=0)
F_PWD_ON	OUT	Bool	Fail-safe password status: Allowed (=1) or disallowed (=0)
FULL_PWD_ON	OUT	Bool	Full access password status: Allowed (=1) or disallowed (=0)
R_PWD_ON	OUT	Bool	Read only password status: Allowed (=1) or disallowed (=0)
HMI_PWD_ON	OUT	Bool	HMI password status: Allowed (=1) or disallowed (=0)
Ret_Val	OUT	Word	Function result

Calling ENDIS_PW with REQ=1 disallows password types where the corresponding password input parameter is FALSE. Each password type can be allowed or disallowed independently. For example, if the fail-safe password is allowed and all other passwords disallowed, then you can restrict CPU access to a small group of employees.

ENDIS_PW is executed synchronously in a program scan and the password output parameters always show the current state of password allowance independent of the input parameter REQ. All passwords that you set to allow must be changeable to disallowed/allowed. Otherwise, an error is returned and all passwords are allowed that were allowed before ENDIS_PW execution. This means that in a standard CPU (where the fail-safe password is not configured) F_PWD must always be set to 1, to result in a return value of 0. In this case, F_PWD_ON is always 1.

Note

- ENDIS_PW execution can block the access of HMI devices, if the HMI password is disallowed.
- Client sessions that were authorized prior to ENDIS_PW execution remain unchanged by ENDIS_PW execution.

After a power-up, CPU access is restricted by passwords previously defined in the regular CPU protection configuration. The ability to disallow a valid password must be re-established with a new ENDIS_PW execution. However, if ENDIS_PW is immediately executed and necessary passwords are disallowed, then TIA portal access can be locked out. You can use a timer instruction to delay ENDIS_PW execution and allow time to enter passwords, before the passwords become disallowed.

Note

Restoring a CPU that locks out TIA portal communication

Refer to the "Recovery from a lost password (Page 144)" topic for details about how to erase the internal load memory of a PLC using a memory card.

An operating mode change to STOP caused by errors, STP execution or STEP 7 does not abolish the protection. The protection is valid until the CPU is power cycled. See the following table for details.

Action	Operating mode	ENDIS_PW password control
After memory reset from STEP 7	STOP	Active: Disallowed passwords remain disallowed.
After powering on, or changing a memory card	STOP	Off: No passwords are disallowed.
After ENDIS_PW execution in a program cycle or startup OB	STARTUP, RUN	Active: Passwords are disallowed according to ENDIS_PW parameters
After change of the operating mode from RUN or STARTUP to STOP through STP instruction, error, or STEP 7	STOP	Active: Disallowed passwords remain disallowed

Note

Password protect CPU access levels with strong passwords. Strong passwords are at least ten characters in length, mix letters, numbers, and special characters, are not words that can be found in a dictionary, and are not names or identifiers that can be derived from personal information. Keep the password secret and change it frequently.

Table 8- 138 Condition codes

RET_VAL (W#16#...)	Description
0000	No error
8090	The instruction is not supported.
80D0	The password for fail-safe is not configured.
80D1	The password for read/write access is not configured.
80D2	The password for read access is not configured.
80D3	The password for HMI access is not configured.

8.8.6 RE_TRIGR (Restart cycle monitoring time) instruction

Table 8- 139 RE_TRIGR instruction

LAD / FBD	SCL	Description
RE_TRIGR EN ENO	`RE_TRIGR();`	RE_TRIGR (Re-trigger scan time watchdog) is used to extend the maximum time allowed before the scan cycle watchdog timer generates an error.

Use the RE_TRIGR instruction to restart the scan cycle monitoring timer during a single scan cycle. This has the effect of extending the allowed maximum scan cycle time by one maximum cycle time period, from the last execution of the RE_TRIGR function.

Note

Prior to S7-1200 CPU firmware version 2.2, RE_TRIGR was restricted to execution from a program cycle OB and could be used to extend the PLC scan time indefinitely. ENO = FALSE and the watchdog timer is not reset when RE_TRIGR was executed from a start up OB, an interrupt OB, or an error OB.

For firmware version 2.2 and later, RE_TRIGR can be executed from any OB (including start up, interrupt, and error OBs). However, the PLC scan can only be extended by a maximum of 10x the configured maximum cycle time.

Setting the PLC maximum cycle time

Configure the value for maximum scan cycle time in the Device configuration for "Cycle time".

Table 8- 140 Cycle time values

Cycle time monitor	Minimum value	Maximum value	Default value
Maximum cycle time	1 ms	6000 ms	150 ms

Watchdog timeout

If the maximum scan cycle timer expires before the scan cycle has been completed, an error is generated. If the user program includes a time error interrupt OB (OB 80), the CPU executes the time error interrupt OB, which can include program logic to create a special reaction.

If the user program does not include a time error interrupt OB, the first timeout condition is ignored and the CPU remains in RUN mode. If a second maximum scan time timeout occurs in the same program scan (2 times the maximum cycle time value), then an error is triggered that causes a transition to STOP mode.

In STOP mode, your program execution stops while CPU system communications and system diagnostics continue.

8.8.7 STP (Exit program) instruction

Table 8- 141 STP instruction

LAD / FBD	SCL	Description
	`STP();`	STP puts the CPU in STOP mode. When the CPU is in STOP mode, the execution of your program and physical updates from the process image are stopped.

For more information see: Configuring the outputs on a RUN-to-STOP transition (Page 108).

If EN = TRUE, then the CPU goes to STOP mode, the program execution stops, and the ENO state is meaningless. Otherwise, EN = ENO = 0.

8.8.8 GET_ERROR and GET_ERROR_ID (Get error and error ID locally) instructions

The get error instructions provide information about program block execution errors. If you add a GET_ERROR or GET_ERROR_ID instruction to your code block, you can handle program errors within your program block.

GET_ERROR

Table 8- 142 GET_ERROR instruction

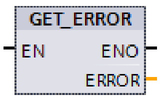
LAD / FBD	SCL	Description
	`GET_ERROR(_out_);`	Indicates that a local program block execution error has occurred and fills a predefined error data structure with detailed error information.

Table 8- 143 Data types for the parameters

Parameter	Data type	Description
ERROR	ErrorStruct	Error data structure: You can rename the structure, but not the members within the structure.

Table 8- 144 Elements of the ErrorStruct data structure

Structure components		Data type	Description
ERROR_ID		Word	Error ID
FLAGS		Byte	Shows if an error occurred during a block call. • 16#01: Error during a block call. • 16#00: No error during a block call.
REACTION		Byte	Default reaction: • 0: Ignore (write error), • 1: Continue with substitute value "0" (read error), • 2: Skip instruction (system error)
CODE_ADDRESS		CREF	Information about the address and type of block
	BLOCK_TYPE	Byte	Type of block where the error occurred: • 1: OB • 2: FC • 3: FB
	CB_NUMBER	UInt	Number of the code block
	OFFSET	UDInt	Reference to the internal memory
MODE		Byte	Access mode: Depending on the type of access, the following information can be output: Mode | (A) | (B) | (C) | (D) | (E)

Structure components		Data type	Description					
			0					
			1					Offset
			2			Area		
			3	Location	Scope		Number	
			4			Area		Offset
			5			Area	DB no.	Offset
			6	PtrNo. /Acc		Area	DB no.	Offset
			7	PtrNo. / Acc	Slot No. / Scope	Area	DB no.	Offset
OPERAND_NUMBER		UInt	Operand number of the machine command					
POINTER_NUMBER_ LOCATION		UInt	(A) Internal pointer					
SLOT_NUMBER_SCOPE		UInt	(B) Storage area in internal memory					
DATA_ADDRESS		NREF	Information about the address of an operand					
	AREA	Byte	(C) Memory area: • L: 16#40 – 4E, 86, 87, 8E, 8F, C0 – CE • I: 16#81 • Q: 16#82 • M: 16#83 • DB: 16#84, 85, 8A, 8B					
	DB_NUMBER	UInt	(D) Number of the data block					
	OFFSET	UDInt	(E) Relative address of the operand					

GET_ERROR_ID

Table 8- 145 GetErrorID instruction

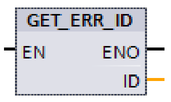
LAD / FBD	SCL	Description
	`GET_ERR_ID();`	Indicates that a program block execution error has occurred and reports the ID (identifier code) of the error.

Table 8- 146 Data types for the parameters

Parameter	Data type	Description
ID	Word	Error identifier values for the ErrorStruct ERROR_ID member

Table 8- 147 Error_ID values

ERROR_ID hexa-decimal	ERROR_ID decimal	Program block execution error
0	0	No error
2520	9504	Corrupted string
2522	9506	Operand out of range read error
2523	9507	Operand out of range write error
2524	9508	Invalid area read error
2525	9509	Invalid area write error
2528	9512	Data alignment read error (incorrect bit alignment)
2529	9513	Data alignment write error (incorrect bit alignment)
252C	9516	Uninitialized pointer error
2530	9520	DB write protected
2533	9523	Invalid pointer used
2538	9528	Access error: DB does not exist
2539	9529	Access error: Wrong DB used
253A	9530	Global DB does not exist
253C	9532	Wrong version or FC does not exist
253D	9533	Instruction does not exist
253E	9534	Wrong version or FB does not exist
253F	9535	Instruction does not exist
2550	9552	Access error: DB does not exist
2575	9589	Program nesting depth error
2576	9590	Local data allocation error
2942	10562	Physical input point does not exist
2943	10563	Physical output point does not exist

Operation

By default, the CPU responds to a block execution error by logging an error in the diagnostics buffer. However, if you place one or more GET_ERROR or GET_ERROR_ID instructions within a code block, this block is now set to handle errors within the block. In this case, the CPU does not log an error in the diagnostics buffer. Instead, the error information is reported in the output of the GET_ERROR or GET_ERROR_ID instruction. You can read the detailed error information with the GET_ERROR instruction, or read just the error identifier with GET_ERROR_ID instruction. Normally the first error is the most important, with the following errors only consequences of the first error.

The first execution of a GET_ERROR or GET_ERROR_ID instruction within a block returns the first error detected during block execution. This error could have occurred anywhere between the start of the block and the execution of either GET_ERROR or GET_ERROR_ID. Subsequent executions of either GET_ERROR or GET_ERROR_ID return the first error since the previous execution of GET_ERROR or GET_ERROR_ID. The history of errors is not saved, and execution of either instruction will re-arm the PLC system to catch the next error.

The ErrorStruct data type used by the GET_ERROR instruction can be added in the data block editor and block interface editors, so your program logic can access these values. Select ErrorStruct from the data type drop-down list to add this structure. You can create multiple ErrorStruct elements by using unique names. The members of an ErrorStruct cannot be renamed.

Error condition indicated by ENO

If EN = TRUE and GET_ERROR or GET_ERROR_ID executes, then:

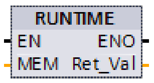
- ENO = TRUE indicates a code block execution error occurred and error data is present
- ENO = FALSE indicates no code block execution error occurred

You can connect error reaction program logic to ENO which activates after an error occurs. If an error exists, then the output parameter stores the error data where your program has access to it.

GET_ERROR and GET_ERROR_ID can be used to send error information from the currently executing block (called block) to a calling block. Place the instruction in the last network of the called block program to report the final execution status of the called block.

8.8.9 RUNTIME (Measure program runtime) instruction

Table 8- 148 RUNTIME instruction

LAD / FBD	SCL	Description
	`Ret_Val := RUNTIME( _lread_inout_);`	Measures the runtime of the entire program, individual blocks, or command sequences.

If you want to measure the runtime of your entire program, call the instruction "Measure program runtime" in OB1. Measurement of the runtime is started with the first call and the output RET_VAL returns the runtime of the program after the second call. The measured runtime includes all CPU processes that can occur during the program execution, for example, interruptions caused by higher-level events or communication. The instruction "Measure program runtime" reads an internal counter of the CPU and write the value to the IN-OUT parameter MEM. The instruction calculates the current program runtime according to the internal counter frequency and writes it to output RET_VAL.

If you want to measure the runtime of individual blocks or individual command sequences, you need three separate networks. Call the instruction "Measure program runtime" in an individual network within your program. You set the starting point of the runtime measurement with this first call of the instruction. Then you call the required program block or the command sequence in the next network. In another network, call the "Measure program runtime" instruction a second time and assign the same memory to the IN-OUT parameter MEM as you did during the first call of the instruction. The "Measure program runtime" instruction in the third network reads an internal CPU counter and calculates the current runtime of the program block or the command sequence according to the internal counter frequency and writes it to the output RET_VAL.

The Measure program runtime" instruction uses an internal high-frequency counter to calculate the time. If the counter overruns, the instruction returns values <= 0.0. Ignore these runtime values.

Note

The CPU cannot exactly determine the runtime of a command sequence, because the sequence of instructions within a command sequence changes during optimized compilation of the program.

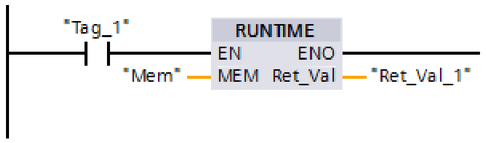
Table 8- 149 Data types for the parameters

Parameter	Data type	Description
MEM	LReal	Starting poing of the runtime measurement
RET_VAL	LReal	Measured runtime in seconds

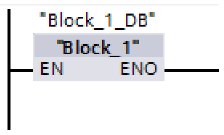
Example: RUNTIME instruction

The following example shows the use of the RUNTIME instruction to measure the execution time of a function block:

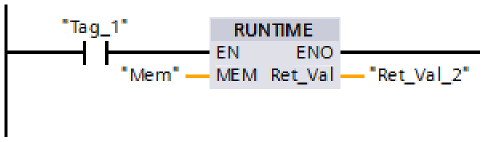
Network 1:



Network 2:



Network 3:



When the "Tag_1" operand in network 1 has the signal state "1", the RUNTIME instruction executes. The starting point for the runtime measurement is set with the first call of the instruction and buffered as reference for the second call of the instruction in the "Mem" operand.

The function block FB1 executes in network 2.

When the FB1 program block completes and the "Tag_1" operand has the signal state "1", the RUNTIME instruction in network 3 executes. The second call of the instruction calculates the runtime of the program block and writes the result to the output RET_VAL_2.

8.8.10 SCL program control statements

8.8.10.1 Overview of SCL program control statements

Structured Control Language (SCL) provides three types of program control statements for structuring your user program:

- Selective statements: A selective statement enables you to direct program execution into alternative sequences of statements.
- Loops: You can control loop execution using iteration statements. An iteration statement specifies which parts of a program should be iterated depending on certain conditions.
- Program jumps: A program jump means an immediate jump to a specified jump destination and therefore to a different statement within the same block.

These program control statements use the syntax of the PASCAL programming language.

Table 8- 150 Types of SCL program control statements

Program control statement		Description
Selective	IF-THEN statement (Page 296)	Enables you to direct program execution into one of two alternative branches, depending on a condition being TRUE or FALSE
	CASE statement (Page 297)	Enables the selective execution into 1 of *n* alternative branches, based on the value of a variable
Loop	FOR statement (Page 299)	Repeats a sequence of statements for as long as the control variable remains within the specified value range
	WHILE-DO statement (Page 300)	Repeats a sequence of statements while an execution condition continues to be satisfied
	REPEAT-UNTIL statement (Page 301)	Repeats a sequence of statements until a terminate condition is met
Program jump	CONTINUE statement (Page 302)	Stops the execution of the current loop iteration
	EXIT statement (Page 303)	Exits a loop at any point regardless of whether the terminate condition is satisfied or not
	GOTO statement (Page 304)	Causes the program to jump immediately to a specified label
	RETURN statement (Page 304)	Causes the program to exit the block currently being executed and to return to the calling block

8.8.10.2 IF-THEN statement

The IF-THEN statement is a conditional statement that controls program flow by executing a group of statements, based on the evaluation of a Bool value of a logical expression. You can also use brackets to nest or structure the execution of multiple IF-THEN statements.

Table 8- 151 Elements of the IF-THEN statement

SCL	Description

`IF "condition" THEN`
`     statement_A;`
`     statement_B;`
`     statement_C;`
`     ;`
	If "condition" is TRUE or 1, then execute the following statements until encountering the END_IF statement.
	If "condition" is FALSE or 0, then skip to END_IF statement (unless the program includes optional ELSIF or ELSE statements).

`[ELSIF "condition-n" THEN`
`    statement_N;`
`    ;]`
	The optional ELSEIF[1] statement provides additional conditions to be evaluated. For example: If "condition" in the IF-THEN statement is FALSE, then the program evaluates "condition-n". If "condition-n" is TRUE, then execute "statement_N".

`[ELSE`
`    statement_X;`
`    ;]`
	The optional ELSE statement provides statements to be executed when the "condition" of the IF-THEN statement is FALSE.

`END_IF;`	The END_IF statement terminates the IF-THEN instruction.

[1] You can include multiple ELSIF statements within one IF-THEN statement.

Table 8- 152 Variables for the IF-THEN statement

Variables	Description
"condition"	Required. The logical expression is either TRUE (1) or FALSE (0).
"statement_A"	Optional. One or more statements to be executed when "condition" is TRUE.
"condition-n"	Optional. The logical expression to be evaluated by the optional ELSIF statement.
"statement_N"	Optional. One or more statements to be executed when "condition-n" of the ELSIF statement is TRUE.
"statement_X"	Optional. One or more statements to be executed when "condition" of the IF-THEN statement is FALSE.

An IF statement is executed according to the following rules:

- The first sequence of statements whose logical expression = TRUE is executed. The remaining sequences of statements are not executed.
- If no Boolean expression = TRUE, the sequence of statements introduced by ELSE is executed (or no sequence of statements if the ELSE branch does not exist).
- Any number of ELSIF statements can exist.

Note

Using one or more ELSIF branches has the advantage that the logical expressions following a valid expression are no longer evaluated in contrast to a sequence of IF statements. The runtime of a program can therefore be reduced.

8.8.10.3 CASE statement

Table 8- 153 Elements of the CASE statement

SCL	Description

`CASE "Test_Value" OF`
`     "ValueList": Statement[; Statement, ...]`
`     "ValueList": Statement[; Statement, ...]`
`[ELSE`
`Else-statement[; Else-statement, ...]]`
`END_CASE;`
	The CASE statement executes one of several groups of statements, depending on the value of an expression.

Table 8- 154 Parameters

Parameter	Description
"Test_Value"	Required. Any numeric expression of data type Int

"ValueList"	Required. A single value or a comma-separated list of values or ranges of values. (Use two periods to define a range of values: 2..8) The following example illustrates the different variants of the value list:
	1: Statement_A;
	2, 4: Statement _B;
	3, 5..7,9: Statement _C;

Statement	Required. One or more statements that are executed when "Test_Value" matches any value in the value list
Else-statement	Optional. One or more statements that are executed if no match with a value of the "ValueList" stated matches

The CASE statement is executed according to the following rules:

- The Test_value expression must return a value of the type Int.
- When a CASE statement is processed, the program checks whether the value of the Test_value expression is contained within a specified list of values. If a match is found, the statement component assigned to the list is executed.
- If no match is found, the program section following ELSE is executed or no statement is executed if the ELSE branch does not exist.

Example: Nested CASE statements

CASE statements can be nested. Each nested case statement must have an associated END_CASE statement.

```
CASE "var1" OF
       1 : #var2 := 'A';
       2 : #var2 := 'B';
ELSE
       CASE "var3" OF

          65..90: #var2 := 'UpperCase';
          97..122: #var2 := 'LowerCase';
       ELSE

          #var2:= 'SpecialCharacter';
       END_CASE;
END_CASE;
```

8.8.10.4 FOR statement

Table 8- 155 Elements of the FOR statement

SCL	Description
`FOR "control_variable" := "begin" TO "end" [BY "increment"] DO statement; ; END_FOR;`	A FOR statement is used to repeat a sequence of statements as long as a control variable is within the specified range of values. The definition of a loop with FOR includes the specification of an initial and an end value. Both values must be the same type as the control variable. You can nest FOR loops. The END_FOR statement refers to the last executed FOR instruction.

```
FOR "control_variable" := "begin" TO "end"
[BY "increment"] DO
     statement;
     ;
END_FOR;
```

Table 8- 156 Parameters

Parameter	Description
"control_variable"	Required. An integer (Int or DInt) that serves as a loop counter
"begin"	Required. Simple expression that specifies the initial value of the control variables
"end"	Required. Simple expression that determines the final value of the control variables
"increment"	Optional. Amount by which a "control variable" is changed after each loop. The "increment" has the same data type as "control variable". If the "increment" value is not specified, then the value of the run tags will be increased by 1 after each loop. You cannot change "increment" during the execution of the FOR statement.

The FOR statement executes as follows:

- At the start of the loop, the control variable is set to the initial value (initial assignment) and each time the loop iterates, it is incremented by the specified increment (positive increment) or decremented (negative increment) until the final value is reached.
- Following each run through of the loop, the condition is checked (final value reached) to establish whether or not it is satisfied. If the end condition is not satisfied, the sequence of statements is executed again, otherwise the loop terminates and execution continues with the statement immediately following the loop.

Rules for formulating FOR statements:

- The control variable may only be of the data type Int or DInt.
- You can omit the statement BY [increment]. If no increment is specified, it is automatically assumed to be +1.

To end the loop regardless of the state of the "condition" expression, use the EXIT statement (Page 303). The EXIT statement executes the statement immediately following the END_FOR statement.

Use the CONTINUE statement (Page 302) to skip the subsequent statements of a FOR loop and to continue the loop with the examination of whether the condition is met for termination.

8.8.10.5 WHILE-DO statement

Table 8- 157 WHILE statement

SCL	Description

`WHILE "condition" DO`
`     Statement;`
`     Statement;`
`     ...;`
`END_WHILE;`
	The WHILE statement performs a series of statements until a given condition is TRUE.
	You can nest WHILE loops. The END_WHILE statement refers to the last executed WHILE instruction.

Table 8- 158 Parameters

Parameter	Description
"condition"	Required. A logical expression that evaluates to TRUE or FALSE. (A "null" condition is interpreted as FALSE.)
Statement	Optional. One or more statements that are executed until the condition evaluates to TRUE.

Note

The WHILE statement evaluates the state of "condition" before executing any of the statements. To execute the statements at least one time regardless of the state of "condition", use the REPEAT statement (Page 301).

The WHILE statement executes according to the following rules:

- Prior to each iteration of the loop body, the execution condition is evaluated.
- The loop body following DO iterates as long as the execution condition has the value TRUE.
- Once the value FALSE occurs, the loop is skipped and the statement following the loop is executed.

To end the loop regardless of the state of the "condition" expression, use the EXIT statement (Page 303). The EXIT statement executes the statement immediately following the END_WHILE statement

Use the CONTINUE statement to skip the subsequent statements of a WHILE loop and to continue the loop with the examination of whether the condition is met for termination.

8.8.10.6 REPEAT-UNTIL statement

Table 8- 159 REPEAT instruction

SCL	Description

`REPEAT`
`     Statement;`
`     ;`
`UNTIL "condition"`
`END_REPEAT;`
	The REPEAT statement executes a group of statements until a given condition is TRUE.
	You can nest REPEAT loops. The END_REPEAT statement always refers to the last executed Repeat instruction.

Table 8- 160 Parameters

Parameter	Description
Statement	Optional. One or more statements that are executed until the condition is TRUE.
"condition"	Required. One or more expressions of the two following ways: A numeric expression or string expression that evaluates to TRUE or FALSE. A "null" condition is interpreted as FALSE.

Note

Before evaluating the state of "condition", the REPEAT statement executes the statements during the first iteration of the loop (even if "condition" is FALSE). To review the state of "condition" before executing the statements, use the WHILE statement (Page 300).

To end the loop regardless of the state of the "condition" expression, use the EXIT statement (Page 303). The EXIT statement executes the statement immediately following the END_REPEAT statement

Use the CONTINUE statement (Page 302) to skip the subsequent statements of a REPEAT loop and to continue the loop with the examination of whether the condition is met for termination.

8.8.10.7 CONTINUE statement

Table 8- 161 CONTINUE statement

SCL	Description

`CONTINUE`
`     Statement;`
`     ;`
	The CONTINUE statement skips the subsequent statements of a program loop (FOR, WHILE, REPEAT) and continues the loop with the examination of whether the condition is met for termination. If this is not the case, the loop continues.

The CONTINUE statement executes according to the following rules:

- This statement immediately terminates execution of a loop body.
- Depending on whether the condition for repeating the loop is satisfied or not the body is executed again or the iteration statement is exited and the statement immediately following is executed.
- In a FOR statement, the control variable is incremented by the specified increment immediately after a CONTINUE statement.

Use the CONTINUE statement only within a loop. In nested loops CONTINUE always refers to the loop that includes it immediately. CONTINUE is typically used in conjunction with an IF statement.

If the loop is to exit regardless of the termination test, use the EXIT statement.

Example: CONTINUE statement

The following example shows the use of the CONTINUE statement to avoid a division-by-0 error when calculating the percentage of a value:

```
FOR i := 0 TO 10 DO
IF value[i] = 0 THEN CONTINUE; END_IF;
     p := part / value[i] * 100;
     s := INT_TO_STRING(p);
     percent := CONCAT(IN1:=s, IN2:="%");
END_FOR;
```

8.8.10.8 EXIT statement

Table 8- 162 EXIT instruction

SCL	Description
`EXIT;`	An EXIT statement is used to exit a loop (FOR, WHILE or REPEAT) at any point, regardless of whether the terminate condition is satisfied.

The EXIT statement executes according to the following rules:

- This statement causes the repetition statement immediately surrounding the exit statement to be exited immediately.
- Execution of the program is continued after the end of the loop (for example after END_FOR).

Use the EXIT statement within a loop. In nested loops, the EXIT statement returns the processing to the next higher nesting level.

Example: EXIT statement

```
FOR i := 0 TO 10 DO
CASE value[i, 0] OF
 1..10: value [i, 1]:="A";
 11..40: value [i, 1]:="B";
 41..100: value [i, 1]:="C";
ELSE
EXIT;
END_CASE;
END_FOR;
```

8.8.10.9 GOTO statement

Table 8- 163 GOTO statement

SCL	Description

`GOTO JumpLabel;`
`Statement;`
` ... ;`
`JumpLabel: Statement;`
	The GOTO statement skips over statements by jumping to a label in the same block.
	The jump label ("JumpLabel") and the GOTO statement must be in the same block. The name of a jump label can only be assigned once within a block. Each jump label can be the target of several GOTO statements.

It is not possible to jump to a loop section (FOR, WHILE or REPEAT). It is possible to jump from within a loop.

Example: GOTO statement

In the following example: Depending on the value of the "Tag_value" operand, the execution of the program resumes at the point defined by the corresponding jump label. If "Tag_value" equals 2, the program execution resumes at the jump label "MyLabel2" and skips "MyLabel1".

```
CASE "Tag_value" OF
1 : GOTO MyLabel1;
2 : GOTO MyLabel2;
ELSE GOTO MyLabel3;
END_CASE;
MyLabel1: "Tag_1" := 1;
MyLabel2: "Tag_2" := 1;
MyLabel3: "Tag_4" := 1;
```

8.8.10.10 RETURN statement

Table 8- 164 RETURN instruction

SCL	Description
`RETURN;`	The Return instruction exits the code block being executed without conditions. Program execution returns to the calling block or to the operating system (when exiting an OB).

Example: RETURN instruction:

```
IF "Error" <> 0 THEN
RETURN;
END_IF;
```

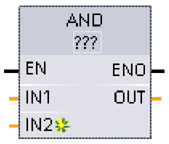
Note

After executing the last instruction, the code block automatically returns to the calling block. Do not insert a RETURN instruction at the end of the code block.

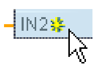
8.9 Word logic operations

8.9.1 AND, OR, and XOR logic operation instructions

Table 8- 165 AND, OR, and XOR logic operation instructions

LAD / FBD	SCL	Description
	`out := in1 AND in2;`	AND: Logical AND
	`out := in1 OR in2;`	OR: Logical OR
	`out := in1 XOR in2;`	XOR: Logical EXCLUSIVE OR

1 For LAD and FBD: Click the "???" and select a data type from the drop-down menu.



To add an input, click the "Create" icon or right-click on an input stub for one of the existing IN parameters and select the "Insert input" command.

To remove an input, right-click on an input stub for one of the existing IN parameters (when there are more than the original two inputs) and select the "Delete" command.

Table 8- 166 Data types for the parameters

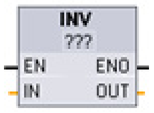
Parameter	Data type	Description
IN1, IN2	Byte, Word, DWord	Logical inputs
OUT	Byte, Word, DWord	Logical output

1 The data type selection sets parameters IN1, IN2, and OUT to the same data type.

The corresponding bit values of IN1 and IN2 are combined to produce a binary logic result at parameter OUT. ENO is always TRUE following the execution of these instructions.

8.9.2 INV (Create ones complement) instruction

Table 8- 167 INV instruction

LAD / FBD	SCL	Description
	`Not available`	Calculates the binary one's complement of the parameter IN. The one's complement is formed by inverting each bit value of the IN parameter (changing each 0 to 1 and each 1 to 0). ENO is always TRUE following the execution of this instruction.

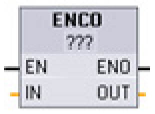
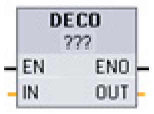
1 For LAD and FBD: Click the "???" and select a data type from the drop-down menu.

Table 8- 168 Data types for the parameters

Parameter	Data type	Description
IN	SInt, Int, DInt, USInt, UInt, UDInt, Byte, Word, DWord	Data element to invert
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Byte, Word, DWord	Inverted output

8.9.3 DECO (Decode) and ENCO (Encode) instructions

Table 8- 169 ENCO and DECO instruction

LAD / FBD	SCL	Description
	`out := ENCO(_in_);`	Encodes a bit pattern to a binary number The ENCO instruction converts parameter IN to the binary number corresponding to the bit position of the least-significant set bit of parameter IN and returns the result to parameter OUT. If parameter IN is either 0000 0001 or 0000 0000, then a value of 0 is returned to parameter OUT. If the parameter IN value is 0000 0000, then ENO is set to FALSE.
	`out := DECO(_in_);`	Decodes a binary number to a bit pattern The DECO instruction decodes a binary number from parameter IN, by setting the corresponding bit position in parameter OUT to a 1 (all other bits are set to 0). ENO is always TRUE following execution of the DECO instruction. Note: The default data type for the DECO instruction is DWORD. In SCL, change the instruction name to DECO_BYTE or DECO_WORD to decode a byte or word value, and assign to a byte or word tag or address.

1 For LAD and FBD: Click the "???" and select a data type from the drop-down menu.

Table 8- 170 Data types for the parameters

Parameter	Data type	Description
IN	ENCO: Byte, Word, DWord DECO: UInt	ENCO: Bit pattern to encode DECO: Value to decode
OUT	ENCO: Int DECO: Byte, Word, DWord	ENCO: Encoded value DECO: Decoded bit pattern

Table 8- 171 ENO status

ENO	Condition	Result (OUT)
1	No error	Valid bit number
0	IN is zero	OUT is set to zero

The DECO parameter OUT data type selection of a Byte, Word, or DWord restricts the useful range of parameter IN. If the value of parameter IN exceeds the useful range, then a modulo operation is performed to extract the least significant bits shown below.

DECO parameter IN range:

- 3 bits (values 0-7) IN are used to set 1 bit position in a Byte OUT
- 4-bits (values 0-15) IN are used to set 1 bit position in a Word OUT
- 5 bits (values 0-31) IN are used to set 1 bit position in a DWord OUT

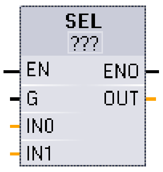
Table 8- 172 Examples

DECO IN value			DECO OUT value (Decode single bit position)
Byte OUT 8 bits	Min. IN	0	00000001
	Max. IN	7	10000000
Word OUT 16 bits	Min. IN	0	0000000000000001
	Max. IN	15	1000000000000000
DWord OUT 32 bits	Min. IN	0	00000000000000000000000000000001
	Max. IN	31	10000000000000000000000000000000

8.9.4 SEL (Select), MUX (Multiplex), and DEMUX (Demultiplex) instructions

Table 8- 173 SEL (select) instruction

LAD / FBD	SCL	Description

	`out := SEL(`
	`    g:=_bool_in,`
	`    in0:-_variant_in,`
	`    in1:=_variant_in);`
		SEL assigns one of two input values to parameter OUT, depending on the parameter G value.

1 For LAD and FBD: Click the "???" and select a data type from the drop-down menu.

Table 8- 174 Data types for the SEL instruction

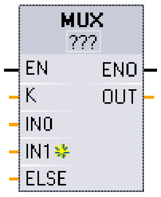
Parameter	Data type 1	Description
G	Bool	• 0 selects IN0 • 1 selects IN1
IN0, IN1	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Time, Date, TOD, Char, WChar	Inputs
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Time, Date, TOD, Char, WChar	Output

1 Input variables and the output variable must be of the same data type.

Condition codes: ENO is always TRUE following execution of the SEL instruction.

Table 8- 175 MUX (multiplex) instruction

LAD / FBD	SCL	Description

	`out := MUX(`
	`k:=_unit_in,`
	`in1:=variant_in,`
	`in2:=variant_in,`
	
	`[...in32:=variant_in,]`
	`inelse:=variant_in);`
		MUX copies one of many input values to parameter OUT, depending on the parameter K value. If the parameter K value exceeds (IN*n* - 1), then the parameter ELSE value is copied to parameter OUT.

1 For LAD and FBD: Click the "???" and select a data type from the drop-down menu.

To add an input, click the "Create" icon or right-click on an input stub for one of the existing IN parameters and select the "Insert input" command.

To remove an input, right-click on an input stub for one of the existing IN parameters (when there are more than the original two inputs) and select the "Delete" command.

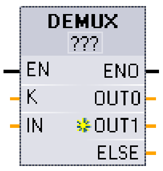
Table 8- 176 Data types for the MUX instruction

Parameter	Data type	Description
K	UInt	• 0 selects IN1 • 1 selects IN2 • *n* selects IN*n*
IN0, IN1, .. INn	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Time, Date, TOD, Char, WChar	Inputs
ELSE	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Time, Date, TOD, Char, WChar	Input substitute value (optional)
OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Time, Date, TOD, Char, WChar	Output

1 Input variables and the output variable must be of the same data type.

Table 8- 177 DEMUX (Demultiplex) instruction

LAD / FBD	SCL	Description

 DEMUX
???
EN
ENO
K
OUT0
IN
OUT1
ELSE
	`DEMUX(`
	`    k:=_unit_in,`
	`    in:=variant_in,`
	`    out1:=variant_in,`
	`    out2:=variant_in,`
	
	`[...out32:=variant_in,]`
	
	`outelse:=variant_in);`
		DEMUX copies the value of the location assigned to parameter IN to one of many outputs. The value of the K parameter selects which output selected as the destination of the IN value. If the value of K is greater than the number (OUT*n* - 1) then the IN value is copied to location assigned to the ELSE parameter.

1 For LAD and FBD: Click the "???" and select a data type from the drop-down menu.

To add an output, click the "Create" icon or right-click on an output stub for one of the existing OUT parameters and select the "Insert output" command.

To remove an output, right-click on an output stub for one of the existing OUT parameters (when there are more than the original two outputs) and select the "Delete" command.

Table 8- 178 Data types for the DEMUX instruction

Parameter	Data type [1]	Description
K	UInt	Selector value: • 0 selects OUT1 • 1 selects OUT2 • n selects OUTn
IN	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Time, Date, TOD, Char, WChar	Input
OUT0, OUT1, .. OUTn	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Time, Date, TOD, Char, WChar	Outputs
ELSE	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal, Byte, Word, DWord, Time, Date, TOD, Char, WChar	Substitute output when K is greater than (OUTn - 1)

1 The input variable and the output variables must be of the same data type.

Table 8- 179 ENO status for the MUX and DEMUX instructions

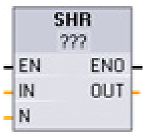
ENO	Condition	Result OUT
1	No error	MUX: Selected IN value is copied to OUT DEMUX: IN value is copied to selected OUT
0	MUX: K is greater than the number of inputs -1	• No ELSE provided: OUT is unchanged, • ELSE provided, ELSE value assigned to OUT
	DEMUX: K is greater than the number of outputs -1	• No ELSE provided: outputs are unchanged, • ELSE provided, IN value copied to ELSE

8.10 Shift and rotate

8.10.1 SHR (Shift right) and SHL (Shift left) instructions

Table 8- 180 SHR and SHL instructions

LAD / FBD	SCL	Description

	`out := SHR(`
	`    in:=_variant_in_,`
	`    n:=_uint_in);`
	`out := SHL(`
	`    in:=_variant_in_,`
	`    n:=_uint_in);`
		Use the shift instructions (SHL and SHR) to shift the bit pattern of parameter IN. The result is assigned to parameter OUT. Parameter N specifies the number of bit positions shifted:
		• SHR: Shift bit pattern right
		• SHL: Shift bit pattern left

1 For LAD and FBD: Click the "???" and select the data types from the drop-down menu.

Table 8- 181 Data types for the parameters

Parameter	Data type	Description
IN	Integers	Bit pattern to shift
N	USInt, UDint	Number of bit positions to shift
OUT	Integers	Bit pattern after shift operation

- For N=0, no shift occurs. The IN value is assigned to OUT.
- Zeros are shifted into the bit positions emptied by the shift operation.
- If the number of positions to shift (N) exceeds the number of bits in the target value (8 for Byte, 16 for Word, 32 for DWord), then all original bit values will be shifted out and replaced with zeros (zero is assigned to OUT).
- ENO is always TRUE for the shift operations.

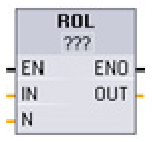
Table 8- 182 Example: SHL for Word data

Shift the bits of a Word to the left by inserting zeroes from the right (N = 1)			
IN	1110 0010 1010 1101	OUT value before first shift:	1110 0010 1010 1101
		After first shift left:	1100 0101 0101 1010
		After second shift left:	1000 1010 1011 0100
		After third shift left:	0001 0101 0110 1000

8.10.2 ROR (Rotate right) and ROL (Rotate left) instructions

Table 8- 183 ROR and ROL instructions

LAD / FBD	SCL	Description

	`out := ROL(`
	`    in:=_variant_in_,`
	`    n:=_uint_in);`
	`out := ROR(`
	`    in:=_variant_in_,`
	`    n:=_uint_in);`
		Use the rotate instructions (ROR and ROL) to rotate the bit pattern of parameter IN. The result is assigned to parameter OUT. Parameter N defines the number of bit positions rotated.
		• ROR: Rotate bit pattern right
		• ROL: Rotate bit pattern left

1 For LAD and FBD: Click the "???" and select the data types from the drop-down menu.

Table 8- 184 Data types for the parameters

Parameter	Data type	Description
IN	Integers	Bit pattern to rotate
N	USInt, UDint	Number of bit positions to rotate
OUT	Integers	Bit pattern after rotate operation

- For N=0, no rotate occurs. The IN value is assigned to OUT.
- Bit data rotated out one side of the target value is rotated into the other side of the target value, so no original bit values are lost.
- If the number of bit positions to rotate (N) exceeds the number of bits in the target value (8 for Byte, 16 for Word, 32 for DWord), then the rotation is still performed.
- ENO is always TRUE following execution of the rotate instructions.

Table 8- 185 Example: ROR for Word data

Rotate bits out the right -side into the left -side (N = 1)			
IN	0100 0000 0000 0001	OUT value before first rotate:	0100 0000 0000 0001
		After first rotate right:	1010 0000 0000 0000
		After second rotate right:	0101 0000 0000 0000

Extended instructions 9

9.1 Date, time-of-day, and clock functions

9.1.1 Date and time-of-day instructions

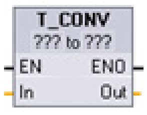
Use the date and time instructions for calendar and time calculations.

- T_CONV converts a value to or from (date and time data types) and (byte, word, and dword size data types)
- T_ADD adds Time and DTL values: (Time + Time = Time) or (DTL + Time = DTL)
- T_SUB subtracts Time and DTL values: (Time - Time = Time) or (DTL - Time = DTL)
- T_DIFF provides the difference between two DTL values as a Time value: DTL - DTL = Time
- T_COMBINE combines a Date value and a Time_and_Date value to create a DTL value

For information about the format of DTL and Time data, refer to the section on the Time and Date data types (Page 120).

Table 9- 1 T_CONV (Convert times and extract) instruction

LAD / FBD	SCL example	Description

	`out := DINT_TO_TIME( in:=_variant_in);`
	
	`out := TIME_TO_DINT( in:=_variant_in);`
		T_CONV converts a value to or from (date and time data types) and (byte, word, and dword size data types).

1 For LAD and FBD boxes: Click "???" and select the source/target data types from the drop-down menu.

2 For SCL: Drag T_CONV from instruction tree and drop into the program editor, then select the source/target data types.

Table 9- 2 Valid data types for T_CONV conversions

Data type IN (or OUT)	Data types OUT (or IN)
TIME (milliseconds)	DInt, Int, SInt, UDInt, UInt, USInt, TOD SCL only: Byte, Word, Dword
DATE (number of days since Jan. 1 1990)	DInt, Int, SInt, UDInt, UInt, USInt, DTL SCL only: Byte, Word, Dword
TOD (milliseconds since midnight- 24:00:00.000)	DInt, Int, SInt, UDInt, UInt, USInt, TIME, DTL SCL only: Byte, Word, Dword

Note

Using T_CONV to convert a larger data size to a smaller data size

Data values can be truncated when you convert a larger data type with more bytes to a smaller data type with less bytes. If this error occurs, then ENO is set to 0.

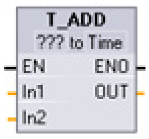
Conversion to/from DTL data type

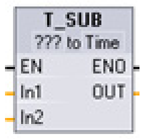
DTL (Date and Time Long) contains year, month, date, and time data. DTL data can be converted to/from DATE and TOD data types.
However, DTL conversion with DATE data only affects the year, month, and day values. DTL conversion with TOD data only affects the hour, minutes, and seconds values.

When T_CONV converts to DTL, the unaffected data elements in the DTL format are left unchanged.

Table 9- 3 T_ADD (Add times) and T_SUB (Subtract times) instructions

LAD / FBD	SCL	Description

	`out := T_ADD(`
	`    in1:=_variant_in,`
	`    in2:=_time_in);`
		T_ADD adds the input IN1 value (DTL or Time data types) with the input IN2 Time value. Parameter OUT provides the DTL or Time value result. Two data type operations are possible:
		• Time + Time = Time
		• DTL + Time = DTL

	`out := T_SUB(`
	`    in1:=_variant_in,`
	`    in2:=_time_in);`
		T_SUB subtracts the IN2 Time value from IN1 (DTL or Time value). Parameter OUT provides the difference value as a DTL or Time data type. Two data type operations are possible:
		• Time - Time = Time
		• DTL - Time = DTL

1 For LAD and FBD: Click the "???" and select the data types from the drop-down menu.

Table 9- 4 Data types for the T_ADD and T_SUB parameters

Parameter and type		Data type	Description
IN1[1]	IN	DTL, Time	DTL or Time value
IN2	IN	Time	Time value to add or subtract
OUT	OUT	DTL, Time	DTL or Time sum or difference

1 Select the IN1 data type from the drop-down list available below the instruction name. The IN1 data type selection also sets the data type of parameter OUT.

Table 9- 5 T_DIFF (Time difference) instruction

LAD / FBD	SCL	Description

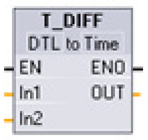
	`out := T_DIFF(`
	`    in1:=_DTL_in,`
	`    in2:=_DTL_in);`
		T_DIFF subtracts the DTL value (IN2) from the DTL value (IN1). Parameter OUT provides the difference value as a Time data type.
		• DTL - DTL = Time

Table 9- 6 Data types for the T_DIFF parameters

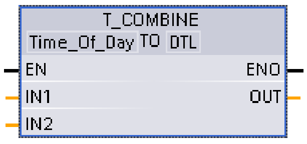
Parameter and type		Data type	Description
IN1	IN	DTL	DTL value
IN2	IN	DTL	DTL value to subtract
OUT	OUT	Time	Time difference

Condition codes: ENO = 1 means no error occurred. ENO = 0 and parameter OUT = 0 errors:

- Invalid DTL value
- Invalid Time value

Table 9- 7 T_COMBINE (Combine times) instruction

LAD / FBD	SCL	Description

 T_COMBINE
Time_Of_Day TO DTL
EN
ENO
IN1
OUT
IN2
	`out :=`
	`CONCAT_DATE_TOD(`
	`    In1 := _date_in,`
	`    In2 := _tod_in);`
		T_COMBINE combines a Date value and a Time_of_Day value to create a DTL value.

[1] Note that the T_COMBINE instruction in the Extended Instructions equates to the CONCAT_DATE_TOD function in SCL.

Table 9- 8 Data types for the T_COMBINE parameters

Parameter and type		Data type	Description
IN1	IN	Date	Date value to be combined must be between DATE#1990-01-01 and DATE#2089-12-31
IN2	IN	Time_of_Day	Time_of_Day values to be combined
OUT	OUT	DTL	DTL value

9.1.2 Clock functions

WARNING
If an attacker can access your networks through Network Time Protocol (NTP) synchronization, the attacker can possibly take limited control of your process by shifting the CPU system time. The NTP client feature of the S7-1200 CPU is disabled by default, and, when enabled, only allows configured IP addresses to act as an NTP server. The CPU disables this feature by default, and you must configure this feature to allow remotely-controlled CPU system time corrections. The S7-1200 CPU supports "time of day" interrupts and clock instructions that depend upon accurate CPU system time. If you configure NTP and accept time synchronization from a server, you must ensure that the server is a trusted source. Failure to do so can cause a security breach that allows an unknown user to take limited control of your process by shifting the CPU system time. For security information and recommendations, please see our "Operational Guidelines for Industrial Security" (http://www.industry.siemens.com/topics/global/en/industrial-security/Documents/operational_guidelines_industrial_security_en.pdf) on the Siemens Service and Support site.

Use the clock instructions to set and read the CPU system clock. The data type DTL (Page 120) is used to provide date and time values.

Table 9- 9 System time instructions

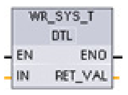
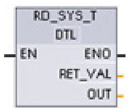
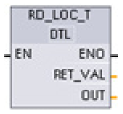
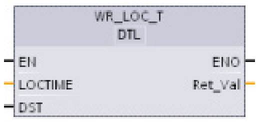
LAD / FBD	SCL	Description
	`ret_val := WR_SYS_T( in:=_DTL_in_);`	WR_SYS_T (Set time-of-day) sets the CPU time of day clock with a DTL value at parameter IN. This time value does not include local time zone or daylight saving time offsets.
	`ret_val := RD_SYS_T( out=>_DTL_out);`	RD_SYS_T (Read time-of-day) reads the current system time from the CPU. This time value does not include local time zone or daylight saving time offsets.
	`ret_val := RD_LOC_T( out=>_DTL_out);`	RD_LOC_T (Read local time) provides the current local time of the CPU as a DTL data type. This time value reflects the local time zone adjusted appropriately for daylight saving time (if configured).
	`ret_val := WR_LOC_T( LOCTIME:=DTL_in_, DST:_in_;`	WR_LOC_T (Write local time) sets the date and time of the CPU clock. You assign the date and time information as local time at LOCTIME with DTL data type. The instruction uses the "TimeTransformationRule (Page 322)" DB structure to calculate the system time. The granularity of the time information for local time and system time is product-specific and is at least one millisecond. Input values at the LOCTIME parameter which are less than those supported by the CPU are rounded up during system time calculation. **Note**: You must use the CPU device configuration to set the "Time of day" properties (time zone, DST activation, DST start, and DST stop). Otherwise, WR_LOC_T cannot interpret the DST time change.

Table 9- 10 Data types for the parameters

Parameter and type		Data type	Description
IN	IN	DTL	Time of day to set in the CPU system clock
OUT	OUT	DTL	RD_SYS_T: Current CPU system time RD_LOC_T: Current local time, including any adjustment for daylight saving time, if configured
LOCTIME	IN	DTL	WR_LOC_T: Local time
DST	IN	BOOL	WR_LOC_T: **D**aylight **S**aving **T**ime only evaluated during the "double hour" when the clocks change to daylight saving time. • TRUE = daylight saving time (first hour) • FALSE = standard time (second hour)
RET_VAL	OUT	Int	Execution condition code

- The local time is calculated by using the time zone and daylight saving time offsets that you set in the device configuration general tab "Time of day" parameters.
- Time zone configuration is an offset to UTC or GMT time.
- Daylight saving time configuration specifies the month, week, day, and hour when daylight saving time begins.
- Standard time configuration also specifies the month, week, day, and hour when standard time begins.
- The time zone offset is always applied to the system time value. The daylight saving time offset is only applied when daylight saving time is in effect.

Note

Daylight saving and standard start time configuration

The "Time of day" properties for "Start for daylight saving time" of the CPU device configuration must be your local time.

Condition codes: ENO = 1 means no error occurred. ENO = 0 means an execution error occurred, and a condition code is provided at the RET_VAL output.

RET_VAL (W#16#....)	Description
0000	The current local time is in standard time.
0001	Daylight saving time has been configured, and the current local time is in daylight saving time.
8080	Local time not available or LOCTIME value is invalid.
8081	Illegal year value or time value assigned by the LOCTIME parameter is invalid
8082	Illegal month value (byte 2 in DTL format)
8083	Illegal day value (byte 3 in DTL format)
8084	Illegal hour value (byte 5 in DTL format)
8085	Illegal minute value (byte 6 in DTL format)
8086	Illegal second value (byte 7 in DTL format)
8087	Illegal nanosecond value (bytes 8 to 11 in DTL format)
8089	Time value does not exist (hour already passed upon changeover to daylight saving time)
80B0	The real-time clock has failed
80B1	The "TimeTransformationRule" structure has not been defined.

9.1.3 TimeTransformationRule data structure

Description

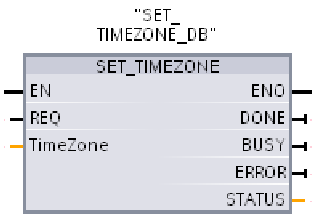
The changeover rules for standard and daylight saving time are defined in the TimeTransformationRule structure. The structure is as follows:

Name		Data type	Description
TimeTransformationRule		STRUCT	
	Bias	INT	Time difference between local time and UTC [minutes] Range: -1439 to 1439
	DaylightBias	INT	Time difference between daylight saving and standard time [minutes] Range: 0 to 60
	DaylightStartMonth	USINT	Month of conversion to daylight saving time Range: 1 to 12
	DaylightStartWeek	USINT	Week of conversion to daylight saving time 1 = First occurrence of the weekday in the month, ..., 5 = Last occurrence of the weekday in the month
	DaylightStartWeekday	USINT	Weekday of daylight saving time changeover: 1 = Sunday
	DaylightStartHour	USINT	Hour of daylight saving time changeover: Range: 0 to 23
	DaylightStartMinute	USINT	Minute of daylight saving time changeover Range: 0 to 59
	StandardStartMonth	USINT	Month of conversion to standard time Range: 1 to 12
	StandardStartWeek	USINT	Week of conversion to standard time 1 = First occurrence of the weekday in the month, ..., 5 = Last occurrence of the weekday in the month
	StandardStartWeekday	USINT	Weekday of standard time changeover: 1 = Sunday
	StandardStartHour	USINT	Hour of standard time changeover Range: 0 to 23
	StandardStartMinute	USINT	Minute of standard time changeover Range: 0 to 59
	TimeZoneName	STRING[80]	Name of time zone: "(GMT+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna"

9.1.4 SET_TIMEZONE (Set timezone) instruction

Table 9- 11 SET_TIMEZONE instruction

LAD / FBD	SCL	Description

 "SET_ TIMEZONE_DB"
SET_TIMEZONE
EN
ENO
REQ
DONE
TimeZone
BUSY
ERROR
STATUS
	`"SET_TIMEZONE_DB"(`
	`REQ:=_bool_in,`
	`Timezone:=_struct_in,`
	`DONE=>_bool_out_,`
	`BUSY=>_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_);`
		Sets the local time zone and daylight saving parameters that are used to transform the CPU system time to local time.

[1] In the SCL example, "SET_TIMEZONE_DB" is the name of the instance DB.

Table 9- 12 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	REQ=1: execute function
Timezone	IN	TimeTransformationRule	Rules for the transformation from system time to local time
DONE	OUT	Bool	Function complete
BUSY	OUT	Bool	Function busy
ERROR	OUT	Bool	Error detected
STATUS	OUT	Word	Function result / error message

To manually configure the time zone parameters for the CPU, use the "Time of day" properties of the "General" tab of the device configuration.

Use the SET_TIMEZONE instruction to set the local time configuration programmatically. The parameters of the "TimeTransformationRule (Page 322)" structure assign the local time zone and timing for automatic switching between standard time and daylight saving time.

Condition codes: ENO = 1 means no error occurred. ENO = 0 means an execution error occurred, and a condition code is provided at the STATUS output.

STATUS (W#16#....)	Description
0	No error
7000	No job processing active
7001	Start of job processing. Parameter BUSY = 1, DONE = 0
7002	Intermediate call (REQ irrelevant): Instruction already active; BUSY has the value "1".
808x	Error at x-th component: For example 8084 indicates that DaylightStartWeekif is not a value from 1to 5.

9.1.5 RTM (Runtime meters) instruction

Table 9- 13 RTM instruction

LAD / FBD	SCL	Description

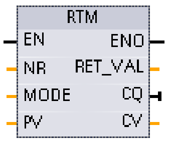
	`RTM(NR:=_uint_in_,`
	`MODE:=_byte_in_,`
	`PV:=_dint_in_,`
	`CQ=>_bool_out_,`
	`CV=>_dint_out_);`
		The RTM (Runtime Meters) instruction can set, start, stop, and read the runtime hour meters in the CPU.

Table 9- 14 Data types for the parameters

Parameter and type		Data type	Description
NR	IN	UInt	Runtime meter number: (possible values: 0..9)
MODE	IN	Byte	RTM Execution mode number: • 0 = Fetch values (the status is then written to CQ and the current value to CV) • 1 = Start (at the last counter value) • 2 = Stop • 4 = Set (to the value specified in PV) • 5 = Set (to the value specified in PV) and then start • 6 = Set (to the value specified in PV) and then stop • 7 = Save all RTM values in the CPU to the MC (Memory Card)
PV	IN	DInt	Preset hours value for the specified runtime meter
RET_VAL	OUT	Int	Function result / error message
CQ	OUT	Bool	Runtime meter status (1 = running)
CV	OUT	DInt	Current runtime hours value for the specified meter

The CPU operates up to 10 runtime hour meters to track the runtime hours of critical control subsystems. You must start the individual hour meters with one RTM execution for each timer. All runtime hour meters are stopped when the CPU makes a run-to-stop transition. You can also stop individual timers with RTM execution mode 2.

When a CPU makes a stop-to-run transition, you must restart the hour timers with one RTM execution for each timer that is started. After a runtime meter value is greater than 2147483647 hours, counting stops and the "Overflow" error is sent. You must execute the RTM instruction once for each timer to reset or modify the timer.

A CPU power failure or power cycle causes a power-down process that saves the current runtime meter values in retentive memory. Upon CPU power-up, the stored runtime meter values are reloaded to the timers and the previous runtime hour totals are not lost. The runtime meters must be restarted to accumulate additional runtime.

Your program can also use RTM execution mode 7 to save the runtime meter values in a memory card. The states of all timers at the instant RTM mode 7 is executed are stored in the memory card. These stored values can become incorrect over time as the hour timers are started and stopped during a program run session. You must periodically update the memory card values to capture important runtime events. The advantage that you get from storing the RTM values in the memory card is that you can insert the memory card in a substitute CPU where your program and saved RTM values will be available. If you did not save the RTM values in the memory card, then the timer values would be lost (in a substitute CPU).

Note

Avoid excessive program calls for memory card write operations

Minimize flash memory card write operations to extend the life of the memory card.

Table 9- 15 Condition codes

RET_VAL (W#16#....)	Description
0	No error
8080	Incorrect runtime meter number
8081	A negative value was passed to the parameter PV
8082	Overflow of the operating hours counter
8091	The input parameter MODE contains an illegal value
80B1	Value cannot be saved to MC (MODE=7)

9.2 String and character

9.2.1 String data overview

String data type

String data is stored as a 2-byte header followed by up to 254 character bytes of ASCII character codes. A String header contains two lengths. The first byte is the maximum length that is given in square brackets when you initialize a string, or 254 by default. The second header byte is the current length that is the number of valid characters in the string. The current length must be smaller than or equal to the maximum length. The number of stored bytes occupied by the String format is 2 bytes greater than the maximum length.

Initialize your String data

String input and output data must be initialized as valid strings in memory, before execution of any string instructions.

Valid String data

A valid string has a maximum length that must be greater than zero but less than 255. The current length must be less than or equal to the maximum length.

Strings cannot be assigned to I or Q memory areas.

For more information see: Format of the String data type (Page 122).

9.2.2 S_MOVE (Move character string) instruction

Table 9- 16 String move instruction

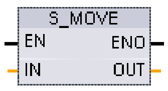
LAD / FBD	SCL	Description
	`out := in;`	Copy the source IN string to the OUT location. S_MOVE execution does not affect the contents of the source string.

Table 9- 17 Data types for the parameters

Parameter	Data type	Description
IN	String	Source string
OUT	String	Target address

If the actual length of the string at the input IN exceeds the maximum length of a string stored at output OUT, then the part of the IN string which can fit in the OUT string is copied.

9.2.3 String conversion instructions

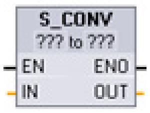
9.2.3.1 S_CONV, STRG_VAL, and VAL_STRG (Convert to/from character string and number) instructions

You can convert number character strings to number values or number values to number character strings with these instructions:

- S_CONV converts (number string to a number value) or (number value to a number string)
- STRG_VAL converts a number string to a number value with format options
- VAL_STRG converts a number value to a number string with format options

S_CONV (convert character string)

Table 9- 18 String conversion instruction

LAD / FBD	SCL	Description
	`out := <Type>_TO_<Type>(in);`	Converts a character string to the corresponding value, or a value to the corresponding character string. The S_CONV instruction has no output formatting options. This makes the S_CONV instruction simpler, but less flexible than the STRG_VAL and VAL_STRG instructions.

1 For LAD / FBD: Click the "???" and select the data type from the drop-down list.

2 For SCL: Select S_CONV from the Extended Instructions, and answer the prompts for the data types for the conversion. STEP 7 then provides the appropriate conversion instruction.

Table 9- 19 Data types (string to value)

Parameter and type		Data type	Description
IN	IN	String, WString	Input character string
OUT	OUT	String, WString, Char, WChar, SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	Output number value

Conversion of the string parameter IN starts at the first character and continues until the end of the string, or until the first character is encountered that is not "0" through "9", "+", "-", or ".". The result value is provided at the location specified in parameter OUT. If the output number value does not fit in the range of the OUT data type, then parameter OUT is set to 0 and ENO is set to FALSE. Otherwise, parameter OUT contains a valid result and ENO is set to TRUE.

Input String format rules:

- If a decimal point is used in the IN string, you must use the "." character.
- Comma characters "," used as a thousands separator to the left of the decimal point are allowed and ignored.
- Leading spaces are ignored.

S_CONV (value to string conversion)

Table 9- 20 Data types (value to string)

Parameter and type		Data type	Description
IN	IN	String, WString, Char, WChar, SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	Input number value
OUT	OUT	String, WString	Output character string

An integer, unsigned integer, or floating point value IN is converted to the corresponding character string at OUT. The parameter OUT must reference a valid string before the conversion is executed. A valid string consists of a maximum string length in the first byte, the current string length in the second byte, and the current string characters in the next bytes. The converted string replaces characters in the OUT string starting at the first character and adjusts the current length byte of the OUT string. The maximum length byte of the OUT string is not changed.

How many characters are replaced depends on the parameter IN data type and number value. The number of characters replaced must fit within the parameter OUT string length. The maximum string length (first byte) of the OUT string should be greater than or equal to the maximum expected number of converted characters. The following table shows S_CONV value to string conversion examples.

Output String format rules:

- Values written to parameter OUT do not use a leading "+" sign.
- Fixed-point representation is used (no exponential notation).
- The period character "." is used to represent the decimal point when parameter IN is the Real data type.
- Values are right-justified in the output string and are preceded by space characters that fill empty character positions.

Table 9- 21 Maximum string lengths for each data type

IN data type	Character positions allocated by S_CONV	Converted string example[1]	Total string length including maximum and current length bytes
USInt	4	"x255"	6
SInt	4	"-128"	6
UInt	6	"x65535"	8
Int	6	"-32768"	8
UDInt	11	"x4294967295"	13
DInt	11	"-2147483648"	13

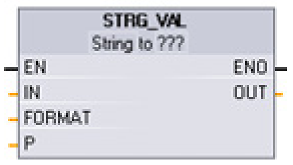
IN data type	Character positions allocated by S_CONV	Converted string example[1]	Total string length including maximum and current length bytes
Real	14	"x-3.402823E+38" "x-1.175495E-38" "x+1.175495E-38" "x+3.402823E+38"	16
LReal	21	"-1.7976931348623E+308" "-2.2250738585072E-308" "+2.2250738585072E-308" "+1.7976931348623E+308"	23

[1] The "x" characters represent space characters that fill empty positions in the right-justified field that is allocated for the converted value.

STRG_VAL (convert characer string to numerical value)

Table 9- 22 String-to-value instruction

LAD / FBD	SCL	Description

 STRG_VAL
String to ???
EN
ENO
IN
OUT
FORMAT
P
	`"STRG_VAL"(`
	`in:=_string_in,`
	`format:=_word_in,`
	`p:=uint_in,`
	`out=>_variant_out);`
		Converts a number character string to the corresponding integer or floating point representation.

[1] For LAD / FBD: Click the "???" and select the data type from the drop-down list.

Table 9- 23 Data types for the STRG_VAL instruction

Parameter and type		Data type	Description
IN	IN	String, WString	The ASCII character string to convert
FORMAT	IN	Word	Output format options
P	IN	UInt, Byte, USInt	IN: Index to the first character to be converted (first character = 1)
OUT	OUT	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	Converted number value

Conversion begins in the string IN at character offset P and continues until the end of the string, or until the first character is encountered that is not "+", "-", ".", ",", "e", "E", or "0" to "9". The result is placed at the location specified in parameter OUT.

String data must be initialized before execution as a valid string in memory.

The FORMAT parameter for the STRG_VAL instruction is defined below. The unused bit positions must be set to zero.

Table 9- 24 Format of the STRG_VAL instruction

Bit 16							Bit 8	Bit 7							Bit 0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	f	r

f = Notation format	1= Exponential notation 0 = Fixed point notation
r = Decimal point format	1 = "," (comma character) 0 = "." (period character)

Table 9- 25 Values of the FORMAT parameter

FORMAT (W#16#)	Notation format	Decimal point representation
0000 (default)	Fixed point	"."
0001		","
0002	Exponential	"."
0003		","
0004 to FFFF	Illegal values	

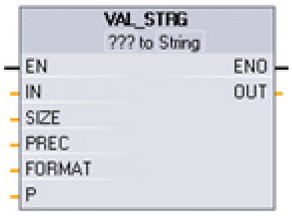
Rules for STRG_VAL conversion:

- If the period character "." is used for the decimal point, then commas "," to the left of the decimal point are interpreted as thousands separator characters. The comma characters are allowed and ignored.
- If the comma character "," is used for the decimal point, then periods "." to the left of the decimal point are interpreted as thousands separator characters. These period characters are allowed and ignored.
- Leading spaces are ignored.

VAL_STRG (convert numerical value to string)

Table 9- 26 Value-to-string operation

LAD / FBD	SCL	Description

 VAL_STRG
??? to String
EN
ENO
IN
OUT
SIZE
PREC
FORMAT
P
	`"VAL_STRG"(`
	`in:=_variant_in,`
	`size:=_usint_in,`
	`prec:=_usint_in,`
	`format:=_word_in,`
	`p:=uint_in,`
	`out=>_string_out);`
		Converts an integer, unsigned integer, or floating point value to the corresponding character string representation.

¹ For LAD / FBD: Click the "???" and select the data type from the drop-down list.

Table 9- 27 Data types for the VAL_STRG instruction

Parameter and type		Data type	Description
IN	IN	SInt, Int, DInt, USInt, UInt, UDInt, Real, LReal	Value to convert
SIZE	IN	USInt	Number of characters to be written to the OUT string
PREC	IN	USInt	The precision or size of the fractional portion. This does not include the decimal point.
FORMAT	IN	Word	Output format options
P	IN	UInt, Byte, USInt	IN: Index to the first OUT string character to be replaced (first character = 1)
OUT	OUT	String, WString	The converted string

This instruction converts the value represented by parameter IN to a string referenced by parameter OUT. The parameter OUT must be a valid string before the conversion is executed.

The converted string replaces characters in the OUT string starting at character offset count P to the number of characters specified by parameter SIZE. The number of characters in SIZE must fit within the OUT string length, counting from character position P. If the SIZE parameter is zero, then the characters overwrite at position P in the OUT string without limitation. This instruction is useful for embedding number characters into a text string. For example, you can put the numbers "120" into the string "Pump pressure = 120 psi".

Parameter PREC specifies the precision or number of digits for the fractional part of the string. If the parameter IN value is an integer, then PREC specifies the location of the decimal point. For example, if the data value is 123 and PREC = 1, then the result is "12.3". The maximum supported precision for the Real data type is 7 digits.

If parameter P is greater than the current size of the OUT string, then spaces are added, up to position P, and the result is appended to the end of the string. The conversion ends if the maximum OUT string length is reached.

The FORMAT parameter for the VAL_STRG instruction is defined below. The unused bit positions must be set to zero.

Table 9- 28 Format of the VAL_STRG instruction

Bit 16							**Bit 8**	**Bit 7**							**Bit 0**
0	0	0	0	0	0	0	0	0	0	0	0	0	s	f	r

s = Number sign character	1= use sign character "+" and "-" 0 = use sign character "-" only
f = Notation format	1= Exponential notation 0 = Fixed point notation
r = Decimal point format	1 = "," (comma character) 0 = "." (period character)

Table 9- 29 Values of the FORMAT parameter

FORMAT (WORD)	Number sign character	Notation format	Decimal point representation
W#16#0000	"-" only	Fixed point	"."
W#16#0001			","
W#16#0002		Exponential	"."
W#16#0003			","
W#16#0004	"+" and "-"	Fixed Point	"."
W#16#0005			","
W#16#0006		Exponential	"."
W#16#0007			","
W#16#0008 to W#16#FFFF	Illegal values		

Parameter OUT string format rules:

- Leading space characters are added to the leftmost part of the string when the converted string is smaller than the specified size.
- When the FORMAT parameter sign bit is FALSE, unsigned and signed integer data type values are written to the output buffer without the leading "+" sign. The "-" sign is used if required.
  <leading spaces><digits without leading zeroes>'.'<PREC digits>
- When the sign bit is TRUE, unsigned and signed integer data type values are written to the output buffer always with a leading sign character.

  <leading spaces><sign><digits without leading zeroes>'.'<PREC digits>
- When the FORMAT is set to exponential notation, Real data type values are written to the output buffer as:

  <leading spaces><sign><digit> '.' <PREC digits>'E' <sign><digits without leading zero>
- When the FORMAT is set to fixed point notation, integer, unsigned integer, and real data type values are written to the output buffer as:

  <leading spaces><sign><digits without leading zeroes>'.'<PREC digits>
- Leading zeros to the left of the decimal point (except the digit adjacent to the decimal point) are suppressed.
- Values to the right of the decimal point are rounded to fit in the number of digits to the right of the decimal point specified by the PREC parameter.
- 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.
- Values are right-justified in the output string.

Conditions reported by ENO

When the conversion operation encounters an error, the instruction returns the following results:

- ENO is set to 0.
- OUT is set to 0, or as shown in the examples for string to value conversion.
- OUT is unchanged, or as shown in the examples when OUT is a string.

Table 9- 30 ENO status

ENO	Description
1	No error
0	Illegal or invalid parameter; for example, an access to a DB that does not exist
0	Illegal string where the maximum length of the string is 0 or 255
0	Illegal string where the current length is greater than the maximum length
0	The converted number value is too large for the specified OUT data type.
0	The OUT parameter maximum string size must be large enough to accept the number of characters specified by parameter SIZE, starting at the character position parameter P.
0	Illegal P value where P=0 or P is greater than the current string length
0	Parameter SIZE must be greater than parameter PREC.

Table 9- 31 Example of S_CONV string to value conversion

IN string	OUT data type	OUT value	ENO
"123"	Int or DInt	123	TRUE
"-00456"	Int or DInt	-456	TRUE
"123.45"	Int or DInt	123	TRUE
"+2345"	Int or DInt	2345	TRUE
"00123AB"	Int or DInt	123	TRUE
"123"	Real	123.0	TRUE
"123.45"	Real	123.45	TRUE
"1.23e-4"	Real	1.23	TRUE
"1.23E-4"	Real	1.23	TRUE
"12,345.67"	Real	12345.67	TRUE
"3.4e39"	Real	3.4	TRUE
"-3.4e39"	Real	-3.4	TRUE
"1.17549e-38"	Real	1.17549	TRUE
"12345"	SInt	0	FALSE
"A123"	N/A	0	FALSE
""	N/A	0	FALSE
"++123"	N/A	0	FALSE
"+-123"	N/A	0	FALSE

Table 9- 32 Examples of S_CONV value to string conversion

Data type	IN value	OUT string [1]	ENO
UInt	123	"xxx123"	TRUE
UInt	0	"xxxxx0"	TRUE
UDInt	12345678	"xxx12345678"	TRUE
Real	+9123.456	"xx+9.123456E+3"	TRUE
LReal	+9123.4567890123	"xx+9.1234567890123 E+3"	TRUE
Real	-INF	"xxxxxxxxxxxINF"	FALSE
Real	+INF	"xxxxxxxxxxxINF"	FALSE
Real	NaN	"xxxxxxxxxxxNaN"	FALSE

[1] The "x" characters represent space characters that fill empty positions in the right-justified field that is allocated for the converted value.

Table 9- 33 Example: STRG_VAL conversion

IN string	FORMAT (W#16#....)	OUT data type	OUT value	ENO
"123"	0000	Int or DInt	123	TRUE
"-00456"	0000	Int or DInt	-456	TRUE
"123.45"	0000	Int or DInt	123	TRUE
"+2345"	0000	Int or DInt	2345	TRUE
"00123AB"	0000	Int or DInt	123	TRUE
"123"	0000	Real	123.0	TRUE
"-00456"	0001	Real	-456.0	TRUE
"+00456"	0001	Real	456.0	TRUE
"123.45"	0000	Real	123.45	TRUE
"123.45"	0001	Real	12345.0	TRUE
"123.45"	0000	Real	12345.0	TRUE
"123.45"	0001	Real	123.45	TRUE
".00123AB"	0001	Real	123.0	TRUE
"1.23e-4"	0000	Real	1.23	TRUE
"1.23E-4"	0000	Real	1.23	TRUE
"1.23E-4"	0002	Real	1.23E-4	TRUE
"12,345.67"	0000	Real	12345.67	TRUE
"12,345.67"	0001	Real	12.345	TRUE
"3.4e39"	0002	Real	+INF	TRUE
"-3.4e39"	0002	Real	-INF	TRUE
"1.1754943e-38" (and smaller)	0002	Real	0.0	TRUE
"12345"	N/A	SInt	0	FALSE
"A123"	N/A	N/A	0	FALSE
""	N/A	N/A	0	FALSE

IN string	FORMAT (W#16#....)	OUT data type	OUT value	ENO
"++123"	N/A	N/A	0	FALSE
"+-123"	N/A	N/A	0	FALSE

The following examples of VAL_STRG conversions are based on an OUT string initialized as follows:

"`Current Temp = xxxxxxxxxx C`"
where the "`x`" character represents space characters allocated for the converted value.

Table 9- 34 Example: VAL_STRG conversion

Data type	IN value	P	SIZE	FORMAT (W#16#....)	PREC	OUT string	ENO
UInt	123	16	10	0000	0	`Current Temp = xxxxxxx123 C`	TRUE
UInt	0	16	10	0000	2	`Current Temp = xxxxxx0.00 C`	TRUE
UDInt	12345678	16	10	0000	3	`Current Temp = x12345.678 C`	TRUE
UDInt	12345678	16	10	0001	3	`Current Temp = x12345,678 C`	TRUE
Int	123	16	10	0004	0	`Current Temp = xxxxxx+123 C`	TRUE
Int	-123	16	10	0004	0	`Current Temp = xxxxxx-123 C`	TRUE
Real	-0.00123	16	10	0004	4	`Current Temp = xxx-0.0012 C`	TRUE
Real	-0.00123	16	10	0006	4	`Current Temp = -1.2300E-3 C`	TRUE
Real	-INF	16	10	N/A	4	`Current Temp = xxxxxx-INF C`	FALSE
Real	+INF	16	10	N/A	4	`Current Temp = xxxxxx+INF C`	FALSE
Real	NaN	16	10	N/A	4	`Current Temp = xxxxxxxNaN C`	FALSE
UDInt	12345678	16	6	N/A	3	`Current Temp = xxxxxxxxxx C`	FALSE

9.2.3.2 Strg_TO_Chars and Chars_TO_Strg (Convert to/from character string and array of CHAR) instructions

Strg_TO_Chars copies an ASCII character string into an array of character bytes.

Chars_TO_Strg copies an array of ASCII character bytes into a character string.

Note

Only the zero based array types (Array [0..n] of Char) or (Array [0..n] of Byte) are allowed as the input parameter Chars for the Chars_TO_Strg instruction, or as the IN_OUT parameter Chars for the Strg_TO_Chars instruction.

Table 9- 35 Strg_TO_Chars instruction

LAD / FBD	SCL	Description

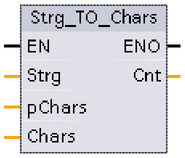
 Strg_TO_Chars
EN
ENO
Strg
Cnt
pChars
Chars
	`Strg_TO_Chars(`
	`    Strg:=_string_in_,`
	`    pChars:=_dint_in_,`
	`    Cnt=>_uint_out_,`
	
	`Chars:=_variant_inout_);`
		The complete input string Strg is copied to an array of characters at IN_OUT parameter Chars.
		The operation overwrites bytes starting at array element number specified by the pChars parameter.
		Strings of all supported max lengths (1..254) may be used.
		An end delimiter is not written; this is your responsibility. To set an end delimiter just after the last written array character, use the next array element number [pChars+Cnt].

Table 9- 36 Data types for the parameters (Strg_TO_Chars)

Parameter and type		Data type	Description
Strg	IN	String, WString	Source string
pChars	IN	DInt	Array element number for the first string character written to the target array
Chars	IN_OUT	Variant	The Chars parameter is a pointer to a zero-based array [0..n] of characters copied from the input string. The array can be declared in a DB or as local variables in the block interface. Example: "DB1".MyArray points to MyArray [0..10] of Char element values in DB1.
Cnt	OUT	UInt	Count of characters copied

Table 9- 37 Chars_TO_Strg instruction

LAD / FBD	SCL	Description

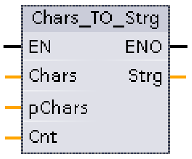
 Chars_TO_Strg
EN
ENO
Chars
Strg
pChars
Cnt
	`Chars_TO_Strg(`
	`Chars:=_variant_in_,`
	`pChars:=_dint_in_,`
	`Cnt:=_uint_in_,`
	`Strg=>_string_out_);`
		All or part of an array of characters is copied to a string.
		The output string must be declared before Chars_TO_Strg is executed. The string is then overwritten by the Chars_TO_Strg operation.
		Strings of all supported maximum lengths (1..254) may be used.
		The string maximum length value is not changed by Chars_TO_Strg operation. Copying from array to string stops when the maximum string length is reached.
		A nul character '$00' or 16#00 value in the character array works as a delimiter and ends copying of characters into the string.

Table 9- 38 Data types for the parameters (Chars_TO_Strg)

Parameter and type		Data type	Description
Chars	IN	Variant	The Chars parameter is a pointer to zero based array [0..n] of characters to be converted into a string. The array can be declared in a DB or as local variables in the block interface. Example: "DB1".MyArray points to MyArray [0..10] of Char element values in DB1.
pChars	IN	Dint	Element number for the first character in the array to copy. Array element [0] is the default value.
Cnt	IN	UInt	Count of characters to copy: 0 means all
Strg	OUT	String, WString	Target string

Table 9- 39 ENO status

ENO	Description
1	No error
0	Chars_TO_Strg: Attempt to copy more character bytes to the output string than allowed by the maximum length byte in the string declaration
0	Chars_TO_Strg: The nul character (16#00) value was found in the input character byte array.
0	Strg_TO_Chars: Attempt to copy more character bytes to the output array than are allowed by the element number limit

9.2.3.3 ATH and HTA (Convert to/from ASCII string and hexadecimal number) instructions

Use the ATH (ASCII to hexadecimal) and HTA (hexadecimal to ASCII) instructions for conversions between ASCII character bytes (characters 0 to 9 and uppercase A to F only) and the corresponding 4-bit hexadecimal nibbles.

Table 9- 40 ATH instruction

LAD / FBD	SCL	Description

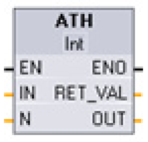
	`ret_val := ATH(`
	`    in:=_variant_in_,`
	`    n:=_int_in_,`
	
	`out=>_variant_out_);`
		Converts ASCII characters into packed hexadecimal digits.

Table 9- 41 Data types for the ATH instruction

Parameter type		Data Type	Description
IN	IN	Variant	Pointer to ASCII character byte array
N	IN	UInt	Number of ASCII character bytes to convert
RET_VAL	OUT	Word	Execution condition code
OUT	OUT	Variant	Pointer to the converted hexadecimal byte array

Conversion begins at the location specified by parameter IN and continues for N bytes. The result is placed at the location specified by OUT. Only valid ASCII characters 0 to 9, lower case a to f, and uppercase A to F can be converted. Any other character will be converted to zero.

8-bit ASCII coded characters are converted to 4-bit hexadecimal nibbles. Two ASCII characters can converted into a single byte containing two 4-bit hexadecimal nibbles.

The IN and OUT parameters specify byte arrays and not hexadecimal String data. ASCII characters are converted and placed in the hexadecimal output in the same order as they are read. If there are an odd number of ASCII characters, then zeros are put in the right-most nibble of the last converted hexadecimal digit.

Table 9- 42 Examples: ASCII-to-hexadecimal (ATH) conversion

IN character bytes	N	OUT value	ENO
'0a23'	4	W#16#0A23	TRUE
'123AFx1a23'	10	16#123AF01023	FALSE
'a23'	3	W#16#A230	TRUE

Table 9- 43 HTA instruction

LAD / FBD	SCL	Description

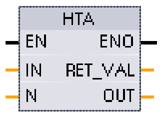
	`ret_val := HTA(`
	`in:=_variant_in_,`
	`n:=_uint_in_,`
	`out=>_variant_out_);`
		Converts packed hexadecimal digits to their corresponding ASCII character bytes.

Table 9- 44 Data types for the HTA instruction

Parameter and type		Data Type	Description
IN	IN	Variant	Pointer to input byte array
N	IN	UInt	Number of bytes to convert (each input byte has two 4-bit nibbles and produces 2N ASCII characters)
RET_VAL	OUT	Word	Execution condition code
OUT	OUT	Variant	Pointer to ASCII character byte array

Conversion begins at the location specified by parameter IN and continues for N bytes. Each 4-bit nibble converts to a single 8-bit ASCII character and produces 2N ASCII character bytes of output. All 2N bytes of the output are written as ASCII characters 0 to 9 through uppercase A to F. The parameter OUT specifies a byte array and not a string.

Each nibble of the hexadecimal byte is converted into a character in the same order as they are read in (left-most nibble of a hexadecimal digit is converted first, followed by the right-most nibble of that same byte).

Table 9- 45 Examples: Hexadecimal -to- ASCII (HTA) conversion

IN value	N	OUT character bytes	ENO (ENO always TRUE after HTA execution)
W#16#0123	2	'0123'	TRUE
DW#16#123AF012	4	'123AF012'	TRUE

Table 9- 46 ATH and HTA condition codes

RET_VAL (W#16#....)	Description	ENO
0000	No error	TRUE
0007	Invalid ATH input character: A character was found that was not an ASCII character 0-9, lowercase a to f, or uppercase A to F	FALSE
8101	Illegal or invalid input pointer, for example, an access to a DB that does not exist.	FALSE
8120	Input string is an invalid format, i.e., max= 0, max=255, current>max, or grant length in pointer < max	FALSE
8182	Input buffer is too small for N	FALSE
8151	Data type not allowed for input buffer	FALSE
8301	Illegal or invalid output pointer, for example, an access to a DB that does not exist.	FALSE
8320	Output string is an invalid format, i.e., max= 0, max=255, current>max, or grant length in pointer < max	FALSE
8382	Output buffer is too small for N	FALSE
8351	Data type not allowed for output buffer	FALSE

9.2.4 String operation instructions

Your control program can use the following string and character instructions to create messages for operator display and process logs.

9.2.4.1 MAX_LEN (Maximum length of a character string) instruction

Table 9- 47 Maximum length instruction

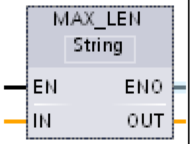
LAD / FBD	SCL	Description
	`out := MAX_LEN(in);`	MAX_LEN (Maximum length of string) provides the maximum length value assigned to string IN at output OUT. If errors occur during processing of the instruction, then an empty string length will be output. The String and WString data types contain two lengths: the first byte (or word) gives the maximum length and the second byte (or word) gives the current length (this is the current number of valid characters). • The maximum length of the character string is assigned for each String or WString declaration in square brackets. The number of bytes occupied by a String is 2 bytes greater than the maximum length. The number of words occupied by a WString is 2 words greater than the maximum length. • The current length represents the number of the characters actually used. The current length must be less than or equal to the maximum length. The current length is in bytes for a String and in words for a WString. Use the MAX_LEN instruction to get the maximum length of the character string and the LEN instruction to get the current length of a string.

Table 9- 48 Data types for the parameters

Parameter and type		Data type	Description
IN	IN	String, WString	Input string
OUT	OUT	DInt	Maximum number of characters allowed for IN string

9.2.4.2 LEN (Determine the length of a character string) instruction

Table 9- 49 Length instruction

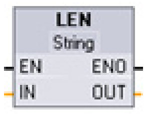
LAD / FBD	SCL	Description
	`out := LEN(in);`	LEN (length) provides the current length of the string IN at output OUT. An empty string has a length of zero.

Table 9- 50 Data types for the parameters

Parameter and type		Data type	Description
IN	IN	String, WString	Input string
OUT	OUT	Int, DInt, Real, LReal	Number of valid characters of IN string

Table 9- 51 ENO status

ENO	Condition	OUT
1	No invalid string condition	Valid string length
0	Current length of IN exceeds maximum length of IN	Current length is set to 0
	Maximum length of IN does not fit within allocated memory range	
	Maximum length of IN is 255 (illegal length)	

9.2.4.3 CONCAT (Combine character strings) instruction

Table 9- 52 Concatenate strings instruction

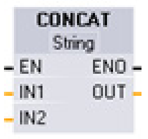
LAD / FBD	SCL	Description
	`out := CONCAT(in1, in2);`	CONCAT (concatenate strings) joins string parameters IN1 and IN2 to form one string provided at OUT. After concatenation, String IN1 is the left part and String IN2 is the right part of the combined string.

Table 9- 53 Data types for the parameters

Parameter and type		Data type	Description
IN1	IN	String, WString	Input string 1
IN2	IN	String, WString	Input string 2
OUT	OUT	String, WString	Combined string (string 1 + string 2)

Table 9- 54 ENO status

ENO	Condition	OUT
1	No errors detected	Valid characters
0	Resulting string after concatenation is larger than maximum length of OUT string	Resulting string characters are copied until the maximum length of the OUT is reached
	Current length of IN1 exceeds maximum length of IN1, current length of IN2 exceeds maximum length of IN2, or current length of OUT exceeds maximum length of OUT (invalid string)	Current length is set to 0
	Maximum length of IN1, IN2 or OUT does not fit within allocated memory range	
	Maximum length of IN1 or IN2 is 255, or the maximum length of OUT is 0 or 255 (String data type)	
	Maximum length of IN1 or IN2 is 65534, or the maximum length of OUT is 0 or 65534 (WString data type)	

9.2.4.4 LEFT, RIGHT, and MID (Read substrings in a character string) instructions

Table 9- 55 Left, right and middle substring operations

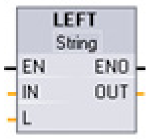
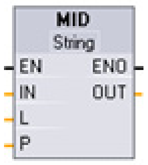
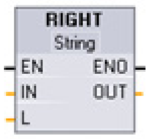
LAD / FBD	SCL	Description
	`out := LEFT(in, L);`	LEFT (Left substring) provides a substring made of the first L characters of string parameter IN. • If L is greater than the current length of the IN string, then the entire IN string is returned in OUT. • If an empty string is the input, then an empty string is returned in OUT.
	`out := MID(in, L, p);`	MID (Middle substring) provides the middle part of a string. The middle substring is L characters long and starts at character position P (inclusive). If the sum of L and P exceeds the current length of the string parameter IN, then a substring is returned that starts at character position P and continues to the end of the IN string.
	`out := RIGHT(in, L);`	RIGHT (Right substring) provides the last L characters of a string. • If L is greater than the current length of the IN string, then the entire IN string is returned in parameter OUT. • If an empty string is the input, then an empty string is returned in OUT.

Table 9- 56 Data types for the parameters

Parameter and type		Data type	Description
IN	IN	String, WString	Input string
L	IN	Int	Length of the substring to be created: • LEFT uses the left-most characters number of characters in the string • RIGHT uses the right-most number of characters in the string • MID uses the number of characters starting at position P within the string
P	IN	Int	MID only: Position of first substring character to be copied P= 1, for the initial character position of the IN string
OUT	OUT	String, WString	Output string

Table 9- 57 ENO status

ENO	Condition	OUT
1	No errors detected	Valid characters
0	• L or P is less than or equal to 0 • P is greater than maximum length of IN • Current length of IN exceeds maximum length of IN, or current length of OUT exceeds maximum length of OUT • Maximum length of IN or OUT does not fit within allocated memory • Maximum length of IN or OUT is 0 or 255 (String data type) or 0 or 65534 (WString data type)	Current length is set to 0
	Substring length (L) to be copied is larger than maximum length of OUT string.	Characters are copied until the maximum length of OUT is reached
	MID only: L or P is less than or equal to 0	Current length is set to 0
	MID only: P is greater than maximum length of IN	
	Current length of IN1 exceeds maximum length of IN1, or current length of IN2 exceeds maximum length of IN2 (invalid string)	Current length is set to 0
	Maximum length of IN1, IN2 or OUT does not fit within allocated memory range	
	Maximum length of IN1, IN2 or OUT is illegal length: 0 or 255 (String data type) or 0 or 65534 (WString data type)	

9.2.4.5 DELETE (Delete characters in a character string) instruction

Table 9- 58 Delete substring instruction

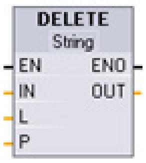
LAD / FBD	SCL	Description
	`out := DELETE(in, L, p);`	Deletes L characters from string IN. Character deletion starts at character position P (inclusive), and the remaining substring is provided at parameter OUT. • If L is equal to zero, then the input string is returned in OUT. • If the sum of L and P is greater than the length of the input string, then the string is deleted to the end.

Table 9- 59 Data types for the parameters

Parameter and type		Data type	Description
IN	IN	String, WString	Input string
L	IN	Int	Number of characters to be deleted
P	IN	Int	Position of the first character to be deleted: The first character of the IN string is position number 1
OUT	OUT	String, WString	Output string

Table 9- 60 ENO status

ENO	Condition	OUT
1	No errors detected	Valid characters
0	P is greater than current length of IN	IN is copied to OUT with no characters deleted
	Resulting string after characters are deleted is larger than maximum length of OUT string	Resulting string characters are copied until the maximum length of OUT is reached
	L is less than 0, or P is less than or equal to 0	Current length is set to 0
	Current length of IN exceeds maximum length of IN, or current length of OUT exceeds maximum length of OUT	
	Maximum length of IN or OUT does not fit within allocated memory	
	Maximum length of IN or OUT is 0 or 255	

9.2.4.6 INSERT (Insert characters in a character string) instruction

Table 9- 61 Insert substring instruction

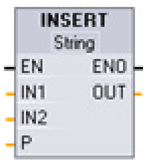
LAD / FBD	SCL	Description
	`out := INSERT(in1, in2, p);`	Inserts string IN2 into string IN1. Insertion begins after the character at position P.

Table 9- 62 Data types for the parameters

Parameter and type		Data type	Description
IN1	IN	String, WString	Input string 1
IN2	IN	String, WString	Input string 2
P	IN	Int	Last character position in string IN1 before the insertion point for string IN2 The first character of string IN1 is position number 1.
OUT	OUT	String, WString	Result string

Table 9- 63 ENO status

ENO	Condition	OUT
1	No errors detected	Valid characters
0	P is greater than length of IN1	IN2 is concatenated with IN1 immediately following the last IN1 character
	P is less than 0	Current length is set to 0
	Resulting string after insertion is larger than maximum length of OUT string	Resulting string characters are copied until the maximum length of OUT is reached
	Current length of IN1 exceeds maximum length of IN1, current length of IN2 exceeds maximum length of IN2, or current length of OUT exceeds maximum length of OUT (invalid string)	Current length is set to 0
	Maximum length of IN1, IN2 or OUT does not fit within allocated memory range	
	Maximum length of IN1 or IN2 is 255, or maximum length of OUT is 0 or 255 (String data type)	
	Maximum length of IN1 or IN2 is 65534, or maximum length of OUT is 0 or 65534 (WString data type)	

9.2.4.7 REPLACE (Replace characters in a character string) instruction

Table 9- 64 Replace substring instruction

LAD / FBD	SCL	Description

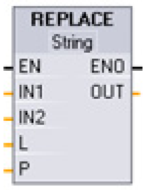
 REPLACE
String
EN
ENO
IN1
OUT
IN2
L
P
	`out := REPLACE(`
	`    in1:=_string_in_,`
	`    in2:=_string_in_,`
	`    L:=_int_in_,`
	`    p:=_int_in);`
		Replaces L characters in the string parameter IN1. Replacement starts at string IN1 character position P (inclusive), with replacement characters coming from the string parameter IN2.

Table 9- 65 Data types for the parameters

Parameter and type		Data type	Description
IN1	IN	String, WString	Input string
IN2	IN	String, WString	String of replacement characters
L	IN	Int	Number of characters to replace
P	IN	Int	Position of first character to be replaced
OUT	OUT	String, WString	Result string

If parameter L is equal to zero, then the string IN2 is inserted at position P of string IN1 without deleting any characters from string IN1.

If P is equal to one, then the first L characters of string IN1 are replaced with string IN2 characters.

Table 9- 66 ENO status

ENO	Condition	OUT
1	No errors detected	Valid characters
0	P is greater than length of IN1	IN2 is concatenated with IN1 immediately following the last IN1 character
	P points within IN1, but fewer than L characters remain in IN1	IN2 replaces the end characters of IN1 beginning at position P
	Resulting string after replacement is larger than maximum length of OUT string	Resulting string characters are copied until the maximum length of OUT is reached
	Maximum length of IN1 is 0	IN2 characters are copied to OUT
	L is less than 0, or P is less than or equal to 0	Current length is set to 0
	Current length of IN1 exceeds maximum length of IN1, current length of IN2 exceeds maximum length of IN2, or current length of OUT exceeds maximum length of OUT	
	Maximum length of IN1, IN2 or OUT does not fit within allocated memory range	
	Maximum length of IN1 or IN2 is 255, or maximum length of OUT is 0 or 255 (String data type)	

ENO	Condition	OUT
	Maximum length of IN1 or IN2 is 65534, or maximum length of OUT is 0 or 65534 (WString data type)	

9.2.4.8 FIND (Find characters in a character string) instruction

Table 9- 67 Find substring instruction

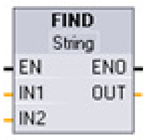
LAD / FBD	SCL	Description
	`out := FIND( in1:=_string_in_, in2:=_string_in);`	Provides the character position of the substring specified by IN2 within the string IN1. The search starts on the left. The character position of the first occurrence of IN2 string is returned at OUT. If the string IN2 is not found in the string IN1, then zero is returned.

Table 9- 68 Data types for the parameters

Parameter and type		Data type	Description
IN1	IN	String, WString	Search inside this string
IN2	IN	String, WString	Search for this string
OUT	OUT	Int	Character position in string IN1 of the first search match

Table 9- 69 ENO status

ENO	Condition	OUT
1	No errors detected	Valid character position
0	IN2 is larger than IN1	Character position is set to 0
	Current length of IN1 exceeds maximum length of IN1, or current length of IN2 exceeds maximum length of IN2 (invalid string)	
	Maximum length of IN1 or IN2 does not fit within allocated memory range	
	Maximum length of IN1 or IN2 is 255 (String data type) or 65535 (WString data type)	

9.3 Distributed I/O (PROFINET, PROFIBUS, or AS-i)

9.3.1 Distributed I/O Instructions

The following Distributed I/O instructions can be used with PROFINET, PROFIBUS, or AS-i:

- RDREC instruction (Page 351): You can read a data record with the number INDEX from a module or device.
- WRREC instruction (Page 351): You can transfer a data record with the number INDEX to a module or device defined by ID.
- RALRM instruction (Page 354): You can receive an interrupt with all corresponding information from a module or device and supply this information to its output parameters.
- DPRD_DAT instruction (Page 362): You must read consistent data areas greater than 64 bytes from a module or device with the DPRD_DAT instruction.
- DPWR_DAT instruction (Page 362): You must write consistent data areas greater than 64 bytes to a module or device with the DPWR_DAT instruction.

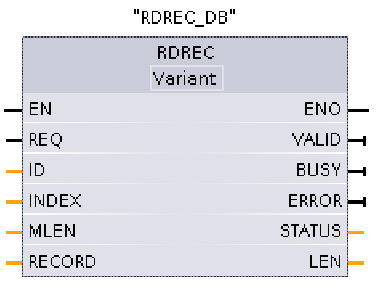
The DPNRM_DG instruction (Page 365) can only be used with PROFIBUS. You can read the current diagnostic data of a DP slave in the format specified by EN 50 170 Volume 2, PROFIBUS.

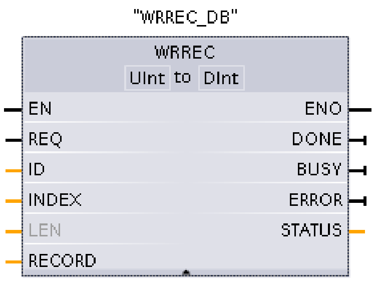
9.3.2 RDREC and WRREC (Read/write data record) instructions

You can use the RDREC (Read data record) and WRREC (Write data record) instructions with PROFINET, PROFIBUS, and AS-i.

Table 9- 70 RDREC and WRREC instructions

LAD / FBD	SCL	Description

 "RDREC_DB"
RDREC
Variant
EN
ENO
REQ
VALID
ID
BUSY
INDEX
ERROR
MLEN
STATUS
RECORD
LEN
	`"RDREC_DB"(`
	`req:=_bool_in_,`
	`ID:=_word_in_,`
	`index:=_dint_in_,`
	`mlen:=_uint_in_,`
	`valid=>_bool_out_,`
	`busy=>_bool_out_,`
	`error=>_bool_out_,`
	`status=>_dword_out_,`
	`len=>_uint_out_,`
	`rec-`
	`ord:=_variant_inout_);`
		Use the RDREC instruction to read a data record with the number INDEX from the component addressed by the ID, such as a central rack or a distributed component (PROFIBUS DP or PROFINET IO). Assign the maximum number of bytes to read in MLEN. The selected length of the target area RECORD should have at least the length of MLEN bytes.

 "WRREC_DB"
WRREC
UInt to DInt
EN
ENO
REQ
DONE
ID
BUSY
INDEX
ERROR
LEN
STATUS
RECORD
	`"WRREC_DB"(`
	`req:=_bool_in_,`
	`ID:=_word_in_,`
	`index:=_dint_in_,`
	`len:=_uint_in_,`
	`done=>_bool_out_,`
	`busy=>_bool_out_,`
	`error=>_bool_out_,`
	`status=>_dword_out_,`
	`rec-`
	`ord:=_variant_inout_);`
		Use the WRREC instruction to transfer a data RECORD with the record number INDEX to a DP slave/PROFINET IO device component addressed by ID, such as a module in the central rack or a distributed component (PROFIBUS DP or PROFINET IO).
		Assign the byte length of the data record to be transmitted. The selected length of the source area RECORD should, therefore, have at least the length of LEN bytes.

1 STEP 7 automatically creates the DB when you insert the instruction.

2 In the SCL examples, "RDREC_DB" and "WRREC_DB" are the names of the instance DBs.

Table 9- 71 RDREC and WRREC data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	REQ = 1: Transfer data record
ID	IN	HW_IO (Word)	Logical address of the DP slave/PROFINET IO component (module or submodule): • For an output module, bit 15 must be set (for example, for address 5: ID:= DW#16#8005). • For a combination module, the smaller of the two addresses should be specified. **Note**: In V3.0, the device ID can be determined in one of two ways: • By making the following "Network view" selections: – Device (gray box) – "Properties" of the device – "Hardware identifier" **Note**: Not all devices display their Hardware identifiers, however. • By making the following "Project tree" menu selections: – PLC tags – Default tag table – System constants tab All configured device Hardware identifiers are displayed. **Note**: In V4.0, the device ID (hardware identifier) for the interface module is determined by going to the tag table and locating the "Device Name [HEAD]" parameter under System Constants.
INDEX	IN	Byte, Word, USInt, UInt, SInt, Int, DInt	Data record number
MLEN	IN	Byte, USInt, UInt	Maximum length in bytes of the data record information to be fetched (RDREC)
VALID	OUT	Bool	New data record was received and valid (RDREC). The VALID bit is TRUE for one scan, after the last request was completed with no error.
DONE	OUT	Bool	Data record was transferred (WRREC). The DONE bit is TRUE for one scan, after the last request was completed with no error.
BUSY	OUT	Bool	• BUSY = 1: The read (RDREC) or write (WRREC) process is not yet terminated. • BUSY = 0: Data record transmission is completed.
ERROR	OUT	Bool	ERROR = 1: A read (RDREC) or write (WRREC) error has occurred. The ERROR bit is TRUE for one scan, after the last request was terminated with an error. The error code value at the STATUS parameter is valid only during the single scan where ERROR = TRUE.
STATUS	OUT	DWord	Block status or error information (Page 455)

Parameter and type		Data type	Description
LEN	OUT (RDREC) IN (WRREC)	UInt	• Length of the fetched data record information (RDREC) • Maximum byte length of the data record to be transferred (WRREC)
RECORD	IN_OUT	Variant	• Target area for the fetched data record (RDREC) • Data record (WRREC)

The RDREC and WRREC instructions operate asynchronously, that is, processing covers multiple instruction calls. Start the job by calling RDREC or WRREC with REQ = 1.

The job status is displayed via output parameter BUSY and the two central bytes of output parameter STATUS. The transfer of the data record is complete when the output parameter BUSY has been set to FALSE

A value of TRUE (only for one scan) on the output parameter VALID (RDREC) or DONE (WRREC) verifies that the data record has been successfully transferred into the target area RECORD (RDREC) or to the target device (WRREC). In the case of the RDREC, the output parameter LEN contains the length of the fetched data in bytes.

The output parameter ERROR (only for one scan when ERROR = TRUE) indicates that a data record transmission error has occurred. In this case, the output parameter STATUS (only for the one scan when ERROR = TRUE) contains the error information.

Data records are defined by the hardware device manufacturer. Refer to the hardware manufacturer's device documentation for details about a data record.

Note

If a DPV1 slave is configured via GSD file (GSD rev. 3 and higher) and the DP interface of the DP master is set to "S7 compatible", then you may not read any data records from the I/O modules in the user program with "RDREC" or write to the I/O modules with "WRREC". In this case, the DP master addresses the wrong slot (configured slot + 3).

Remedy: set the interface of the DP master to "DPV1".

Note

The interfaces of the "RDREC" and "WRREC" instructions are identical to the "RDREC" and "WRREC" FBs defined in "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3".

Note

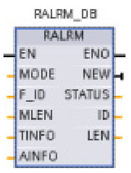
If you use "RDREC" or "WRREC" to read or write a data record for PROFINET IO, then negative values in the INDEX, MLEN, and LEN parameters will be interpreted as an unsigned 16-bit integer.

9.3.3 RALRM (Receive interrupt) instruction

You can use the RALRM (Read alarm) instruction with PROFINET and PROFIBUS.

Table 9- 72 RALRM instruction

LAD / FBD	SCL	Description

 RALRM_DB
RALRM
EN
ENO
MODE
NEW
F_ID
STATUS
MLEN
ID
TINFO
LEN
AINFO
	`"RALRM_DB"(`
	`mode:=_int_in_,`
	`f_ID:=_word_in_,`
	`mlen:=_uint_in_,`
	`new=>_bool_out_,`
	`status=>_dword_out_,`
	`ID=>_word_out_,`
	`len=>_uint_out_,`
	`tinfo:=_variant_inout_,`
	`ainfo:=_variant_inout_);`
		Use the RALRM (read alarm) instruction to read diagnostic interrupt information from PROFIBUS or PROFINET I/O modules/devices.
		The information in the output parameters contains the start information of the called OB as well as information of the interrupt source.
		Call RALRM in an interrupt OB to return information regarding the event(s) that caused the interrupt. In the S7-1200. The following Diagnostic OB interrupts are supported: Status, Update, Profile, Diagnostic error interrupt, Pull or plug of modules, Rack or station failure

1 STEP 7 automatically creates the DB when you insert the instruction.

2 In the SCL example, "RALRM_DB" is the name of the instance DB.

Table 9- 73 Data types for the parameters

Parameter and type		Data type	Description
MODE	IN	Byte, USInt, SInt, Int	Operating mode
F_ID	IN	HW_IO (Word)	Logical start address of the component (module) from which interrupts are to be received **Note**: The device ID can be determined in one of two ways: • By making the following "Network view" selections: – Device (gray box) – "Properties" of the device – "Hardware identifier" **Note**: Not all devices display their Hardware identifiers. • By making the following "Project tree" menu selections: – PLC tags – Default tag table – System constants tab – All configured device Hardware identifiers are displayed.
MLEN	IN	Byte, USInt, UInt	Maximum length in bytes of the data interrupt information to be received. MLEN of 0 will allow receipt of as much data interrupt information as is available in the AINFO Target Area.
NEW	OUT	Bool	A new interrupt was received.
STATUS	OUT	DWord	Status of the RALRM instruction. Refer to "STATUS parameter for RDREC, WRREC, and RALRM" (Page 358) for more information.

Parameter and type		Data type	Description
ID	OUT	HW_IO (Word)	Hardware identifier of the I/O module that caused the diagnostic interrupt **Note**: Refer to the F_ID parameter for an explanation of how to determine the device ID.
LEN	OUT	DWord, UInt, UDInt, DInt, Real, LReal	Length of the received AINFO interrupt information
TINFO	IN_OUT	Variant	Task information: Target range for OB start and management information. The TINFO length is always 32 bytes.
AINFO	IN_OUT	Variant	Interrupt information: Target area for header information and additional interrupt information. For AINFO, provide a length of at least the MLEN bytes, if MLEN is greater than 0. The AINFO length is variable.

Note

If you call "RALRM" in an OB whose start event is not an I/O interrupt, the instruction will provide correspondingly reduced information in its outputs.

Make sure to use different instance DBs when you call "RALRM" in different OBs. If you evaluate data resulting from a "RALRM" call outside of the associated interrupt OB, you should use a separate instance DB per OB start event.

Note

The interface of the "RALRM" instruction is identical to the "RALRM" FB defined in "PROFIBUS Guideline PROFIBUS Communication and Proxy Function Blocks according to IEC 61131-3".

Calling RALRM

You can call the RALRM instruction in three different operating modes (MODE).

Table 9- 74 RALRM instruction operating modes

MODE	Description
0	• ID contains the hardware identifier of the I/O module that triggered the interrupt. • Output parameter NEW is set to TRUE. • LEN produces an output of 0. • AINFO and TINFO are not updated with any information.
1	• ID contains the hardware identifier of the I/O module that triggered the interrupt. • Output parameter NEW is set to TRUE. • LEN produces an output of the amount in bytes of AINFO data that is returned. • AINFO and TINFO are updated with interrupt-related information.
2	If the hardware identifier assigned to input parameter F_ID has triggered the interrupt then: • ID contains the hardware identifier of the I/O module that triggered the interrupt. Should be the same as the value at F_ID. • Output parameter NEW is set to TRUE. • LEN produces an output of the amount in bytes of AINFO data that is returned. • AINFO and TINFO are updated with interrupt-related information.

Note

If you assign a destination area for TINFO or AINFO that is too short, RALRM cannot return the full information.

MLEN can limit the amount of AINFO data that is returned.

Refer to the AINFO parameters and TINFO parameters of the online information system of STEP 7 for information on how to interpret the TINFO and AINFO data.

TInfo organization block data

The table below shows how the TInfo data is arranged for the RALRM instruction:

Same for OBs: Status, Update, Profile, Diagnostic error interrupt, Pull or plug of modules, Rack or station failure	0	SI_Format	OB_Class	OB_Nr	
	4	LADDR			
TI_Submodule - OBs: Status, Update, Profile	4			Slot	
	8	Specifier		0	
TI_DiagnosticInterrupt - OB: Diagnostic error interrupt	4			IO_State	
	8	Channel		MultiError	0
TI_PlugPullModule - OB: Pull or plug of modules	4			Event_Class	Fault_ID
	8	0		0	
TI_StationFailure - OB: Rack or station failure	4			Event_Class	Fault_ID
	8	0		0	
Same for OBs: Status, Update, Profile, Diagnostic error interrupt, Pull or plug of modules, Rack or station failure	12	0			
	16				
	20	address		slv_prfl	intr_type
	24	flags1	flags2	id	
	28[1]	manufacturer		instance	

[1] Bytes 28 - 31 (manufacturer and instance) are not used with PROFIBUS.

Note

Refer to the online information system of STEP 7 for more detailed information on TINFO data.

9.3.4 STATUS parameter for RDREC, WRREC, and RALRM

The output parameter STATUS contains error information that is interpreted as ARRAY[1...4] OF BYTE, with the following structure:

Table 9- 75 STATUS output array

Array element	Name	Description
STATUS[1]	Function_Num	• B#16#00, if no error • Function ID from DPV1-PDU: If an error occurs, B#16#80 is OR'ed (for read data record: B#16#DE; for write data record: B#16#DF). If no DPV1 protocol element is used, then B#16#C0 will be output.
STATUS[2]	Error_Decode	Location of the error ID
STATUS[3]	Error_Code_1	Error ID
STATUS[4]	Error_Code_2	Manufacturer-specific error ID expansion

Table 9- 76 STATUS[2] values

Error_decode (B#16#....)	Source	Description
00 to 7F	CPU	No error or no warning
80	DPV1	Error according to IEC 61158-6
81 to 8F	CPU	B#16#8x shows an error in the "xth" call parameter of the instruction.
FE, FF	DP Profile	Profile-specific error

Table 9- 77 STATUS[3] values

Error_decode (B#16#....)	Error_code_1 (B#16#....)	Explanation (DVP1)	Description
00	00		No error, no warning
70	00	Reserved, reject	Initial call; no active data record transfer
	01	Reserved, reject	Initial call; data record transfer has started
	02	Reserved, reject	Intermediate call; data record transfer already active
80	90	Reserved, pass	Invalid logical start address
	92	Reserved, pass	Illegal type for Variant pointer
	93	Reserved, pass	The DP component addressed via ID or F_ID is not configured.
	96		The "RALRM (Page 354)" cannot supply the OB start information, management information, header information, or additional interrupt information. For the following OBs, you can use the "DPNRM_DG (Page 365)" instruction to read the current diagnostics message frame of the relevant DP slave asynchronously (address information from OB start information): • Hardware interrupt (Page 90) • Status (Page 97), Update (Page 98) or Profile (Page 98) • Diagnostic error interrupt (Page 92) • Pull or plug of modules (Page 95)
	A0	Read error	Negative acknowledgement while reading from the module
	A1	Write error	Negative acknowledgement while writing to the module
	A2	Module failure	DP protocol error at layer 2 (for example, slave failure or bus problems)
	A3	Reserved, pass	• PROFIBUS DP: DP protocol error with Direct-Data-Link-Mapper or User-Interface/User • PROFINET IO: General CM error
	A4	Reserved, pass	Communication on the communication bus disrupted
	A5	Reserved, pass	-
	A7	Reserved, pass	DP slave or modules is occupied (temporary error).
	A8	Version conflict	DP slave or module reports non-compatible versions.
	A9	Feature not supported	Feature not supported by DP slave or module
	AA to AF	User specific	DP slave or module reports a manufacturer-specific error in its application. Please check the documentation from the manufacturer of the DP slave or module.
	B0	Invalid index	Data record not known in module; illegal data record number ≥ 256

Error_decode (B#16#....)	Error_code_1 (B#16#....)	Explanation (DVP1)	Description
	B1	Write length error	The length information in the RECORD parameter is incorrect. • With "RALRM": Length error in AINFO **Note**: Refer to the online information system of STEP 7 for immediate access to information on how to interpret the "AINFO" returned buffers. • With "RDREC (Page 351)" and "WRREC (Page 351)": Length error in "MLEN"
	B2	Invalid slot	The configured slot is not occupied.
	B3	Type conflict	Actual module type does not match specified module type.
	B4	Invalid area	DP slave or module reports access to an invalid area.
	B5	Status conflict	DP slave or module not ready
	B6	Access denied	DP slave or module denies access.
	B7	Invalid range	DP slave or module reports an invalid range for a parameter or value.
	B8	Invalid parameter	DP slave or module reports an invalid parameter.
	B9	Invalid type	DP slave or module reports an invalid type: • With "RDREC (Page 351)": Buffer too small (subsets cannot be read) • With "WRREC (Page 351)": Buffer too small (subsets cannot be written)
	BA to BF	User specific	DP slave or module reports a manufacturer-specific error when accessing. Please check the documentation from the manufacturer of the DP slave or module.
	C0	Read constraint conflict	• With "WRREC (Page 351)": The data can only be written when the CPU is in STOP mode. **Note**: This means that data cannot be written by the user program. You can only write the data online with a PG/PC. • With "RDREC (Page 351)": The module routes the data record, but either no data is present or the data can only be read when the CPU is in STOP mode. **Note**: If data can only be read when the CPU is in STOP mode, no evaluation by the user program is possible. In this case, you can only read the data online with a PG/PC.
	C1	Write constraint conflict	The data of the previous write request to the module for the same data record has not yet been processed by the module.
	C2	Resource busy	The module is currently processing the maximum possible number of jobs for a CPU.
	C3	Resource unavailable	The required operating resources are currently occupied.

Error_decode (B#16#....)	Error_code_1 (B#16#....)	Explanation (DVP1)	Description
	C4		Internal temporary error. Job could not be carried out. Repeat the job. If this error occurs often, check your installation for sources of electrical interference.
	C5		DP slave or module not available
	C6		Data record transfer was cancelled due to priority class cancellation.
	C7		Job aborted due to warm or cold restart on the DP master.
	C8 to CF		DP slave or module reports a manufacturer-specific resource error. Please check the documentation from the manufacturer of the DP slave or module.
	Dx	User specific	DP Slave specific. Refer to the description of the DP Slave.
81	00 to FF		Error in the initial call parameter (with "RALRM (Page 354)": MODE)
	00		Illegal operating mode
82	00 to FF		Error in the second call parameter
88	00 to FF		Error in the eighth call parameter (with "RALRM (Page 354)": TINFO) **Note**: Refer to the online information system of STEP 7 for immediate access to information on how to interpret the "TINFO" returned buffers.
	01		Wrong syntax ID
	23		Quantity structure exceeded or destination area too small
	24		Wrong range ID
	32		DB/DI number out of user range
	3A		DB/DI number is NULL for area ID DB/DI, or specified DB/DI does not exist.
89	00 to FF		Error in the ninth call parameter (with "RALRM (Page 354)": AINFO) **Note**: Refer to the online information system of STEP 7 for immediate access to information on how to interpret the "AINFO" returned buffers.
	01		Wrong syntax ID
	23		Quantity structure exceeded or destination area too small
	24		Wrong range ID
	32		DB/DI number out of user range
	3A		DB/DI number is NULL for area ID DB/DI, or specified DB/DI does not exist.
8A	00 to FF		Error in the 10th call parameter
8F	00 to FF		Error in the 15th call parameter
FE, FF	00 to FF		Profile-specific error

Array element STATUS[4]

With DPV1 errors, the DP Master passes on STATUS[4] to the CPU and to the instruction. Without a DPV1 error, this value is set to 0, with the following exceptions for the RDREC:

- STATUS[4] contains the target area length from RECORD, if MLEN > the destination area length from RECORD.
- STATUS[4]=MLEN, if the actual data record length < MLEN < the destination area length from RECORD.
- STATUS[4]=0, if STATUS[4] > 255; would have to be set

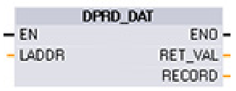
In PROFINET IO, STATUS[4] has the value 0.

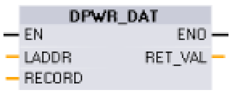
9.3.5 DPRD_DAT and DPWR_DAT (Read/write consistent data for DP slaves) instructions

You can use the DPRD_DAT (Read consistent data) and DPWR_DAT (Write consistent data) instructions with PROFINET and PROFIBUS.

Table 9- 78 DPRD_DAT and DPWR_DAT instructions

LAD / FBD	SCL	Description

	`ret_val := DPRD_DAT(`
	`laddr:=_word_in_,`
	`rec-`
	`ord=>_variant_out_);`
		Use the DPRD_DAT instruction to read one or more bytes of data from one of the following locations:
		• Module in the local base
		• DP standard slave
		• PROFINET I/O device
		The CPU transfers the data read consistently. If no errors occur during the data transfer, the CPU enters the read data into the target area set up by the RECORD parameter. The target area must have the same length as you configured with STEP 7 for the selected module. When you execute the DPRD_DAT instruction, you can only access the data of one module. The transfer starts at the configured start address.

	`ret_val := DPWR_DAT(`
	`laddr:=_word_in_,`
	`rec-`
	`ord:=_variant_in_);`
		Use the DPWR_DAT instruction to transfer the data in RECORD consistently to the following locations:
		• Addressed module in the local base
		• DP standard slave
		• PROFINET I/O device
		The source area must have the same length as you configured with STEP 7 for the selected module.

- The S7-1200 CPU supports up to 64 bytes of consistent data on the local bus. Use the DPRD_DAT and DPWR_DAT instructions to access more than 64 bytes of data consistently.
- PROFIBUS supports up to 4 bytes of consistent data. Use the DPRD_DAT and DPWR_DAT instructions to access more than 4 bytes of data consistently.

- PROFINET supports up to 1472 bytes of consistent data. You do not need to use these instructions for consistent transfers between the S7-1200 and PROFINET devices.
- You can use these instructions for data area of 1 or more bytes. If the access is rejected, error code W#16#8090 results.

Note

If you are using the DPRD_DAT and DPWR_DAT instructions with consistent data, you must remove this consistent data from the process-image automatic update. Refer to "PLC concepts: Execution of the user program" (Page 79) for more information.

Table 9- 79 Data types for the parameters

Parameter and type		Data type	Description
LADDR	IN	HW_IO (Word)	• Configured start address from the "I" area of the module from which the data will be read (DPRD_DAT) • Configured start address from the process image output area of the module to which the data will be written (DPWR_DAT) Addresses have to be entered in hexadecimal format (for example, an input or output address of 100 means: LADDR:=W#16#64).
RECORD	OUT	Variant	Destination area for the user data that were read (DPRD_DAT) or source area for the user data to be written (DPWR_DAT). This must be exactly as large as you configured for the selected module with STEP 7.
RET_VAL	OUT	Int	If an error occurs while the function is active, the return value contains an error code.

DPRD_DAT operations

The destination area must have the same length as configured for the selected module with STEP 7. If no error occurs during the data transfer, the data that have been read are entered into the destination area identified by RECORD.

If you read from a DP standard slave with a modular design or with several DP identifiers, you can only access the data of one module/DP identifier for each DPRD_DAT instruction call, specifying the configured start address.

DPWR_DAT operations

You transfer the data in RECORD consistently to the addressed DP standard slave/PROFINET IO. The data is transferred synchronously, that is, the write process is completed when the instruction is completed.

The source area must have the same length as you configured for the selected module with STEP 7.

If the DP standard slave has a modular design, you can only access one module of the DP slave.

Table 9- 80 DPRD_DAT and DPWR_DAT error codes

Error code	Description
0000	No error occurred
8090	One of the following cases apply: • You have not configured a module for the specified logical base address. • You have ignored the restriction concerning the length of consistent data. • You have not entered the start address in the LADDR parameter in hexadecimal format.
8092	The RECORD parameter supports the following data types: Byte, Char, Word, DWord, Int, UInt, USInt, SInt, DInt, UDInt, and arrays of these types.
8093	No DP module/PROFINET IO device from which you can read (DPRD_DAT) or to which you can write (DPWR_DAT) consistent data exists at the logical address specified in LADDR.
80A0	Access error detected while the I/O devices were being accessed (DPRD_DAT).
80A1	Access error detected while the I/O devices were being accessed (DPWR_DAT).
80B0	Slave failure on external DP interface module (DPRD_DAT) and (DPWR_DAT)
80B1	The length of the specified destination (DPRD_DAT) or source (DPWR_DAT) area is not identical to the user data length configured with STEP 7 Basic.
80B2	System error with external DP interface module (DPRD_DAT) and (DPWR_DAT)
80B3	System error with external DP interface module (DPRD_DAT) and (DPWR_DAT)
80C0	The data have not yet been read by the module (DPRD_DAT).
80C1	The data of the previous write job on the module have not yet been processed by the module (DPWR_DAT).
80C2	System error with external DP interface module (DPRD_DAT) and (DPWR_DAT)
80Fx	System error with external DP interface module (DPRD_DAT) and (DPWR_DAT)
85xy	System error with external DP interface module (DPWR_DAT)
87xy	System error with external DP interface module (DPRD_DAT)
808x	System error with external DP interface module (DPRD_DAT)
8xyy	General error information Refer to "Common error codes for the 'Extended' instructions" (Page 455) for more information on general error codes.

x = parameter number

y = event number

Note

If you access DPV1 slaves, error information from these slaves can be forwarded from the DP master to the instruction.

9.3.6 DPNRM_DG (Read diagnostic data from a DP slave) instruction

You can use the DPNRM_DG (Read diagnostic data) instruction with PROFIBUS.

Table 9- 81 DPNRM_DG instruction

LAD / FBD	SCL	Description

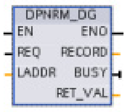
 DPNRM_DG
EN
ENO
REQ
RECORD
LADDR
BUSY
RET_VAL
	`ret_val := DPNRM_DG(`
	`    req:=_bool_in_,`
	`    laddr:=_word_in_,`
	`    record=>_variant_out_,`
	`    busy=>_bool_out_);`
		Use the DPNRM_DG instruction to read the current diagnostic data of a DP slave in the format specified by EN 50 170 Volume 2, PROFIBUS. The data that has been read is entered in the destination area indicated by RECORD following error-free data transfer.

Table 9- 82 DPNRM_DG instruction data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	REQ=1: Read request
LADDR	IN	HW_DPSLAVE	Configured diagnostic address of the DP slave: Must be the address of the station and not for the I/O device. Select the station (and not the image of the device) in the "Network" view of the "Device configuration" to determine the diagnostic address. Enter the addresses in hexadecimal format. For example, diagnostic address 1022 means LADDR:=W#16#3FE.
RET_VAL	OUT	Int	If an error occurs while the function is active, the return value contains an error code. If no error occurs, the length of the data actually transferred is entered in RET_VAL.
RECORD	OUT	Variant	Destination area for the diagnostic data that were read. The minimum length of the data record to be read (or the destination area) is 6 bytes. The maximum length of the data record to be sent is 240 bytes. Standard slaves can provide more than 240 bytes of diagnostic data up to a maximum of 244 bytes. In this case, the first 240 bytes are transferred to the destination area, and the overflow bit is set in the data.
BUSY	OUT	Bool	BUSY=1: The read job is not yet completed

You start the read job by assigning 1 to the input parameter REQ in the DPNRM_DG instruction call. The read job is executed asynchronously, in other words, it requires several DPNRM_DG instruction calls. The status of the job is indicated by the output parameters RET_VAL and BUSY.

Table 9- 83 Slave diagnostic data structure

Byte	Description
0	Station status 1
1	Station status 2
2	Station status 3
3	Master station number
4	Vendor ID (high byte)
5	Vendor ID (low byte)
6 ...	Additional slave-specific diagnostic information

Table 9- 84 DPNRM_DG instruction error codes

Error code	Description	Restriction
0000	No error	-
7000	First call with REQ=0: No data transfer active; BUSY has the value 0.	-
7001	First call with REQ =1: No data transfer active; BUSY has the value 1.	Distributed I/Os
7002	Interim call (REQ irrelevant): Data transfer already active; BUSY has the value 1.	Distributed I/Os
8090	Specified logical base address invalid: There is no base address.	-
8092	The RECORD parameter supports the following data types: Byte, Char, Word, DWord, Int, UInt, USInt, SInt, DInt, UDInt, and arrays of these types.	-
8093	• This instruction is not permitted for the module specified by LADDR (S7-DP modules for S7-1200 are permitted). • LADDR specifies the I/O device instead of specifying the station. Select the station (and not the image of the device) in the "Network" view of the "Device configuration" to determine the diagnostic address for LADDR.	-
80A2	• DP protocol error at layer 2 (for example, slave failure or bus problems) • For ET200S, data record cannot be read in DPV0 mode.	Distributed I/Os
80A3	DP protocol error with user interface/user	Distributed I/Os
80A4	Communication problem on the communication bus	The error occurs between the CPU and the external DP interface module.
80B0	• The instruction is not possible for module type. • The module does not recognize the data record. • Data record number 241 is not permitted.	-
80B1	The length specified in the RECORD parameter is incorrect.	Specified length > record length
80B2	The configured slot is not occupied.	-

Error code	Description	Restriction
80B3	Actual module type does not match the required module type.	-
80C0	There is no diagnostic information.	-
80C1	The data of the previous write job for the same data record on the module have not yet been processed by the module.	-
80C2	The module is currently processing the maximum possible number of jobs for a CPU.	-
80C3	The required resources (memory, etc.) are currently occupied.	-
80C4	Internal temporary error. The job could not be processed. Repeat the job. If this error occurs frequently, check your system for electrical disturbance sources.	-
80C5	Distributed I/Os not available	Distributed I/Os
80C6	Data record transfer was stopped due to a priority class abort (restart or background)	Distributed I/Os
8xyy[1]	General error codes	

Refer to "Extended instructions, Distributed I/O: Error information for RDREC, WRREC, and RALRM" (Page 358) for more information on general error codes.

9.4 Interrupts

9.4.1 ATTACH and DETACH (Attach/detach an OB and an interrupt event) instructions

You can activate and deactivate interrupt event-driven subprograms with the ATTACH and DETACH instructions.

Table 9- 85 ATTACH and DETACH instructions

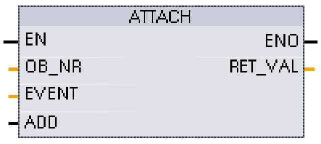
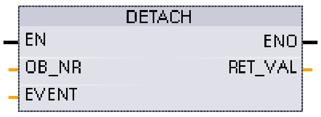
LAD / FBD	SCL	Description
	`ret_val := ATTACH( ob_nr:=_int_in_, event:=_event_att_in_, add:=_bool_in_);`	ATTACH enables interrupt OB subprogram execution for a hardware interrupt event.
	`ret_val := DETACH( ob_nr:=_int_in_, event:=_event_att_ in);`	DETACH disables interrupt OB subprogram execution for a hardware interrupt event.

Table 9- 86 Data types for the parameters

Parameter and type		Data type	Description
OB_NR	IN	OB_ATT	Organization block identifier: Select from the available hardware interrupt OBs that were created using the "Add new block" feature. Double-click on the parameter field, then click on the helper icon to see the available OBs.
EVENT	IN	EVENT_ATT	Event identifier: Select from the available hardware interrupt events that were enabled in PLC device configuration for digital inputs or high-speed counters. Double-click on the parameter field, then click on the helper icon to see the available events.
ADD (ATTACH only)	IN	Bool	• ADD = 0 (default): This event replaces all previous event attachments for this OB. • ADD = 1: This event is added to previous event attachments for this OB.
RET_VAL	OUT	Int	Execution condition code

Hardware interrupt events

The following hardware interrupt events are supported by the CPU:

- Rising edge events: first 12 built-in CPU digital inputs (DIa.0 to DIb.3) and all SB digital inputs
  - A rising edge occurs when the digital input transitions from OFF to ON as a response to a change in the signal from a field device connected to the input.
- Falling edge events: first 12 built-in CPU digital inputs (DIa.0 to DIb.3) and all SB digital inputs
  - A falling edge occurs when the digital input transitions from ON to OFF.
- High-speed counter (HSC) current value = reference value (CV = RV) events (HSC 1 through 6)
  - A CV = RV interrupt for a HSC is generated when the current count transitions from an adjacent value to the value that exactly matches a reference value that was previously established.
- HSC direction changed events (HSC 1 through 6)
  - A direction changed event occurs when the HSC is detected to change from increasing to decreasing, or from decreasing to increasing.
- HSC external reset events (HSC 1 through 6)
  - Certain HSC modes allow the assignment of a digital input as an external reset that is used to reset the HSC count value to zero. An external reset event occurs for such a HSC, when this input transitions from OFF to ON.

Enabling hardware interrupt events in the device configuration

Hardware interrupts must be enabled during the device configuration. You must check the enable-event box in the device configuration for a digital input channel or a HSC, if you want to attach this event during configuration or run time.

Check box options within the PLC device configuration:

- Digital input
  - Enable rising edge detection
  - Enable falling edge detection
- High-speed counter (HSC)
  - Enable this high-speed counter for use
  - Generate interrupt for counter value equals reference value count
  - Generate interrupt for external reset event
  - Generate interrupt for direction change event

Adding new hardware interrupt OB code blocks to your program

By default, no OB is attached to an event when the event is first enabled. This is indicated by the "HW interrupt:" device configuration "<not connected>" label. Only hardware-interrupt OBs can be attached to a hardware interrupt event. All existing hardware-interrupt OBs appear in the "HW interrupt:" drop-down list. If no OB is listed, then you must create an OB of type "Hardware interrupt" as follows. Under the project tree "Program blocks" branch:

1. Double-click "Add new block", select "Organization block (OB)" and choose "Hardware interrupt".
2. Optionally, you can rename the OB, select the programming language (LAD, FBD or SCL), and select the block number (switch to manual and choose a different block number than that suggested).
3. Edit the OB and add the programmed reaction that you want to execute when the event occurs. You can call FCs and FBs from this OB, to a nesting depth of six.

OB_NR parameter

All existing hardware-interrupt OB names appear in the device configuration "HW interrupt:" drop-down list and in the ATTACH / DETACH parameter OB_NR drop-list.

EVENT parameter

When a hardware interrupt event is enabled, a unique default event name is assigned to this particular event. You can change this event name by editing the "Event name:" edit box, but it must be a unique name. These event names become tag names in the "Constants" tag table, and appear on the EVENT parameter drop-down list for the ATTACH and DETACH instruction boxes. The value of the tag is an internal number used to identify the event.

General operation

Each hardware event can be attached to a hardware-interrupt OB which will be queued for execution when the hardware interrupt event occurs. The OB-event attachment can occur at configuration time or at run time.

You have the option to attach or detach an OB to an enabled event at configuration time. To attach an OB to an event at configuration time, you must use the "HW interrupt:" drop-down list (click on the down arrow on the right) and select an OB from the list of available hardware-interrupt OBs. Select the appropriate OB name from this list, or select "<not connected>" to remove the attachment.

You can also attach or detach an enabled hardware interrupt event during run time. Use the ATTACH or DETACH program instructions during run time (multiple times if you wish) to attach or detach an enabled interrupt event to the appropriate OB. If no OB is currently attached (either from a "<not connected>" selection in device configuration, or as a result of executing a DETACH instruction), the enabled hardware interrupt event is ignored.

DETACH operation

Use the DETACH instruction to detach either a particular event or all events from a particular OB. If an EVENT is specified, then only this one event is detached from the specified OB_NR; any other events currently attached to this OB_NR will remain attached. If no EVENT is specified, then all events currently attached to OB_NR will be detached.

Condition codes

Table 9- 87 Condition codes

RET_VAL (W#16#....)	ENO	Description
0000	1	No error
0001	1	Nothing to Detach (DETACH only)
8090	0	OB does not exist
8091	0	OB is wrong type
8093	0	Event does not exist

9.4.2 Cyclic interrupts

9.4.2.1 SET_CINT (Set cyclic interrupt parameters) instruction

Table 9- 88 SET_CINT (Set cyclic interrupt parameters)

LAD / FBD	SCL	Description

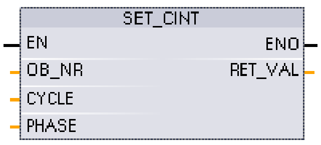
	`ret_val := SET_CINT(`
	`    ob_nr:=_int_in_,`
	`    cycle:=_udint_in_,`
	`    phase:=_udint_in_);`
		Set the specified interrupt OB to begin cyclic execution that interrupts the program scan.

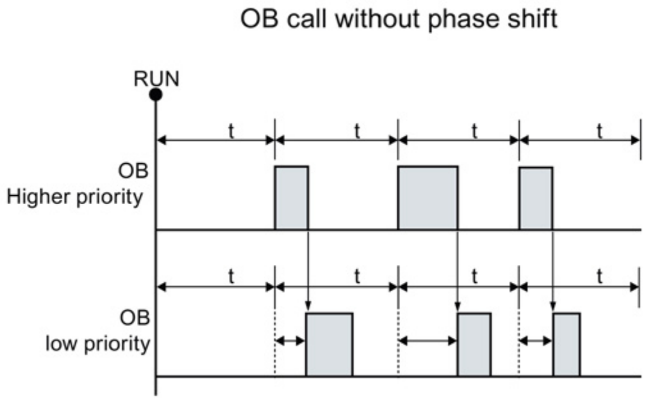
Table 9- 89 Data types for the parameters

Parameter and type		Data type	Description
OB_NR	IN	OB_CYCLIC	OB number (accepts symbolic name)
CYCLE	IN	UDInt	Time interval, in microseconds
PHASE	IN	UDInt	Phase shift, in microseconds
RET_VAL	OUT	Int	Execution condition code

Examples: time parameter

- If the CYCLE time = 100 us, then the interrupt OB referenced by OB_NR interrupts the cyclic program scan every 100 us. The interrupt OB executes and then returns execution control to the program scan, at the point of interruption.
- If the CYCLE time = 0, then the interrupt event is deactivated and the interrupt OB is not executed.
- The PHASE (phase shift) time is a specified delay time that occurs before the CYCLE time interval begins. You can use the phase shift to control the execution timing of lower priority OBs.

If lower and higher priority OBs are called in the same time interval, the lower priority OB is only called after the higher priority OB has finished processing. The execution start time for the low priority OB can shift depending on the processing time of higher priority OBs.



If you want to start the execution of a lower priority OB on a fixed time cycle, then phase shift time should be greater then the processing time of higher priority OBs.

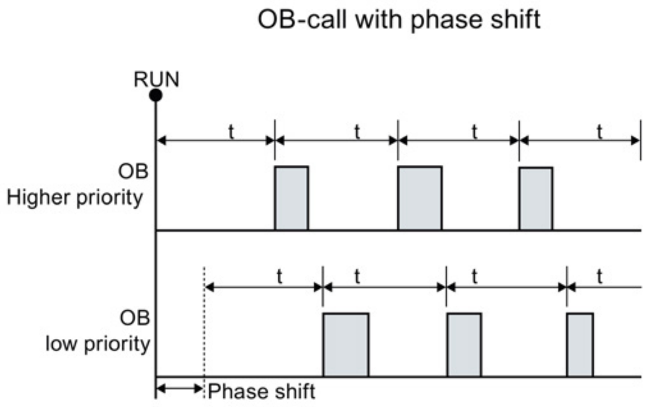


Table 9- 90 Condition codes

RET_VAL (W#16#....)	Description
0000	No error
8090	OB does not exist or is of wrong type
8091	Invalid cycle time
8092	Invalid phase shift time
80B2	OB has no attached event

9.4.2.2 QRY_CINT (Query cyclic interrupt parameters) instruction

Table 9- 91 **QRY_CINT (Query cyclic interrupt)**

LAD / FBD	SCL	Description

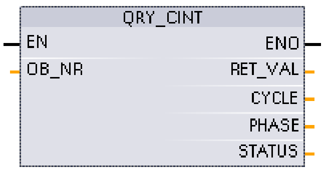
 QRY_CINT
EN
ENO
OB_NR
RET_VAL
CYCLE
PHASE
STATUS
	`ret_val := QRY_CINT(`
	`    ob_nr:=_int_in_,`
	`    cycle=>_udint_out_,`
	`    phase=>_udint_out__,`
	`    status=>_word_out_);`
		Get parameter and execution status from a cyclic interrupt OB. The values that are returned existed at the time QRY_CINT was executed.

Table 9- 92 Data types for the parameters

Parameter and type		Data type	Description
OB_NR	IN	OB_CYCLIC	OB number (accepts symbolic name like OB_MyOBName)
RET_VAL	OUT	Int	Execution condition code
CYCLE	OUT	UDInt	Time interval, in microseconds
PHASE	OUT	UDInt	Phase shift, in microseconds
STATUS	OUT	Word	Cyclic interrupt status code: • Bits 0 to 4, see the STATUS table below • Other bits, always 0

Table 9- 93 STATUS parameter

Bit	Value	Description
0	0	During CPU RUN
	1	During startup
1	0	The interrupt is enabled.
	1	Interrupt is disabled via the DIS_IRT instruction.
2	0	The interrupt is not active or has elapsed.
	1	The interrupt is active.
4	0	The OB identified by OB_NR does not exist.
	1	The OB identified by OB_NR exists.
Other Bits		Always 0

If an error occurs, RET_VAL displays the appropriate error code and the parameter STATUS = 0.

Table 9- 94 RET_VAL parameter

RET_VAL (W#16#....)	Description
0000	No error
8090	OB does not exist or is of wrong type.
80B2	OB has no attached event.

9.4.3 Time of day interrupts

WARNING

If an attacker can access your networks through Network Time Protocol (NTP) synchronization, the attacker can possibly take limited control of your process by shifting the CPU system time.

The NTP client feature of the S7-1200 CPU is disabled by default, and, when enabled, only allows configured IP addresses to act as an NTP server. The CPU disables this feature by default, and you must configure this feature to allow remotely-controlled CPU system time corrections.

The S7-1200 CPU supports "time of day" interrupts and clock instructions that depend upon accurate CPU system time. If you configure NTP and accept time synchronization from a server, you must ensure that the server is a trusted source. Failure to do so can cause a security breach that allows an unknown user to take limited control of your process by shifting the CPU system time.

For security information and recommendations, please see our "Operational Guidelines for Industrial Security" (http://www.industry.siemens.com/topics/global/en/industrial-security/Documents/operational_guidelines_industrial_security_en.pdf) on the Siemens Service and Support site.

9.4.3.1 SET_TINTL (Set time of day interrupt)

Table 9- 95 SET_TINTL (Set date and time of day interrupt with DTL data type)

LAD / FBD	SCL	Description

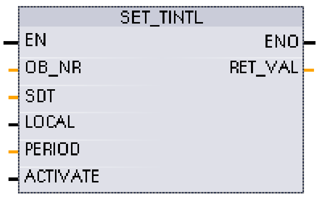
 SET_TINTL
EN
ENO
OB_NR
RET_VAL
SDT
LOCAL
PERIOD
ACTIVATE
	`ret_val := SET_TINTL(`
	`OB_NR:=_int_in_,`
	`SDT:=_dtl_in_,`
	`LOCAL:=_bool_in_`
	`PERIOD:=_word_in_`
	`ACTIVATE:=_bool_in_);`
		Set a date and time of day interrupt. The program interrupt OB can be set for one execution, or for recurring execution with an assigned time period.

Table 9- 96 Data types for the parameters

Parameter and type		Data type	Description
OB_NR	IN	OB_TOD (INT)	OB number (accepts symbolic name)
SDT	IN	DTL	Start date and time: Seconds and milliseconds are ignored and can be set to 0.
LOCAL	IN	Bool	0 = Use system time 1 = Use local time (if the CPU is configured for local time, otherwise use system time)
PERIOD	IN	Word	The period from the starting date and time for recurring interrupt event. • W#16#0000 = Once • W#16#0201 = Every minute • W#16#0401 = Every hour • W#16#1001 = Daily • W#16#1201 = Weekly • W#16#1401 = Monthly • W#16#1801 = yearly • W#16#2001 = End of month
ACTIVATE	IN	Bool	0 = ACT_TINT must be executed to activate the interrupt event. 1 = The interrupt event is activated.
RET_VAL	OUT	Int	Execution condition code

Your program can use SET_TINTL to set a date and time of day interrupt event that will execute the assigned interrupt OB. The start date and time is set by parameter SDT and the time period for recurring interrupts (for example, daily or weekly) is set by parameter PERIOD. If you set the repetition period to monthly, then you must set the start date to a day from 1 to 28. The days 29 to 31 may not be used because they do not occur in February. If you want an interrupt event at the end of each month, then use end of month for parameter PERIOD.

The DTL data weekday value in parameter SDT is ignored. Set a CPU's current date and time using the "Set time of day" function in the "Online & diagnostics" view of an online CPU. You must set the month, day of month, and year. STEP 7 calculates the interrupt period based on the CPU date and time clock.

Note

The first hour of the day does not exist when changing from summer to winter (daylight saving time). Use a start time within the second hour or use an additional time delay interrupt within the first hour.

Table 9- 97 Condition code

RET_VAL (W#16#....)	Description
0000	No error
8090	Invalid OB_NR parameter
8091	Invalid SDT start time parameter: (for example, a start time within the skipped hour at the start of daylight savings time)
8092	Invalid PERIOD parameter
80A1	The start time is in the past. (This error code only occurs with PERIOD = W #16#0000.)

9.4.3.2 CAN_TINT (Cancel time of day interrupt)

Table 9- 98 CAN_TINT (Cancel date and time of day interrupt)

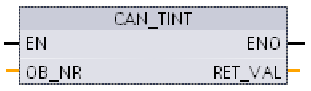
LAD / FBD	SCL	Description
	`ret_val:=CAN_TINT(_int_in);`	Cancels the start date and time of day interrupt event for the specified interrupt OB.

Table 9- 99 Data types for the parameters

Parameter and type		Data type	Description
OB_NR	IN	OB_TOD (INT)	OB number (accepts symbolic name)
RET_VAL	OUT	Int	Execution condition code

Table 9- 100 Condition codes

RET_VAL (W#16#....)	Description
0000	No error
8090	Invalid OB_NR parameter
80A0	No start date / time set for that interrupt OB

9.4.3.3 ACT_TINT (Activate time of day interrupt)

Table 9- 101 ACT_TINT (Activate date and time of day interrupt)

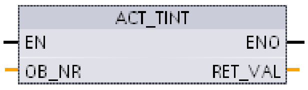
LAD / FBD	SCL	Description
	`ret_val:=ACT_TINT(_int_in_);`	Activates the start date and time of day interrupt event for the specified interrupt OB.

Table 9- 102 Data types for the parameters

Parameter and type		Data type	Description
OB_NR	IN	OB_TOD (INT)	OB number (accepts symbolic name)
RET_VAL	OUT	Int	Execution condition code

Table 9- 103 Condition codes

RET_VAL (W#16#....)	Description
0000	No error
8090	Invalid OB_NR parameter
80A0	Start date and time-of day not set, for the relevant time-of-day interrupt OB
80A1	The activated time is in the past. The error only occurs when the interrupt OB is set to execute once only.

9.4.3.4 QRY_TINT (Query status of time of day interrupt)

Table 9- 104 QRY_TINT (Query date and time of day interrupt)

LAD / FBD	SCL	Description

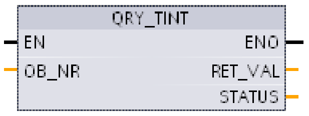
	`ret_val:=QRY_TINT(`
	`    OB_NR:=_int_in_,`
	`    STATUS=>_word_out_);`
		Queries the date and time of day interrupt status for the specified interrupt OB.

Table 9- 105 Data types for the parameters

Parameter and type		Data type	Description
OB_NR	IN	OB_TOD (INT)	OB number (accepts symbolic name) of the interrupt OB to query
RET_VAL	OUT	Int	Execution condition code
STATUS	OUT	Word	Status of the specified interrupt OB

Table 9- 106 STATUS parameter

Bit	Value	Description
0	0	In Run
	1	In Startup
1	0	The interrupt is enabled.
	1	The interrupt is disabled.
2	0	The interrupt is not active or has expired.
	1	The interrupt is active.
4	0	The assigned OB_NR does not exist.
	1	An OB with the assigned OB_NR exists.
6	1	The date and time of day interrupt uses local time.
	0	The date and time of day interrupt uses system time.
Others		Always 0

Table 9- 107 Condition code

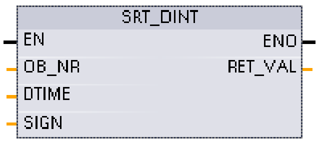
RET_VAL (W#16#....)	Description
0000	No error
8090	Invalid OB_NR parameter

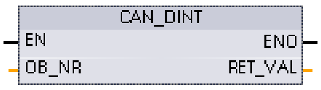
9.4.4 Time delay interrupts

You can start and cancel time delay interrupt processing with the SRT_DINT and CAN_DINT instructions, or query the interrupt status with the QRY_DINT instruction. Each time delay interrupt is a one-time event that occurs after the specified delay time. If the time delay event is cancelled before the time delay expires, the program interrupt does not occur.

Table 9- 108 SRT_DINT, CAN_DINT, and QRY_DINT instructions

LAD / FBD	SCL	Description

	`ret_val := SRT_DINT(`
	`ob_nr:=_int_in_,`
	`dtime:=_time_in_,`
	`sign:=_word_in_);`
		SRT_DINT starts a time delay interrupt that executes an OB when the delay time specified by parameter DTIME has elapsed.

	`ret_val := CAN_DINT(`
	`ob_nr:=_int_in_);`
		CAN_DINT cancels a time delay interrupt that has already started. The time delay interrupt OB is not executed in this case.

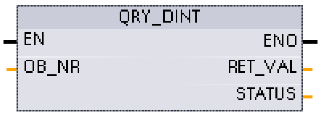
	`ret_val := QRY_DINT(`
	`ob_nr:=_int_in_,`
	`sta-`
	`tus=>_word_out_);`
		QRY_DINT queries the status of the time delay interrupt specified by the OB_NR parameter.

Table 9- 109 Data types for the parameters

Parameter and type		Data type	Description
OB_NR	IN	OB_DELAY	Organization block (OB) to be started after a time-delay: Select from the available time-delay interrupt OBs that were created using the "Add new block" project tree feature. Double-click on the parameter field, then click on the helper icon to see the available OBs.
DTIME [1]	IN	Time	Time delay value (1 to 60000 ms)
SIGN [1]	IN	Word	Not used by the S7-1200: Any value is accepted. A value must be assigned to prevent errors.
RET_VAL	OUT	Int	Execution condition code
STATUS	OUT	Word	QRY_DINT instruction: Status of the specified time-delay interrupt OB, see the table below

[1] Only for SRT_DINT

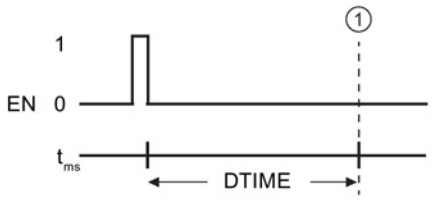
Operation

When EN=1, the SRT_DINT instruction starts the internal time delay timer (DTIME). When the time delay elapses, the CPU generates a program interrupt that triggers the execution of the associated time delay interrupt OB. You can cancel an in-process time delay interrupt before the specified time delay occurs by executing the CAN_DINT instruction. The total number of active time delay interrupt events must not exceed four.

Note

The SRT_DINT starts the time delay timer on every scan when EN=1. Assert EN=1 as a one-shot rather than just setting EN=1 to begin your time delay.

Timing diagram for the SRT_DINT instruction:



① Time delay interrupt executes

Adding time delay interrupt OBs to your project

you can only assign time delay interrupt OBs to the SRT_DINT and CAN_DINT instructions. No time delay interrupt OB exists in a new project. You must add time delay interrupt OBs to your project. To create a time-delay interrupt OB, follow these steps:

1. Double-click the "Add new block" item in the "Program blocks" branch of the project tree, select "Organization block (OB)", and choose "Time delay interrupt".
2. You have the option to rename the OB, select the programming language, or select the block number. Switch to manual numbering if you want to assign a different block number than the number that was assigned automatically.
3. Edit the time delay interrupt OB subprogram and create programmed reaction that you want to execute when the time delay timeout event occurs. You can call other FC and FB code blocks from the time delay interrupt OB, with a maximum nesting depth of six.
4. The newly assigned time delay interrupt OB names will be available when you edit the OB_NR parameter of the SRT_DINT and CAN_DINT instructions.

QRY_DINT parameter STATUS

Table 9- 110 If there is an error (REL_VAL <> 0), then STATUS = 0.

Bit	Value	Description
0	0	In RUN
	1	In startup
1	0	The interrupt is enabled.
	1	The interrupt is disabled.
2	0	The interrupt is not active or has elapsed.
	1	The interrupt is active.
4	0	An OB with an OB number given in OB_NR does not exist.
	1	An OB with an OB number given in OB_NR exists.
Other bits		Always 0

Condition codes

Table 9- 111 Condition codes for SRT_DINT, CAN_DINT, and QRY_DINT

RET_VAL (W#16#...)	Description
0000	No error occurred
8090	Incorrect parameter OB_NR
8091	Incorrect parameter DTIME
80A0	Time delay interrupt has not started.

9.4.5 DIS_AIRT and EN_AIRT (Delay/enable execution of higher priority interrupts and asynchronous error events) instructions

Use the DIS_AIRT and EN_AIRT instructions to disable and enable alarm interrupt processing.

Table 9- 112 DIS_AIRT and EN_AIRT instructions

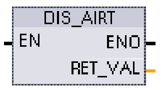
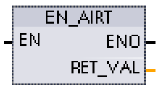
LAD / FBD	SCL	Description
	**`DIS_AIRT();`**	DIS_AIRT delays the processing of new interrupt events. You can execute DIS_AIRT more than once in an OB.
	**`EN_AIRT();`**	EN_AIRT enables the processing of interrupt events that you previously disabled with the DIS_AIRT instruction. Each DIS_AIRT execution must be cancelled by an EN_AIRT execution. The EN_AIRT executions must occur within the same OB, or any FC or FB called from the same OB, before interrupts are enabled again for this OB.

Table 9- 113 Data types for the parameters

Parameter and type		Data type	Description
RET_VAL	OUT	Int	Number of delays = number of DIS_AIRT executions in the queue.

The DIS_AIRT executions are counted by the operating system. Each of these remains in effect until it is cancelled again specifically by an EN_AIRT instruction, or until the current OB has been completely processed. For example: if you disabled interrupts five times with five DIS_AIRT executions, you must cancel these with five EN_AIRT executions before interrupts become enabled again.

After the interrupt events are enabled again, the interrupts that occurred while DIS_AIRT was in effect are processed, or the interrupts are processed as soon as the current OB has been executed.

Parameter RET_VAL indicates the number of times that interrupt processing was disabled, which is the number of queued DIS_AIRT executions. Interrupt processing is only enabled again when parameter RET_VAL = 0.

9.5 Diagnostics (PROFINET or PROFIBUS)

9.5.1 Diagnostic instructions

The following diagnostic instructions can be used with either PROFINET or PROFIBUS:

- LED instruction (Page 385): You can read the state of the LEDs for a distributed I/O device.
- DeviceStates instruction (Page 386): You can retrieve the operational states for a distributed I/O device within an I/O subsystem.
- ModuleStates instruction (Page 392): You can retrieve the operational states for the modules in a distributed I/O device.
- GET_DIAG instruction (Page 398): You can read the diagnostic information from a specified device.
- Get_IM_Data instruction (Page 404): You can check the identification and maintenance (I&M) data for a specified module or sub-module.

9.5.2 Diagnostic events for distributed I/O

Note

With a PROFIBUS IO system, after a download or power cycle, the CPU will go to RUN mode unless the hardware compatibility is set to allow acceptable substitute modules (Page 162) and one or more modules is missing or is not an acceptable substitute for the configured module.

As shown in the following table, the CPU supports diagnostics that can be configured for the components of the distributed I/O system. Each of these errors generates a log entry in the diagnostic buffer.

Table 9- 114 Handling of diagnostic events for PROFINET and PROFIBUS

Type of error	Diagnostic information for the station?	Entry in the diagnostic buffer?	CPU operating mode
Diagnostic error	Yes	Yes	Stays in RUN mode
Rack or station failure	Yes	Yes	Stays in RUN mode
I/O access error [1]	No	Yes	Stays in RUN mode
Peripheral access error [2]	No	Yes	Stays in RUN mode
Pull / plug event	Yes	Yes	Stays in RUN mode

[1] I/O access error example cause: A module that has been removed.

[2] Peripheral access error example cause: Acyclic communication to a submodule that is not communicating.

Use the GET_DIAG instruction (Page 398) for each station to obtain the diagnostic information. This will allow you to programmatically handle the errors encountered on the device and if desired take the CPU to STOP mode. This method requires you to specify the hardware device from which to read the status information.

The GET_DIAG instruction uses the "L address" (LADDR) of the station to obtain the health of the entire station. This L Address can be found within the Network Configuration view and by selecting the entire station rack (entire gray area), the L Address is shown in the Properties Tab of the station. You can find the LADDR for each individual module either in the properties for the module (in the device configuration) or in the default tag table for the CPU.

9.5.3 LED (Read LED status) instruction

Table 9- 115 LED instruction

LAD / FBD	SCL	Description

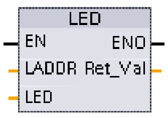
	`ret_val := LED(`
	`     laddr:=_word_in_,`
	`     LED:=_uint_in_);`
		Use the LED instruction to read the state of the LEDs on a CPU or interface. The specified LED state is returned by the RET_VAL output.

Table 9- 116 Data types for the parameters

Parameter and type		Data type	Description		
LADDR	IN	HW_IO	Identification number of the CPU or interface[1]		
LED	IN	UInt	LED identifier number		
			1	RUN/STOP	Color 1 = green, color 2 = yellow
			2	Error	Color 1 = red
			3	Maintenance	Color 1 = yellow
			4	Redundancy	Not applicable
			5	Link	Color 1 = green
			6	Tx/Rx	Color 1 = yellow
RET_VAL	OUT	Int	Status of the LED		

[1] For example, you can select the CPU (such as "PLC_1") or the PROFINET interface from the drop-down list of the parameter.

Table 9- 117 Status of RET_VAL

RET_VAL (W#16#...)	Description	
0 to 9 LED state	0	LED does not exist
	1	Off
	2	Color 1 On (solid)
	3	Color 2 On (Solid)
	4	Color 1 flashing at 2 Hz
	5	Color 2 flashing 2 Hz
	6	Color 1 & 2 flashing alternatively at 2 Hz
	7	Color 1 on (Tx/Rx)
	8	Color 2 on (Tx/Rx)
	9	State of the LED is not available
8091	Device identified by LADDR does not exist	
8092	Device identified by LADDR does not support LEDs	
8093	LED identifier not defined	
80Bx	CPU identified by LADDR does not support the LED instruction	

9.5.4 DeviceStates instruction

You can use the DeviceStates instruction to return the states of all distributed I/O slave devices connected to a specified distributed I/O Master.

Table 9- 118 DeviceStates instruction

LAD / FBD	SCL	Description

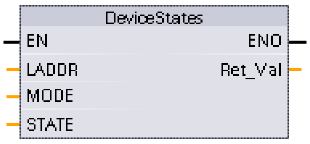
 DeviceStates
EN
ENO
LADDR
Ret_Val
MODE
STATE
	`ret_val := DeviceStates(`
	`    laddr:=_word_in_,`
	`    mode:=_uint_in_,`
	`    state:=_variant_inout_);`
		DeviceStates retrieves the I/O device operational states of an I/O subsystem. After execution, the STATE parameter contains the error state of each I/O device in a bit list (for the assigned LADDR and MODE). This information corresponds with the device status seen in the STEP 7 diagnostics view.
		The LADDR input of DeviceStates uses the hardware identifier of a distributed I/O interface. In the TIA portal, the hardware identifiers for a PLC can be found by looking for "Hw_IoSystem" data types in the system constants tab in the PLC tag table.

Table 9- 119 Data types for the parameters

Parameter and type		Data type	Description
LADDR	IN	HW_IOSYSTEM	Logical address: (Identifier for the I/O system)
MODE	IN	UInt	Supports five modes of operation. The MODE input determines which data will be returned to the location specified for STATE information. The modes are as follows: • 1: Device configuration active • 2: Device defective • 3: Device disabled • 4: Device exists • 5: Problem in Device
RET_VAL	OUT	Int	Execution condition code
STATE[1]	InOut	Variant	Buffer that receives the error status of each device: The data type that you choose for the STATE parameter can be any bit type (Bool, Byte, Word, or DWord) or an array of a bit type • Bit 0 of the first byte of the returned STATE data is a summary bit. When it is set to TRUE, it indicates that other data is available. • The data returned by the STATE parameter shows a one-to-one correlation between a bit location and a distributed I/O address. This device addressing is TRUE for PROFIBUS and PROFINET. For example, Bit 4 in the first Byte correlates to PROFIBUS address 4 or PROFINET device number 4.

[1] For PROFIBUS-DP, the length of the status information is 128 bits. For PROFINET I/O, the length is 1024 bits.

After execution, the STATE parameter contains the error state of each I/O device as a bit list (for the assigned LADDR and MODE).

Table 9- 120 Condition codes

RET_VAL (W#16#...)	Description
0	No error
8091	LADDR does not exist.
8092	LADDR does not address an I/O system.
8093	Invalid data type assigned for STATE parameter: Valid data types are (Bool, Byte, Word, or Dword), or an array of (Bools, Bytes, Words, or Dwords)
80Bx	DeviceStates instruction not supported by the CPU for this LADDR.
8452	The complete state data is too large for the assigned STATE parameter. The STATE buffer contains a partial result.

9.5.4.1 DeviceStates example configurations

PROFIBUS example

The PROFIBUS example consists of the following:

- 16 PROFIBUS devices named "DPSlave_10" through "DPSlave_25"
- The 16 PROFIBUS devices use PROFIBUS addresses 10 through 25, respectively.
- Each slave device is configured with multiple I/O modules.
- The first four bytes of the returned STATE parameter information is displayed.

MODE	Example 1: Normal operation with no errors	Example 2: PROFIBUS slave device DPSlave_12 with single module pulled	Example 3: PROFIBUS slave device DPSlave_12 disconnected
1: Device configuration active	0x01FC_FF03	0x01FC_FF03	0x01FC_FF03
2: Device defective	0x0000_0000	0x0110_0000	0x0110_0000
3: Device disabled	0x0000_0000	0x0000_0000	0x0000_0000
4: Device exists	0x01FC_FF03	0x01FC_FF03	0x01EC_FF03
5: Problem in device	0x0000_0000	0x0110_0000	0x0110_0000

The following four tables show a binary breakdown of the four bytes of data that are being analyzed:

Table 9- 121 Example 1: No errors: A value of 0x01FC_FF03 is returned for MODE 1 (Device configuration active).

Byte with value	Bit pattern with value	Notes
Byte 1 0x01	Bit 7 0000-0001 Bit 0	Bit 0 is true; data is available.
Byte 2 0xFC	Bit 15 1111-1100 Bit 8	
Byte 3 0xFF	Bit 23 1111-1111 Bit 16	
Byte 4 0x03	Bit 31 0000-0011 Bit 24	

The devices are configured in addresses 10 (Bit 10) through 25 (Bit 25).
No devices are configured in addresses 1 through 9.
MODE 4 (Device exists) data matches MODE 1 (Device configuration active), so the configured devices match the existing devices.

Table 9- 122 Example 2: A module has been pulled from PROFIBUS slave device "DPSlave_12". A value of 0x0110_0000 is returned for MODE 2 (Device defective).

Byte with value	Bit pattern with value	Notes
Byte 1 0x01	Bit 7 0000-0001 Bit 0	Bit 0 is true; data is available.
Byte 2 0x10	Bit 15 000**1**-0000 Bit 8	
Byte 3 0x00	Bit 23 0000-0000 Bit 16	
Byte 4 0x00	Bit 31 0000-0000 Bit 24	

Device 12 (Bit 12) is marked as defective.
MODE 5 (Problem in device) returns the same information as MODE 2 (Device defective).

Table 9- 123 Example 2 (continued): A module has been pulled from PROFIBUS slave device "DPSlave_12". A value of 0x01FC_FF03 is returned for MODE 4 (Device exists).

Byte with value	Bit pattern with value	Notes
Byte 1 0x01	Bit 7 0000-0001 Bit 0	Bit 0 is true; data is available.
Byte 2 0xFC	Bit 15 111**1**-1100 Bit 8	
Byte 3 0xFF	Bit 23 1111-1111 Bit 16	
Byte 4 0x03	Bit 31 0000-0011 Bit 24	

Even though device 12 (Bit 12) has an error as shown in MODE 2 (Device defective) above, the device is still functioning on the network which causes MODE 4 (Device exists) to show the device as an "existing device".

Table 9- 124 Example 3: PROFIBUS slave device "DPSlave_12" is disconnected (cable disconnected or power loss) from the PROFIBUS network. "DPSlave_12" is still detected as a defective device as well as an error in the device. The difference is that "DPSlave_12" is no longer detected as a device that exists. A value of 0x01EC_FF03 is returned for MODE 4 (Device exists).

Byte with value	Bit pattern with value	Notes
Byte 1 0x01	Bit 7 0000-0001 Bit 0	Bit 0 is true; data is available.
Byte 2 0xEC	Bit 15 111**0**-1100 Bit 8	
Byte 3 0xFF	Bit 23 1111-1111 Bit 16	
Byte 4 0x03	Bit 31 0000-0011 Bit 24	

Device 12 (Bit 12) is marked as not existing. With this exception, devices 10 through 25 still report as existing.

PROFINET example

The PROFINET example consists of the following:

- 16 PROFINET slave devices named "et200s_1" through "et200s_16"
- The 16 PROFINET devices use PROFINET device numbers 1 through 16, respectively.
- Each slave device is configured with multiple I/O modules.
- The first four bytes of the returned STATE parameter information is displayed.

MODE	Example 1: Normal operation with no errors	Example 2: PROFINET slave et200s_1 module pulled	Example 3: PROFINET slave et200s_1 disconnect-ed
1: Device configuration active	0xFFFF_0100	0xFFFF_0100	0xFFFF_0100
2 - Device defective	0x0000_0000	0x0300_0000	0x0300_0000
3 - Device disabled	0x0000_0000	0x0000_0000	0x0000_0000
4 - Device exists	0xFFFF_0100	0xFFFF_0100	0xFDFF_0100
5 - Problem in device	0x0000_0000	0x0300_0000	0x0300_0000

The following four tables show a binary breakdown of the four bytes of data that are being analyzed:

Table 9- 125 Example 1: No errors: A value of 0xFFFF_0100 is returned for MODE 1 (Device configuration active).

Byte with value	Bit pattern with value	Notes
Byte 1 0xFF	Bit 7 1111-1111 Bit 0	Bit 0 is true; data is available.
Byte 2 0xFF	Bit 15 1111-1111 Bit 8	
Byte 3 0x01	Bit 23 0000-0001 Bit 16	
Byte 4 0x00	Bit 31 0000-0000 Bit 24	

The devices are configured in addresses 1 (Bit 1) through 16 (Bit 16).
No devices are configured in addresses 1 through 9.
MODE 4 (Device exists) data matches MODE 1 (Device configuration active), so the configured devices match the existing devices.

Table 9- 126 Example 2: A module has been pulled from PROFINET slave device "et200s_1". A value of 0x0300_0000 is returned for MODE 2 (Device defective).

Byte with value	Bit pattern with value	Notes
Byte 1 0x03	Bit 7 0000-0011 Bit 0	Bit 0 is true; data is available.
Byte 2 0x00	Bit 15 0000-0000 Bit 8	
Byte 3 0x00	Bit 23 0000-0000 Bit 16	
Byte 4 0x00	Bit 31 0000-0000 Bit 24	

Device 1 (Bit 1) is marked as defective. Since the device still exists, MODE 4 (Device exists) shows the same data as when operating normally.
MODE 5 (Problem in device) returns the same information as MODE 2 (Device defective).

Table 9- 127 Example 2 (continued): A module has been pulled from PROFIBUS slave device "et200s_1". A value of 0xFFFF_0100 is returned for MODE 4 (Device exists).

Byte with value	Bit pattern with value	Notes
Byte 1 0xFF	Bit 7 1111-1111 Bit 0	Bit 0 is true; data is available.
Byte 2 0xFF	Bit 15 1111-1111 Bit 8	
Byte 3 0x01	Bit 23 0000-0001 Bit 16	
Byte 4 0x00	Bit 31 0000-0000 Bit 24	

Even though device 1 (Bit 1) has an error as shown in MODE 2 (Device defective) above, the device is still functioning on the network which causes MODE 4 (Device exists) to show the device as an "existing device".

Table 9- 128 Example 3: PROFINET slave device "et200s_1" is disconnected (cable disconnected or power loss) from the PROFINET network. A value of 0xFDFF_0100 is returned for MODE 4 (Device exists).

Byte with value	Bit pattern with value	Notes
Byte 1 0xFD	Bit 7 1111-1101 Bit 0	Bit 0 is true; data is available.
Byte 2 0xFF	Bit 15 1111-1111 Bit 8	
Byte 3 0x01	Bit 23 0000-0001 Bit 16	
Byte 4 0x00	Bit 31 0000-0000 Bit 24	

Device 1 (Bit 1) does not exist. Devices 2 (Bit 2) through 16 (Bit 16) do exist.

9.5.5 ModuleStates instruction

You can use the ModuleStates instruction to return the status of all of the modules in a PROFIBUS or PROFINET station.

Table 9- 129 ModuleStates instruction

LAD / FBD	SCL	Description

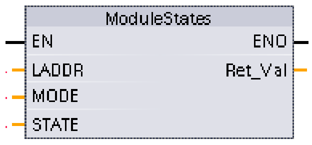
 ModuleStates
EN
ENO
LADDR
Ret_Val
MODE
STATE
	`ret_val := ModuleStates(`
	`    laddr:=_word_in_,`
	`    mode:=_uint_in,`
	`    state:=_variant_inout);`
		ModuleStates retrieves the operational states of I/O modules. After execution, the STATE parameter contains the error state of each I/O module in a bit list (for the assigned LADDR and MODE). This information corresponds with the module status seen in the STEP 7 diagnostics view.
		The LADDR input of ModuleStates uses is a hardware identifier of a distributed I/O station and not of the head module itself. The hardware identifier can be found by selecting the entire station in the network view and then looking in the hardware identifier section under properties. It can also be found by looking for "Hw_Device" and "Hw_DpSlave" data types in the system constants tab in the PLC tag table.

Table 9- 130 Data types for the parameters

Parameter and type		Data type	Description
LADDR	IN	HW_DEVICE	Logical address (Identifier for the I/O modules)
MODE	IN	UInt	Supports five modes of operation. The MODE input determines which data will be returned to the location specified for STATE information. The modes are as follows: • 1: Module configuration active • 2: Module defective • 3: Module disabled • 4: Module exists • 5: Problem in Module
RET_VAL	OUT	Int	Status (condition code)
STATE[1]	InOut	Variant	Buffer that receives the error status of each module: The data type you use for the STATE parameter can be any bit type (Bool, Byte, Word, or DWord) or an array of a bit type. • Bit 0 of the first byte of the returned STATE data is a summary bit. When it is set to TRUE, it indicates that other data is available. • The data returned by the STATE parameter shows a one-to-one correlation between a bit location and a module position. This slot addressing is TRUE for PROFIBUS and PROFINET. For example, for an ET 200S with a head module, power module, and a pair of I/O modules, Bit 1 in the first Byte correlates to the head module, Bit 2 to the power module, and Bits 3 and 4 to the I/O modules, respectively.

[1] A maximum of 128 bits can be assigned. The number of bits required is dependent on your I/O module usage.

Table 9- 131 Condition codes

RET_VAL (W#16#...)	Description
0	No error
8091	Module identified by LADDR does not exist.
8092	Module identified by LADDR does not address an I/O device.
8093	Invalid data type for STATE parameter: Valid data types are (Bool, Byte, Word, or Dword), or an array of (Bools, Bytes, Words, or Dwords).
80Bx	ModuleStates instruction not supported by this CPU for this LADDR.
8452	The complete state data is too large for the assigned STATE parameter. The STATE buffer contains a partial result.

9.5.5.1 ModuleStates example configurations

PROFIBUS example

The PROFIBUS example consists of the following:

- 16 PROFIBUS devices named "DPSlave_10" through "DPSlave_25"
- The 16 PROFIBUS devices use PROFIBUS addresses 10 through 25, respectively.
- Each slave device is configured with multiple I/O modules.
- The example uses the LADDR parameter of PROFIBUS slave "DPSlave_12" which contains a head module, a power module, and two I/O modules.
- The first four bytes of the returned STATE parameter information is displayed.

MODE	Example 1: Normal operation with no errors	Example 2: PROFIBUS slave device DPSlave_12 module pulled	Example 3: PROFIBUS slave device DPSlave_12 disconnected
1: Module configuration active	0x1F00_0000	0x1F00_0000	0x1F00_0000
2: Module defective	0x0000_0000	0x0900_0000	0x1F00_0000
3: Module disabled	0x0000_0000	0x0000_0000	0x0000_0000
4: Module exists	0x1F00_0000	0x1700_0000	0x0000_0000
5: Problem in module	0x0000_0000	0x0900_0000	0x1F00_0000

The following four tables show a binary breakdown of the four bytes of data that are being analyzed:

Table 9- 132 Example 1: No errors: A value of 0x1F00_0000 is returned for MODE 1 (Module configuration active).

Byte with value	Bit pattern with value	Notes
Byte 1 0x1F	Bit 7 0001-1111 Bit 0	Bit 0 is true; data is available.
Byte 2 0x00	Bit 15 0000-0000 Bit 8	
Byte 3 0x00	Bit 23 0000-0000 Bit 16	
Byte 4 0x00	Bit 31 0000-0000 Bit 24	

Slots 1 (Bit 1) through 4 (Bit 4) contain modules. Slots 5 (Bit 5) and beyond do not contain modules. MODE 4 (Module exists) data matches MODE 1 (Module configuration active), so the configured modules match the existing modules.

Table 9- 133 Example 2: A module has been pulled from PROFIBUS slave device "DPSlave_12". A value of 0x0900_0000 is returned for MODE 2 (Module defective).

Byte with value	Bit pattern with value	Notes
Byte 1 0x09	Bit 7 0000-1001 Bit 0	Bit 0 is true; data is available.
Byte 2 0x00	Bit 15 0000-0000 Bit 8	
Byte 3 0x00	Bit 23 0000-0000 Bit 16	
Byte 4 0x00	Bit 31 0000-0000 Bit 24	

Only module 3 (Bit 3) is marked as defective. All other modules are functional.

Table 9- 134 Example 2 (continued): A module has been pulled from PROFIBUS slave device "DPSlave_12". A value of 0x1700_0000 is returned for MODE 4 (Module exists).

Byte with value	Bit pattern with value	Notes
Byte 1 0x17	Bit 7 0001-0111 Bit 0	Bit 0 is true; data is available.
Byte 2 0x00	Bit 15 0000-0000 Bit 8	
Byte 3 0x00	Bit 23 0000-0000 Bit 16	
Byte 4 0x00	Bit 31 0000-0000 Bit 24	

Module 3 (Bit 3) is shown as missing. Modules 1, 2, and 4 (Bits 1, 2, and 4) are shown as existing.

Table 9- 135 Example 3: PROFIBUS slave device "DPSlave_12" is disconnected (cable disconnected or power loss) from the PROFIBUS network. A value of 0x1F00_0000 is returned for MODE 2 (Module defective).

Byte with value	Bit pattern with value	Notes
Byte 1 0x1F	Bit 7 0001-1111 Bit 0	Bit 0 is true; data is available.
Byte 2 0x00	Bit 15 0000-0000 Bit 8	
Byte 3 0x00	Bit 23 0000-0000 Bit 16	
Byte 4 0x00	Bit 31 0000-0000 Bit 24	

The modules in slots 1 through 4 (Bits 1 through 4) are all marked as defective since the device is missing.
MODE 5 (Problem in module) shows the same information as MODE 2 (Module defective).

PROFINET example

The PROFINET example consists of the following:

- 16 PROFINET slave devices named "et200s_1" through "et200s_16"
- The 16 PROFINET devices use PROFINET device numbers 1 through 16, respectively.
- Each slave device is configured with multiple I/O modules.
- The example uses PROFINET slave "et200s_1" which contains a head module, a power module, and 18 I/O modules.
- The first four bytes of the returned STATE parameter information is displayed.

MODE	Example 1: Normal operation with no errors	Example 2: PROFINET et200s_1 slave module pulled	Example 3: PROFINET et200s_1 slave disconnected
1: Module configuration active	0xFFFF_1F00	0xFFFF_1F00	0xFFFF_1F00
2: Module defective	0x0000_0000	0x0180_0000	0xFFFF_1F00
3: Module disabled	0x0000_0000	0x0000_0000	0x0000_0000
4: Module exists	0xFFFF_1F00	0xFF7F_1F00	0x0000_0000
5: Problem in module	0x0000_0000	0x0180_0000	0xFFFF_1F00

The following four tables show a binary breakdown of the four bytes of data that are being analyzed:

Table 9- 136 Example 1: No errors: A value of 0xFFFF_1F00 is returned for MODE 1 (Module configuration active).

Byte with value	Bit pattern with value	Notes
Byte 1 0xFF	Bit 7 1111-1111 Bit 0	Bit 0 is true; data is available.
Byte 2 0xFF	Bit 15 1111-1111 Bit 8	
Byte 3 0x1F	Bit 23 0001-1111 Bit 16	
Byte 4 0x00	Bit 31 0000-0000 Bit 24	

Slots 1 (Bit 1) through 20 (Bit 20) contain modules. Slot 21 (Bit 21) and beyond do not contain modules.
MODE 4 (Module exists) data matches MODE 1 (Module configuration active), so the configured modules match the existing modules.

Table 9- 137 Example 2: A module has been pulled from PROFINET slave device "et200s_1". A value of 0x0180_0000 is returned for MODE 2 (Module defective).

Byte with value	Bit pattern with value	Notes
Byte 1 0x01	Bit 7 0000-0001 Bit 0	Bit 0 is true; data is available.
Byte 2 0x80	Bit 15 **1**000-0000 Bit 8	
Byte 3 0x00	Bit 23 0000-0000 Bit 16	
Byte 4 0x00	Bit 31 0000-0000 Bit 24	

Only module 15 (Bit 15) is marked as defective. All other modules are functional.

Table 9- 138 Example 2 (continued): A module has been pulled from PROFIBUS slave device "et200s_1". A value of 0xFF7F_1F00 is returned for MODE 4 (Module exists).

Byte with value	Bit pattern with value	Notes
Byte 1 0xFF	Bit 7 1111-1111 Bit 0	Bit 0 is true; data is available.
Byte 2 0x7F	Bit 15 **0**111-1111 Bit 8	
Byte 3 0x1F	Bit 23 0001-1111 Bit 16	
Byte 4 0x00	Bit 31 0000-0000 Bit 24	

Module 15 (Bit 15) is shown as missing. Modules 1 through 14 (Bits 1 through 14) and 16 through 20 (Bits 16 through 20) are shown as existing.

Table 9- 139 Example 3: PROFINET slave device "et200s_1" is disconnected (cable disconnected or power loss) from the PROFINET network. A value of 0xFFFF_1F00 is returned for MODE 2 (Module defective).

Byte with value	Bit pattern with value	Notes
Byte 1 0xFF	Bit 7 **1111-1111** Bit 0	Bit 0 is true; data is available.
Byte 2 0xFF	Bit 15 **1111-1111** Bit 8	
Byte 3 0x1F	Bit 23 000**1-1111** Bit 16	
Byte 4 0x00	Bit 31 0000-0000 Bit 24	

The modules in slots 1 through 20 (Bits 1 through 20) are all marked as defective since the device is missing.
MODE 5 (Problem in module) shows the same information as MODE 2 (Module defective).

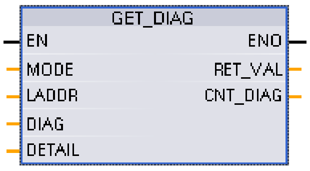
9.5.6 GET_DIAG (Read diagnostic information) instruction

Description

You can use the "GET_DIAG" instruction to read out the diagnostic information of a hardware device. The hardware device is selected with the LADDR parameter. With the MODE parameter, you select which diagnostic information to read.

Table 9- 140 GET_DIAG instruction

LAD / FBD	SCL	Description

 GET_DIAG
EN
ENO
MODE
RET_VAL
LADDR
CNT_DIAG
DIAG
DETAIL
	`ret_val := GET_DIAG(`
	`mode:=_uint_in_,`
	`laddr:=_word_in_,`
	`cnt_diag=>_uint_out_,`
	`diag:=_variant_inout_,`
	`detail:=_variant_inout_);`
		Reads the diagnostic information from an assigned hardware device.

Parameters

The following table shows the parameters of the "GET_DIAG" instruction:

Table 9- 141 Data types for the parameters

Parameter and type		Data type	Description
MODE	IN	UInt	Use the MODE parameter to select which diagnostic data is to be output.
LADDR	IN	HW_ANY (Word)	Hardware ID of the device
RET_VAL	OUT	Int	Status of the instruction
CNT_DIAG	OUT	UInt	Number of output diagnostic details
DIAG	InOut	Variant	Pointer to data area for storage of diagnostic information of the se-lected mode
DETAILS	InOut	Variant	Pointer to data area for storage of diagnostic details in accordance with the selected mode

MODE parameter

Depending on the value at the MODE parameter, different diagnostics data is output at the DIAG, CNT_DIAG and DETAILS output parameters:

Table 9- 142 MODE parameter

MODE	Description	DIAG	CNT_DIAG	DETAILS
0	Output of all supported diagnostic information for a module as DWord, where Bit X=1 indicates that mode X is supported.	Bit string of the supported modes as DWord, where Bit X=1 indicates that mode X is supported.	0	-
1	Output of the inherent status of the addressed hardware object.	Diagnostics status: Output in accordance with the DIS structure. (Note: Refer to the "DIS structure" information below and GET_DIAG instruction example at the end of the section.)	0	-
2	Output of the status of all subordinate modules of the addressed hardware object.	Output of diagnostics data in accordance with the DNN structure. (Note: Refer to the "DNN structure" information below and GET_DIAG instruction example at the end of the section.)	0	-

DIS structure

With the MODE parameter = 1, the diagnostics information is output in accordance with the DIS structure. The following table shows the meaning of the individual parameter values:

Table 9- 143 Structure of the Diagnostic Information Source (DIS)

Parameter	Data type	Value	Description
MaintenanceState	DWord	Enum	
		0	No maintenance required
		1	The module or device is disabled.
		2	-
		3	-
		4	-
		5	Maintenance required
		6	Maintenance demanded
		7	Error
		8	Status unknown / error in subordinate module
		9	-
		10	Inputs/outputs are not available.

Parameter	Data type	Value	Description
Componentstate Detail	DWord	Bit array	Status of the module submodules: • Bit 0 to 15: Status message of the module • Bit 16 to 31: Status message of the CPU
		0 to 2 (enum)	Additional information: • Bit 0: No additional information • Bit 1: Transfer not permitted
		3	Bit 3 = 1: At least one channel supports qualifiers for diagnostics.
		4	Bit 4 = 1: Maintenance required for at least one channel or one component
		5	Bit 5 = 1: Maintenance demanded for at least one channel or one component
		6	Bit 6 = 1: Error in at least one channel or one component
		7 to 10	Reserved (always = 0)
		11 to 14	Bit 11 = 1: PNIO - submodule correct Bit 12 = 1: PNIO - replacement module Bit 13 = 1: PNIO - incorrect module Bit 14 = 1: PNIO - module disconnected
		15	Reserved (always = 0)
		16 to 31	Status information for modules generated by the CPU: Bit 16 = 1: Module disabled Bit 17 = 1: CiR operation active Bit 18 = 1: Input not available Bit 19 = 1: Output not available Bit 20 = 1: Overflow diagnostics buffer Bit 21 = 1: Diagnostics not available Bit 22 - 31: Reserved (always 0)
OwnState	Uint16	Enum	The value of the OwnState parameter describes the maintenance status of the module.
		0	No fault
		1	The module or device is disabled.
		2	Maintenance required
		3	Maintenance demanded
		4	Error
		5	The module or the device cannot be reached from the CPU (valid for modules and devices below a CPU).
		6	Inputs/outputs are not available.
		7	-
IO State	Uint16	Bit array	I/O status of the module
		0	Bit 0 = 1: No maintenance required
		1	Bit 1 = 1: The module or device is disabled.
		2	Bit 2 = 1: Maintenance required
		3	Bit 3 = 1: Maintenance demanded
		4	Bit 4 = 1: Error

Parameter	Data type	Value	Description
		5	Bit 5 = 1: The module or the device cannot be reached from the CPU (valid for modules and devices below a CPU).
		6	Inputs/outputs are not available.
		7	Qualifier; bit 7 = 1, if bit 0, 2, or 3 are set
		8 to 15	Reserved (always = 0)
OperatingState	UInt16	Enum	
		0	-
		1	In STOP / firmware update
		2	In STOP / reset memory
		3	In STOP / self start
		4	In STOP
		5	Memory reset
		6	In START
		7	In RUN
		8	-
		9	In HOLD
		10	-
		11	-
		12	Module defective
		13	-
		14	No power
		15	CiR
		16	In STOP / without DIS
		17	In
		18	
		19	
		20	

DNN structure

With the MODE parameter = 2, the diagnostics information details are output in accordance with the DNN structure. The following table shows the meaning of the individual parameter values:

Table 9- 144 Structure of the Diagnostic Navigation Node (DNN)

Parameter	Data type	Value	Description
SubordinateState	UINT	Enum	Status of the subordinate module (See parameter OwnState of the DIS structure.)
SubordinateIOState	WORD	Bitarray	Status of the inputs and outputs of the subordinate module (See parameter IO State of the DIS structure.)
DNNmode	WORD	Bitarray	• Bit 0 = 0: Diagnostics enabled • Bit 0 = 1: Diagnostics disabled • Bit 1 to 15: Reserved

RET_VAL parameter

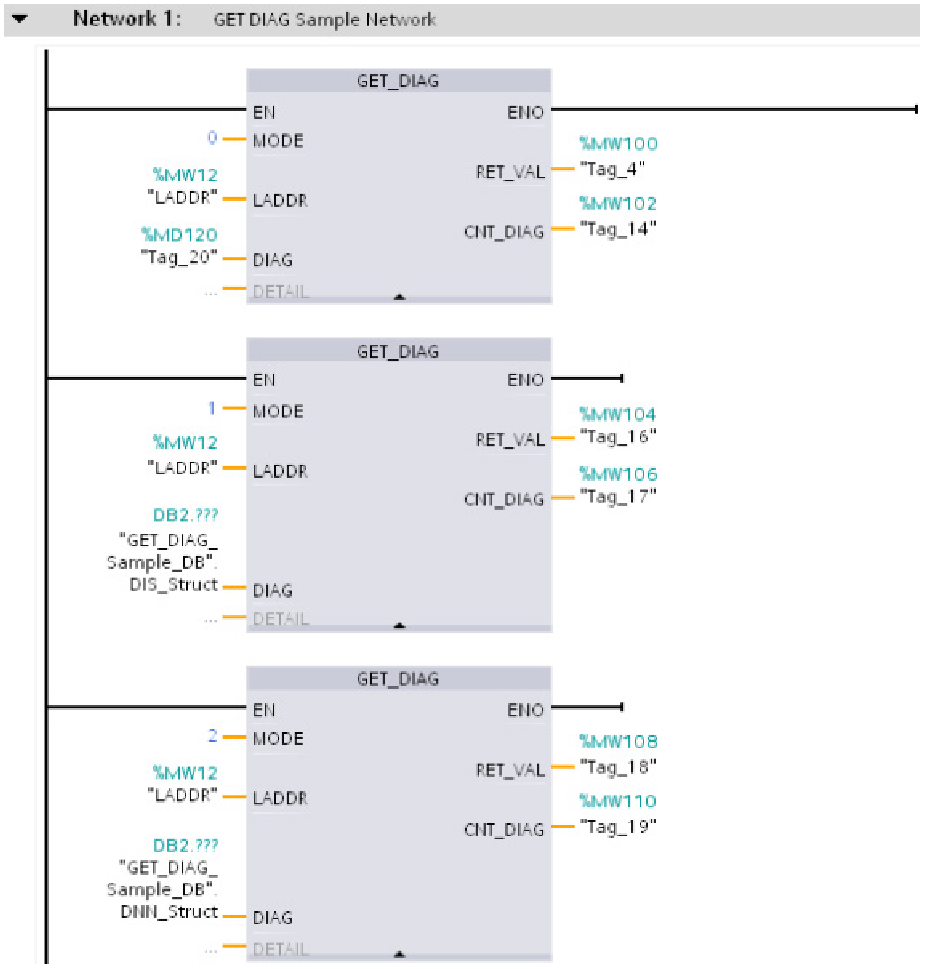
Table 9- 145 Error codes of the RET_VAL parameter

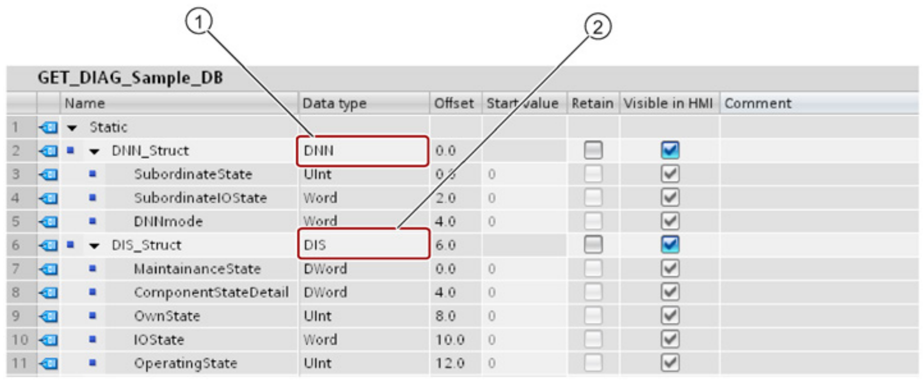
Error code (W#16#...)	Description
0	No error
8080	Value in the MODE parameter is not supported.
8081	Type in the DIAG parameter is not supported with the selected mode (parameter MODE).
8082	Type in the DETAILS parameter is not supported with the selected mode (parameter MODE).
8090	LADDR does not exist.
8091	The selected channel in the CHANNEL parameter does not exist.
80C1	Insufficient resources for parallel execution

Example

The following ladder logic network and DB show how to use the three modes with the three structures:

- DIS
- DNN





GET_DIAG_Sample_DB

	Name	Data type	Offset	Start value	Retain	Visible in HMI	Comment
1	▼ Static						
2	▼ DNN_Struct	DNN	0.0		☐	☑	
3	SubordinateState	UInt	0.0	0	☐	☑	
4	SubordinateIOState	Word	2.0	0	☐	☑	
5	DNNmode	Word	4.0	0	☐	☑	
6	▼ DIS_Struct	DIS	6.0		☐	☑	
7	MaintainanceState	DWord	0.0	0	☐	☑	
8	ComponentStateDetail	DWord	4.0	0	☐	☑	
9	OwnState	UInt	8.0	0	☐	☑	
10	IOState	Word	10.0	0	☐	☑	
11	OperatingState	UInt	12.0	0	☐	☑	

①	DNN
②	DIS

Note

In the DB, you must manually type in the data type to access each of the three structures; there is no dropdown list selection. Type in the data types exactly as shown below:

- DNN
- DIS

9.5.7 Get_IM_Data (Read the identification and maintenance data) instruction

You use the Get_IM_Data instruction to check the identification and maintenance (I&M) data for the specified module or sub-module.

Table 9- 146 Get_IM_Data instruction

LAD / FBD	SCL	Description

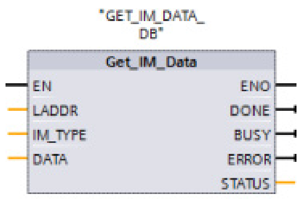
 "GET_IM_DATA_DB"
Get_IM_Data
EN
ENO
LADDR
DONE
IM_TYPE
BUSY
DATA
ERROR
STATUS
	`"GET_IM_DATA_DB"(LADDR:=16#0`
	`,`
	`IM_TYPE:=0,`
	`DONE=>_bool_out_,`
	`BUSY=>_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_,`
	
	`DATA:=_variant_inout_);`
		Use the Get_IM_Data instruction to check the identification and maintenance (I&M) data for the specified module or sub-module.

Table 9- 147 Data types for the parameters

Parameter and type		Data type	Description
LADDR	Input	HW_IO	Identifier of the module
IM_TYPE	Input	UInt	Identification and maintenance (I&M) data number: • 0: I&M0 (MLFB, serial number, version, and other information) • 1: I&M1 (Designators) • 2: I&M2 (Installation date) • 3: I&M3 (Descriptor) • 4: I&M4 (Signature)
RET_VAL	Output	Int	Status (condition code)
DATA	InOut	Variant	I&M data (STRING or an array of BYTE)

Identification and maintenance (I&M) data can help you to check the system configuration, detect hardware changes, or view maintenance data. Module identification data (I data) is read only. Module maintenance data (M data) depends on system information, such as the installation date. M data are created during maintenance planning and written to the module:

- If the data type used at the parameter DATA is a string, then the current length of the string is set according to the length of the I&M data.
- If the data type used at the parameter DATA is an array of Byte or Char, then the I&M data are copied in as a sequence of bytes.
- If the data type used at the parameter DATA is a structure, then the I&M data are copied in as a sequence of bytes.
- If the given array of byte/char at DATA is longer than the requested I&M data, then the byte value 16#00 is appended.
- Other data types are not supported and error 8093 is returned.

Table 9- 148 Condition codes

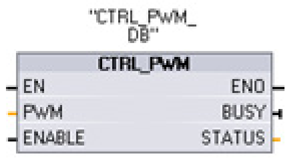
RET_VAL (W#16#...)	Description
0	No error
8091	LADDR does not exist
8092	LADDR does not address a HW object which supports I&M data
8093	Data type given at parameter DATA is not supported
80B1	DATA instruction not supported by the CPU for this LADDR
80B2	IM_TYPE not supported by the CPU
8452	The complete I&M information does not fit into the variable given at the DATA parameter. A partial result up to the byte length of the variable is returned.

9.6 Pulse

9.6.1 CTRL_PWM (Pulse width modulation) instruction

Table 9- 149 CTRL_PWM (Pulse Width Modulation) instruction

LAD / FBD	SCL	Description

 "CTRL_PWM_DB"
CTRL_PWM
EN
ENO
PWM
BUSY
ENABLE
STATUS
	`"CTRL_PWM_DB"(`
	`PWM:=W#16#0,`
	`enable:=FALSE,`
	`busy=>_bool_out_,`
	`status=>_word_out_);`
		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.

1 When you insert the instruction, STEP 7 displays the "Call Options" dialog for creating the associated DB.

2 In the SCL example, "CTRL_PWM_DB" is the name of the instance DB.

Table 9- 150 Data types for the parameters

Parameter and type		Data type	Description
PWM	IN	HW_PWM (Word)	PWM identifier: Names of enabled pulse generators will become tags in the "constant" tag table, and will be available for use as the PWM parameter. (Default value: 0)
ENABLE	IN	Bool	1=start pulse generator 0 = stop pulse generator
BUSY	OUT	Bool	Function busy (Default value: 0)
STATUS	OUT	Word	Execution condition code (Default value: 0)

The CTRL_PWM instruction stores the parameter information in the DB. The data block parameters are not separately changed by the user, but are controlled by the CTRL_PWM instruction.

Specify the enabled pulse generator to use, by using its tag name for the PWM parameter.

When the EN input is TRUE, the PWM_CTRL instruction starts or stops the identified PWM based on the value at the ENABLE input. Pulse width is specified by the value in the associated Q word output address.

Because the CPU processes the request when the CTRL_PWM instruction is executed, parameter BUSY will always report FALSE. If an error is detected, then ENO is set to FALSE, and parameter STATUS contains a condition code.

The pulse width will be set to the initial value configured in device configuration when the CPU first enters RUN mode. You write values to the Q-word location specified in device configuration ("Output addresses" / "Start address:") as needed to change the pulse width. You use an instruction such as a move, convert, math, or PID box to write the desired pulse width to the appropriate Q word. You must use the valid range for the Q-word value (percent, thousandths, ten-thousandths, or S7 analog format).

Note

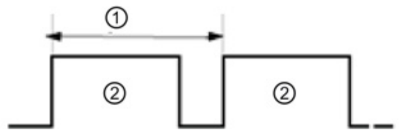
Digital I/O points assigned to PWM and PTO cannot be forced

The digital I/O points used by the pulse-width modulation (PWM) and pulse-train output (PTO) devices are assigned during device configuration. When digital I/O point addresses are assigned to these devices, the values of the assigned I/O point addresses cannot be modified by the Watch table force function.

Table 9- 151 Value of the STATUS parameter

STATUS	Description
0	No error
80A1	PWM identifier does not address a valid PWM.

9.6.2 Operation of the pulse outputs



① Cycle time

② Pulse width

Pulse width can be expressed as hundredths of the cycle time (0 to 100), as thousandths (0 to 1000), as ten thousandths (0 to 10000), or as S7 analog format.

The pulse width can vary from 0 (no pulse, always off) to full scale (no pulse, always on).

Since the PWM output can be varied from 0 to full scale, it provides a digital output that in many ways is the same as 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 position of a valve from closed to fully opened.

Four pulse generators are available for controlling high-speed pulse output functions: PWM and Pulse train output (PTO). PTO is used by the motion control instructions. You can assign each pulse generator to either PWM or PTO, but not both at the same time.

You can use onboard CPU outputs, or you can use the optional signal board outputs. The output point numbers are shown in the following table (assuming the default output configuration). If you have changed the output point numbering, then the output point numbers will be those you assigned. Note that PWM requires only one output, while PTO can optionally use two outputs per channel. If an output is not required for a pulse function, it is available for other uses. Refer to the table below for I/O assignment.

The table below shows the default I/O assignments; however, the four pulse generators can be configured to any CPU built-in or SB digital output. Different output points support different voltages and speeds, so take that into account when assigning PWM/PTO locations.

Note

Pulse-train outputs cannot be used by other instructions in the user program.

When you configure the outputs of the CPU or signal board as pulse generators (for use with the PWM or motion control PTO instructions), the corresponding outputs addresses are removed from the Q memory and cannot be used for other purposes in your user program. If your user program writes a value to an output used as a pulse generator, the CPU does not write that value to the physical output.

Note

PTO direction outputs can be freed for use elsewhere in your program.

Each PTO requires the assignment of two outputs: one as a pulse output and one as a direction output. You can use just the pulse output and not the direction output. You can then free the direction output for other purposes in your user program.

Table 9- 152 Default output assignments for the pulse generators[3]

Description	Pulse	Direction
PTO1		
Built-in I/O	Q0.0	Q0.1
SB I/O	Q4.0	Q4.1
PWM1		
Built-in outputs	Q0.0	-
SB outputs	Q4.0	-
PTO2		
Built-in I/O	Q0.2	Q0.3
SB I/O	Q4.2	Q4.3
PWM2		
Built-in outputs	Q0.2	-
SB outputs	Q4.2	-
PTO3		
Built-in I/O	Q0.4[1]	Q0.5[1]
SB I/O	Q4.0	Q4.1
PWM3		
Built-in outputs	Q0.4[1]	-
SB outputs	Q4.1	-
PTO4		
Built-in I/O	Q0.6[2]	Q0.7[2]
SB I/O	Q4.2	Q4.3

Description	Pulse	Direction
PWM4		
Built-in outputs	Q0.6[2]	-
SB outputs	Q4.3	-

[1] The CPU 1211C does not have outputs Q0.4, Q0.5, Q0.6, or Q0.7. Therefore, these outputs cannot be used in the CPU 1211C.

[2] The CPU 1212C does not have outputs Q0.6 or Q0.7. Therefore, these outputs cannot be used in the CPU 1212C.

[3] This table applies to the CPU 1211C, CPU 1212C, CPU 1214C, CPU 1215C, and CPU 1217C PTO/PWM functions.

9.6.3 Configuring a pulse channel for PWM

To prepare for PWM operation, first configure a pulse channel in the device configuration by selecting the CPU, then Pulse Generator (PTO/PWM), and choose PWM1 through PWM4. Enable the pulse generator (check box). If a pulse generator is enabled, a unique default name is assigned to this particular pulse generator. You can change this name by editing it in the "Name:" edit box, but it must be a unique name. Names of enabled pulse generators will become tags in the "constant" tag table, and will be available for use as the PWM parameter of the CTRL_PWM instruction.

Table 9- 153 CPU output: maximum frequency

CPU	CPU output channel	Pulse and direction output	A/B, quadrature, up/down, and pulse/direction
1211C	Qa.0 to Qa.3	100 kHz	100 kHz
1212C	Qa.0 to Qa.3	100 kHz	100 kHz
	Qa.4, Qa.5	20 kHz	20 kHz
1214C and 1215C	Qa.0 to Qa.4	100kHz	100kHz
	Qa.5 to Qb.1	20 kHz	20 kHz
1217C	DQa.0 to DQa.3 (.0+, .0- to .3+, .3-)	1 MHz	1 MHz
	DQa.4 to DQb.1	100 kHz	100 kHz

Table 9- 154 SB signal board output: maximum frequency (optional board)

SB signal board	SB output channel	Pulse and direction output	A/B, quadrature, up/down, and pulse/direction
SB 1222, 200 kHz	DQe.0 to DQe.3	200kHz	200 kHz
SB 1223, 200 kHz	DQe.0, DQe.1	200kHz	200 kHz
SB 1223	DQe.0, DQe.1	20 kHz	20 kHz

Note

The maximum pulse frequency of the pulse output generators is 1 MHz for the CPU 1217C and 100 kHz for CPUs 1211C, 1212C, 1214C, and 1215C; 20 kHz (for a standard SB); or 200 kHz (for a high-speed SB). However, STEP 7 does not alert you when you configure an axis with a maximum speed or frequency that exceeds this hardware limitation. Problems can result with your application, so always ensure that you do not exceed the maximum pulse frequency of the hardware.

You have the option to rename the pulse generator, add a comment, and assign parameters as follows:

- Pulse generator used as follows: PWM or PTO (choose PWM)
- Output source: onboard CPU or SB
- Time base: milliseconds or microseconds
- Pulse width format:
  - Hundredths (0 to 100)
  - Thousandths (0 to 1000)
  - Ten-thousandths (0 to 10000)
  - S7 analog format (0 to 27648)
- Cycle time (range is 0 to 16,777,215): Enter your cycle time value. This value can only be changed in Device configuration.
- Initial pulse width: Enter your initial pulse width value. The pulse width value can be changed during runtime.

Enter the start address to configure the output addresses. Enter the Q word address where you want to locate the pulse width value.

Note

Pulse-train outputs cannot be used by other instructions in the user program

When you configure the outputs of the CPU or signal board as pulse generators (for use with the PWM or motion control instructions), the corresponding outputs addresses are removed from the Q memory and cannot be used for other purposes in your user program. If your user program writes a value to an output used as a pulse generator, the CPU does not write that value to the physical output.

The default locations for the pulse width values are as follows:

- PWM1: QW1000
- PWM2: QW1002
- PWM3: QW1004
- PWM4: QW1006

The value at this location controls the width of the pulse and is initialized to the "Initial pulse width:" value specified above each time the CPU transitions from STOP to RUN mode. You change this Q-word value during run time to cause a change in the pulse width.

9.7 Recipes and Data logs

9.7.1 Recipes

9.7.1.1 Recipe overview

Recipe data storage

- A recipe data block that you create in your project must be stored in CPU **load** memory. Internal CPU memory or an external memory "Program" card can be used.
- Another DB that you must create is the active recipe data block. This DB must be in **work** memory, where one active recipe record is read or written with your program logic.

Recipe data management

The recipe DB uses an array of product recipe records. Each element of the recipe array represents a different recipe flavor that is based on a common set of components.

- You create a PLC data type or struct that defines all the components in one recipe record. This data type template is reused for all recipe records. Product recipes vary according to the start values that are assigned to the recipe components.
- One of the recipes can be transferred at any time from the recipe DB (all recipes in load memory) to the active recipe DB (one recipe in work memory) using the READ_DBL instruction. After a recipe record is moved to work memory, then your program logic can read the component values and begin a production run. This transfer minimizes the amount of CPU work memory that is required for recipe data.
- If the active recipe component values are adjusted by an HMI device during a production run, you can write the modified values back to the recipe DB, using the WRIT_DBL instruction.

Recipe export (from recipe DB to CSV file)

The complete set of recipe records can be generated as a CSV file using the RecipeExport instruction. Unused recipe records are also exported.

Recipe import (from CSV file to recipe DB)

Once a recipe export operation is completed, then you can use the generated CSV file as a data structure template.

1. Use the file browser page in the CPU web server to download an existing recipe CSV file from the CPU to a PC
2. Modify the recipe CSV with an ASCII text editor. You can modify the start values assigned to components, but not the data types or data structure
3. Upload the modified CSV file from PC back to the CPU. However, the old CSV file in CPU load memory (with the same name) must be deleted or renamed before the CPU Web server allows the upload operation.
4. After the modified CSV file is uploaded to the CPU, then you can use the RecipeImport instruction to transfer the new start values from the modified CSV file (in CPU load memory) to the recipe DB (in CPU load memory).

9.7.1.2 Recipe example

Example recipes

The table below shows how to prepare recipe information for use in a recipe DB. The example recipe DB stores five records, three of which are used. The fourth and fifth records are free for later expansions. Each table row represents one record that stores the recipe name, component data types, and component values.

productname	water	barley	wheat	hops	yeast	waterTmp	mashTmp	mashTime	QTest
Pils	10	9	3	280	39	40	30	100	0
Lager	10	9	3	150	33	50	30	120	0
BlackBeer	10	9	3	410	47	60	30	90	1
Not_used	0	0	0	0	0	0	0	0	0
Not_used	0	0	0	0	0	0	0	0	0

Creating a recipe data block

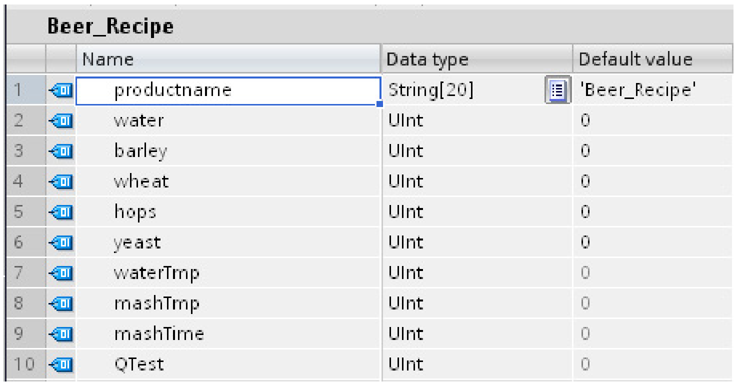
Note

Rules for recipe data blocks

- The recipe DB must contain a single dimension array of either a PLC data type or a struct. The recipe example shows how to create a recipe DB with a PLC data type.
- In the example, the data type of the component ingredients are all the UINT data type. The component data types may also be a mix of any data type except for structs. In a recipe DB array element, a struct in a PLC data type or a struct in a struct is not allowed.

First, create a new PLC data type

Add a new PLC data type whose name is the recipe type. In the following image, "Beer_Recipe" is the new complex PLC data type that stores a sequence of simple data types. The "Beer_Recipe" PLC data type is a data template that is reused in each recipe DB record and also in the active recipe DB. Enter the component names and data types that are common to all the example recipes. The individual component values are added later in the recipe DB.



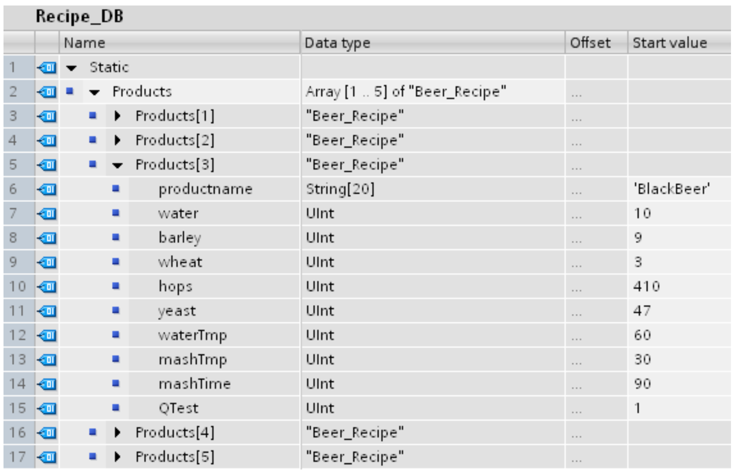
Beer_Recipe

	Name	Data type	Default value
1	productname	String[20]	'Beer_Recipe'
2	water	UInt	0
3	barley	UInt	0
4	wheat	UInt	0
5	hops	UInt	0
6	yeast	UInt	0
7	waterTmp	UInt	0
8	mashTmp	UInt	0
9	mashTime	UInt	0
10	QTest	UInt	0

Second, create a recipe data block

- Create your recipe DB as a global data block with the DB property "Only store in load memory" enabled.
- The name of a recipe data block is used as file name of the corresponding CSV file. The DB name characters you assign must follow the Windows file system naming restrictions. Characters \ / : * ? " < > | and the space character are not allowed.
- The recipe array assignment is "Products" as Array [1.. 5] of "Beer_Recipe". The array size 5 is the maximum number of recipe flavors that are possible.
- The values for recipe components are added as DB start values.

In the following image, the "BlackBeer" recipe is expanded to show all the components of a recipe record.



Recipe_DB

	Name	Data type	Offset	Start value
1	Static			
2	Products	Array [1 .. 5] of "Beer_Recipe"	...	
3	Products[1]	"Beer_Recipe"	...	
4	Products[2]	"Beer_Recipe"	...	
5	Products[3]	"Beer_Recipe"	...	
6	productname	String[20]	...	'BlackBeer'
7	water	UInt	...	10
8	barley	UInt	...	9
9	wheat	UInt	...	3
10	hops	UInt	...	410
11	yeast	UInt	...	47
12	waterTmp	UInt	...	60
13	mashTmp	UInt	...	30
14	mashTime	UInt	...	90
15	QTest	UInt	...	1
16	Products[4]	"Beer_Recipe"	...	
17	Products[5]	"Beer_Recipe"	...	

Recipe export (from recipe DB to CSV file)

"RecipeExport (Page 415)" execution transfers recipe DB data to a CSV file, as shown in the following text file.

```
Recipe_DB.csv
index,productname,water,barley,wheat,hops,yeast,waterTmp,
mashTmp,mashTime,QTest
1,”Pils”,10,9,3,280,39,40,30,100,0
2,”Lager”,10,9,3,150,33,50,30,120,0
3,”BlackBeer”,10,9,3,410,47,60,30,90,1
4 ”Not_used”,0,0,0,0,0,0,0,0,0
5 ”Not_used”,0,0,0,0,0,0,0,0,0
```

Recipe import (from CSV file to recipe DB)

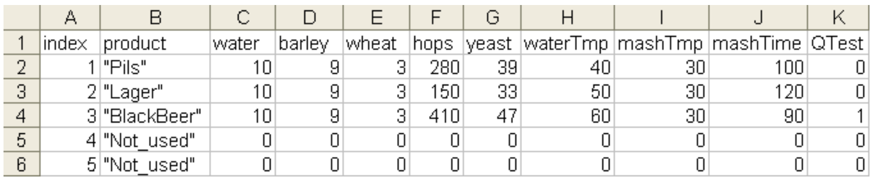
1. Use the file browser page in the CPU web server to download an existing recipe CSV file from CPU load memory to a PC
2. Modify the recipe CSV with an ASCII text editor. You can modify the start values assigned to components, but not the data types or data structure
3. Upload the modified CSV file from PC back to the CPU. However, the old CSV file in CPU load memory (with the same name) must be deleted or renamed before the CPU Web server allows the upload operation.
4. After the modified CSV file is uploaded to the CPU, then you can use the RecipeImport instruction to transfer the new start values from the modified CSV file (in CPU load memory) to the recipe DB (in CPU load memory).

CSV files must exactly match the corresponding recipe DB structure

- The values in the CSV file can be changed, but changing the structure is not allowed. The RecipeImport instruction requires that the exact number of records and components matches the destination recipe DB structure. Otherwise, RecipeImport execution fails. For example, if 10 recipes are defined in the recipe DB but only 6 are in use, then line 7 to 10 in the CSV file are also transferred to the DB. You must coordinate whether this data is valid or not. For example, you can assign a variable “Not_used” for the product name in unused recipe records.
- If you add data records to the text file and import the modified file, make sure the recipe DB array limit you assign has enough elements for all the recipe records.
- An index number is automatically generated during export to the CSV file. If you create additional data records, add consecutive index numbers accordingly.
- RecipeImport execution checks the CSV file data for correct structure and whether the values fit in the data types assigned in the associated recipe DB. For example, a Bool data type cannot store an integer value and the RecipeImport execution fails.

Display CSV recipe data in Excel

The CSV file can be opened in Excel to make reading and editing easier. If the commas are not recognized as decimal separators, use the Excel import function to output the data in structured form



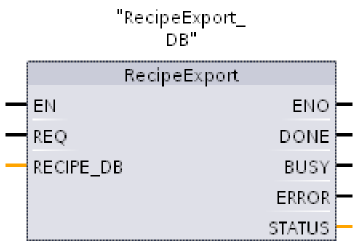
	A	B	C	D	E	F	G	H	I	J	K
1	index	product	water	barley	wheat	hops	yeast	waterTmp	mashTmp	mashTime	QTest
2	1	"Pils"	10	9	3	280	39	40	30	100	0
3	2	"Lager"	10	9	3	150	33	50	30	120	0
4	3	"BlackBeer"	10	9	3	410	47	60	30	90	1
5	4	"Not_used"	0	0	0	0	0	0	0	0	0
6	5	"Not_used"	0	0	0	0	0	0	0	0	0

9.7.1.3 Program instructions that transfer recipe data

RecipeExport (Recipe export) instruction

Table 9- 155 RecipeExport instruction

LAD/FBD	SCL	Description

 "RecipeExport_ DB"
RecipeExport
EN
REQ
RECIPE_DB
ENO
DONE
BUSY
ERROR
STATUS
	`"RecipeExport_DB"(`
	`    req:=_bool_in_,`
	`    done=>_bool_out_,`
	`    busy=>_bool_out_,`
	`    error=>_bool_out_,`
	`    status=>_word_out_,`
	`    Reci-`
	`pe_DB:=_variant_inout_);`
		The "RecipeExport" instruction exports all recipe records from a recipe data block to the CSV file format. The CSV file contains product names, component names, and start values. The CSV file is stored in internal load memory or external load memory, if an optional external "program" memory card is installed.
		The export operation is triggered by the "REQ" parameter. The BUSY parameter is set to "1" during export processing. After the execution of RecipeExport stops, BUSY is reset to "0" and the completion of the operation is indicated with "1" at the DONE parameter. If an error occurs during execution, then parameters ERROR and STATUS indicate the result.

A recipe DB must be created before a recipe export is possible. The name of a recipe data block is used as file name of the new CSV file. If a CSV file with an identical name already exists, then it is overwritten during the export operation.

You can use the File Browser page (Page 810) of the CPU's built-in Web server to access the recipe CSV file. The file is put in the recipe folder in the root directory of CPU load memory.

Table 9- 156 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	Control parameter REQUEST: Activates the export on a positive edge.
RECIPE_DB	In/Out	Variant	Pointer to the recipe data block. Refer to the "Recipe DB example (Page 412)" for details. The DB name characters must follow the Windows file system naming restrictions. Characters \ / : * ? " < > | and the space character are not allowed.
DONE	OUT	Bool	The DONE bit is TRUE for one scan, after the last request was completed with no error. (Default value: False)
BUSY	OUT	Bool	RecipeExport execution • 0: No operation in progress • 1: Operation on progress
ERROR	OUT	Bool	The ERROR bit is TRUE for one scan, after the last request was terminated with an error. The error code value at the STATUS parameter is valid only during the single scan where ERROR = TRUE. • 0: No warning or error • 1: An error has occurred. The STATUS parameter provides information on the type of error.
STATUS	OUT	Word	Execution condition code

Table 9- 157 Values of ERROR and STATUS

ERROR	STATUS (W#16#....)	Description
0	0000	No error
0	7000	Call with no REQ edge: BUSY = 0, DONE = 0
0	7001	First call with REQ edge (working): BUSY = 1, DONE = 0
0	7002	Nth call (working): BUSY = 1, DONE = 0
1	8070	All instance memory is in use.
1	8090	File name contains invalid characters
1	8091	The data structure referenced with RECIPE_DB cannot be processed.
1	8092	Data structure specified in RECIPE_DB exceeds 5000 bytes
1	80B3	Not enough space in on MC or in internal load memory
1	80B4	MC is write protected
1	80B6	Recipe DB attribute "Only store in load memory" is not enabled.
1	80C0	CSV file is temporarily locked
1	80C1	DB is temporarily locked

RecipeImport (Recipe import) instruction

Table 9- 158 RecipeImport instruction

LAD/FBD	SCL	Description

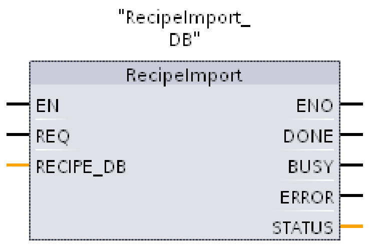
 "RecipeImport_DB"
RecipeImport
EN
REQ
RECIPE_DB
ENO
DONE
BUSY
ERROR
STATUS
	`"RecipeImport_DB"(`
	`    req:=_bool_in_,`
	`    done=>_bool_out_,`
	`    busy=>_bool_out_,`
	`    error=>_bool_out_,`
	`    status=>_word_out_,`
	`    Reci-`
	`pe_DB:=_variant_inout_);`
		The "RecipeImport" instruction imports recipe data from a CSV file, in CPU load memory, to a recipe data block referenced by the RECIPE_DB parameter. Start values in the recipe data block are overwritten by the import process. The import operation is triggered by the "REQ" parameter. The BUSY parameter is set to "1" during import processing. After the execution of RecipeImport stops, BUSY is reset to "0" and the completion of the operation is indicated with "1" at the DONE parameter. If an error occurs during execution, then parameters ERROR and STATUS indicate the result.

Table 9- 159 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	Control parameter REQUEST: Activates the import on a positive edge.
RECIPE_DB	In/Out	Variant	Pointer to the recipe data block. Refer to "Recipe DB example (Page 412)" for details. The DB name characters must follow the Windows file system naming restrictions. Characters \ / : * ? " < > | and the space character are not allowed.
DONE	OUT	Bool	The DONE bit is TRUE for one scan, after the last request was completed with no error. (Default value: False)
BUSY	OUT	Bool	• 0 - No operation in progress • 1 - Operation on progress
ERROR	OUT	Bool	The ERROR bit is TRUE for one scan, after the last request was terminated with an error. The error code value at the STATUS parameter is valid only during the single scan where ERROR = TRUE.
STATUS	OUT	Word	Execution condition code (Default value: 0)

A recipe DB that contains a structure which is consistent with the CSV file data structure must exist, before a recipe import operation is possible.

CSV file rules:

- The CSV file must be located in the root directory "Recipes" folder of internal load memory or external load memory, if an optional external "program" memory card is installed.
- The name of the CSV file must match the name of the data block at the RECIPE_DB parameter.
- The first line (header) of the CSV file contains the name of the recipe components. The first line is ignored during import. The names of the recipe components in the CSV file and the data block are not reconciled during the import process.
- In each case the first value in each line of the CSV file is the index number of the recipe. The individual recipes are imported in the order of the index. For this, the index in the CSV file has to be in ascending order and may contain no gaps (if this is not the case, the error message 80B0 is output at the STATUS parameter).
- The CSV file may not contain more recipe data records than provided for in the recipe data block. The maximum number of data records is indicated by the array limits in the data block.

Table 9- 160 Values of ERROR and STATUS

ERROR	STATUS (W#16#....)	Description
0	0000	No error
0	7000	Call with no REQ edge: BUSY = 0, DONE = 0
0	7001	First call with REQ edge (working): BUSY = 1, DONE = 0
0	7002	N^{th} call (working): BUSY = 1, DONE = 0
1	8070	All instance memory is in use.
1	8090	The file name contains invalid characters.
1	8092	No matching CSV file found for the import. Possible cause: The name of the CSV file does not match the name of the recipe DB.
1	80C0	CSV file is temporarily locked.
1	80C1	Data block is temporarily locked.
1	80B0	Numbering in the index of the CSV file is not continuous, not ascending or exceeds the maximum number (array limit) in the data block.
1	80B1	Structure of the recipe data block and the CSV file do not match: The CSV file contains too many fields.
1	80B2	Structure of the recipe data block and the CSV file do not match: The CSV file contains too few fields.
1	80B6	Recipe DB attribute "Only store in load memory" is not enabled.
1	80D0 +n	Structure of the recipe data block and the CSV file do not match: Data type in field n does not match (n<=46).
1	80FF	Structure of the recipe data block and the CSV file do not match: Data type in field n does not match (n>46).

9.7.1.4 Recipe example program

Prerequisites for the recipe example program

The prerequisites for the recipe example program are as follows:

- A recipe DB that stores all recipe records. The recipe DB is stored in load memory.
- An active recipe DB that stores a copy of one recipe in work memory.

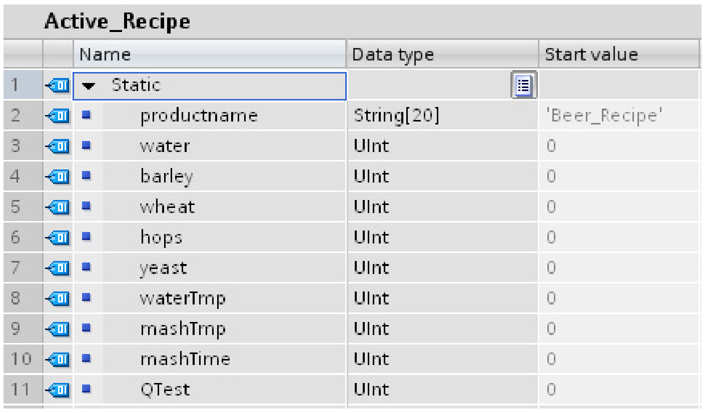
Refer to the "Recipe DB example (Page 412)" for details about the recipe DB and the corresponding CSV file.

Create the active recipe DB

On the "Add new block" window:

- Select the "Data block" button on the "Add new block" window
- On the "Type" drop-down menu, select the "Beer_recipe" PLC data type that you created previously.

Start values are not required. The DB data values are set when one recipe is transferred from the recipe DB to the active recipe DB. In the example, the active recipe DB is the destination for READ_DBL data and provides source data for WRITE_DBL. The following image shows the Active_Recipe DB.


Active_Recipe

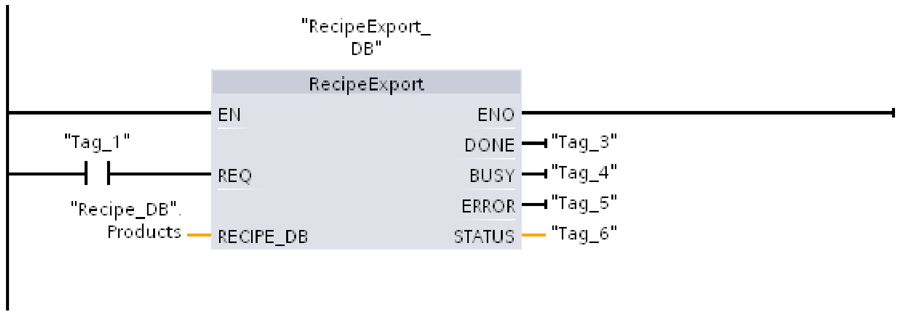
	Name	Data type	Start value
1	▼ Static		
2	productname	String[20]	'Beer_Recipe'
3	water	UInt	0
4	barley	UInt	0
5	wheat	UInt	0
6	hops	UInt	0
7	yeast	UInt	0
8	waterTmp	UInt	0
9	mashTmp	UInt	0
10	mashTime	UInt	0
11	QTest	UInt	0

Instance DBs

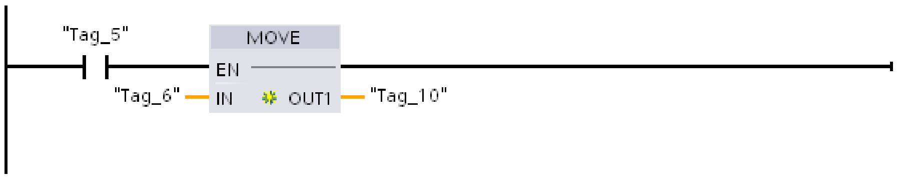
The instance DBs used by instructions RecipeExport ("RecipeExport_DB") and RecipeImport ("RecipeImport_DB") are created automatically when you place the instructions in your program. The instance DBs are used to control instruction execution and are not referenced in the program logic.

Example recipe program

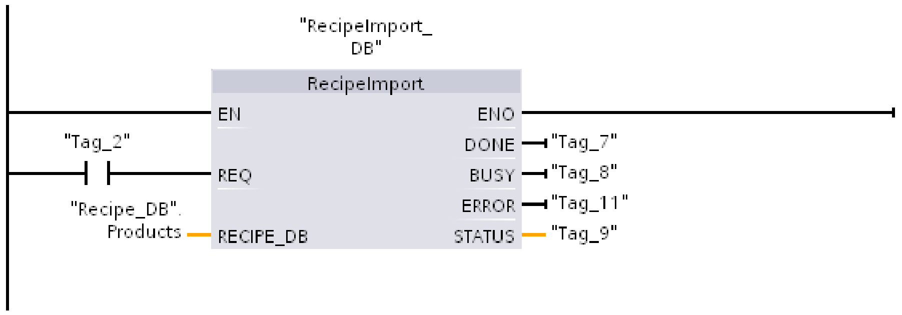
Network 1 A rising edge on REQ starts the export process. A CSV file is generated from the recipe DB data and placed in the CPU memory recipes folder.



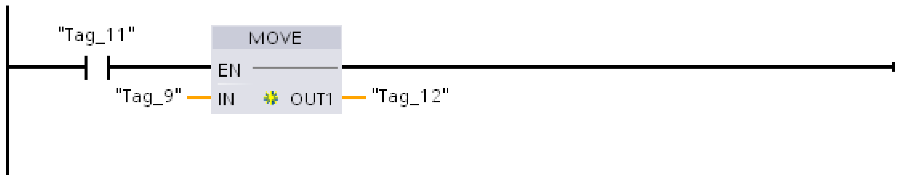
Network 2 Capture the STATUS output from RecipeExport execution, because it is only valid for one scan.



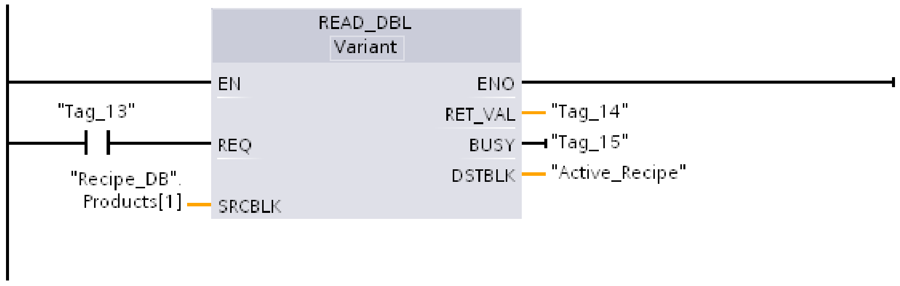
Network 3 A rising edge on REQ starts the import process. The existing recipe DB is loaded with all recipe data read from the corresponding CSV file that is located in the CPU memory recipes folder.



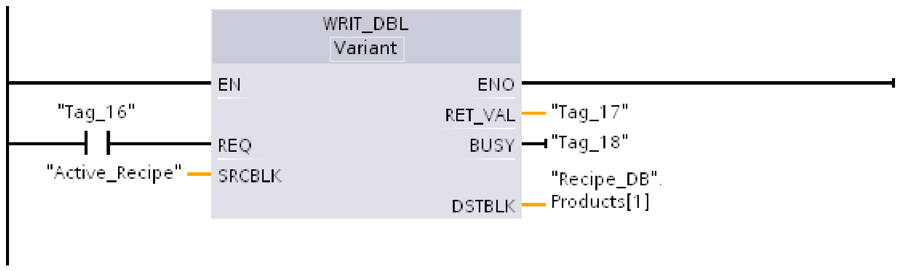
Network 4 Capture the STATUS output from RecipeImport execution, because it is only valid for one scan.



Network 5 READ_DBL copies the start values from one recipe "Recipe_DB". Products[1] (in CPU load memory) to the Active_Recipe DB current values (in CPU work memory). After READ_DBL execution, your program logic can access the recipe component values by addressing locations in the Active_Recipe DB. For example, the symbolic addresses ("Active_Recipe".productname) and ("Active_Recipe.water) provide your program logic with the current recipe name and quantity of water.



Network 6 During run time, An HMI device could modify a component value stored in the Active_Recipe DB. Improved recipe data can be stored by executing WRIT_DBL. In the example, all Recipe_DB start values for the single recipe "Recipe_DB". Products[1] are overwritten by the current values from the "Active_Recipe" DB.



9.7.2 Data logs

Your control program can use the Data log instructions to store run-time data values in persistent log files. The data log files are stored in flash memory (CPU or memory card). Log file data is stored in standard CSV (Comma Separated Value) format. The data records are organized as a circular log file of a pre-determined size.

The Data log instructions are used in your program to create, open, write a record, and close the log files. You decide which program values will be logged by creating a data buffer that defines a single log record. Your data buffer is used as temporary storage for a new log record. New current values must be programmatically moved into the buffer during run-time. When all of the current data values are updated, you can execute the DataLogWrite instruction to transfer data from the buffer to a data log record.

You can open, edit, save, rename, and delete data log files from the File Browser page of the Web Server. You must have read privileges to view the file browser and you must have modify privileges to edit, delete, or rename data log files.

9.7.2.1 Data log record structure

The DATA and HEADER parameters of the DataLogCreate instruction assign the data type and the column header description of all data elements in a log record.

DATA parameter for the DataLogCreate instruction

The DATA parameter points to memory used as a temporary buffer for a new log record and must be assigned to an M or DB location.

You can assign an entire DB (derived from a PLC data type that you assign when the DB is created) or part of a DB (the specified DB element can be any data type, data type structure, PLC data type, or data array).

Structure data types are limited to a single nesting level. The total number of data elements declared should correspond to the number of columns specified in the header parameter. The maximum number of data elements you can assign is 253 (with a timestamp) or 255 (without a timestamp). This restriction keeps your record inside the 256 column limit of an Excel sheet.

The DATA parameter can assign either retentive or non-retentive data elements in a "Standard" (compatible with S7-300/400) or "Optimized" DB type.

In order to write a Data log record you must first load the temporary DATA record with new process values and then execute the DataLogWrite instruction that saves new record values in the Datalog file.

HEADER parameter for the DataLogCreate instruction

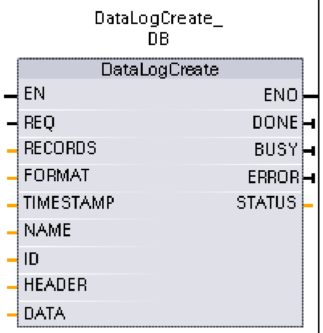
The HEADER parameter points to column header names for the top row of the data matrix encoded in the CSV file. HEADER data must be located in DB or M memory and the characters must follow standard CSV format rules with commas separating each column name. The data type may be a string, byte array, or character array. Character/byte arrays allow increased size, where strings are limited to a maximum of 255 bytes. The HEADER parameter is optional. If the HEADER is not assigned, then no header row is created in the Data log file.

9.7.2.2 Program instructions that control data logs

DataLogCreate (Create data log) instruction

Table 9- 161 DataLogCreate instruction

LAD/FBD	SCL	Description

 DataLogCreate_ DB
DataLogCreate
EN
ENO
REQ
DONE
RECORDS
BUSY
FORMAT
ERROR
TIMESTAMP
STATUS
NAME
ID
HEADER
DATA
	`"DataLogCreate_DB"(`
	`    req:=_bool_in_,`
	`    records:=_udint_in_,`
	`    format:=_uint_in_,`
	`    timestamp:=_uint_in_,`
	`    done=>_bool_out_,`
	`    busy=>_bool_out_,`
	`    error=>_bool_out_,`
	`    status=>_word_out_,`
	`    name:=_string_inout_,`
	`    ID:=_dword_inout_,`
	`    header:=_variant_inout_,`
	`    data:=_variant_inout_);`
		Creates and initializes a data log file. The file is created in the PLC \DataLogs directory, named by the NAME parameter, and implicitly opened for write operations. You can use the Data log instructions to programmatically store run-time process data in flash memory of the CPU.
		STEP 7 automatically creates the associated instance DB when you insert the instruction.

1 In the SCL example, "DataLogCreate_DB" is the name of the instance DB.

Table 9- 162 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	A low to high (positive edge) signal starts the operation. (Default value: False)
RECORDS	IN	UDint	The maximum number of data records the circular data log can contain before overwriting the oldest entry: The header record is not included. Sufficient available PLC load memory must exist in order to successfully create the data log. (Default value - 1)
FORMAT	IN	UInt	Data log format: • 0 - Internal format (not supported) • 1 - Comma separated values "csv-eng" (Default value)
TIMESTAMP	IN	UInt	Data time stamp format: Column headers for date and time fields are not required. The time stamp uses the system time (Coordinated Universal Time - UTC) and not the local time. • 0 - No time stamp • 1 - Date and time stamp (Default value)

Parameter and type		Data type	Description
NAME	IN	Variant	Data log name: You provide the name. This variant only supports a String data type and can only be located in local, DB, or M memory. (Default value: ' ') The string reference is also used as the name of the data log file. The name characters must follow the Windows file system naming restrictions. Characters \ / : * ? " < > | and the space character are not allowed.
ID	In/Out	DWord	Data log numeric identifier: You store this generated value for use with other Data log instructions. The ID parameter is only used as an output with the DataLog-Create instruction. (Default value: 0) Symbolic name access for this parameter is not allowed.
HEADER	In/Out	Variant	Pointer to data log column header names for the top row of the data matrix encoded in the CSV file. (Default val-ue: null). HEADER data must be located in DB or M memory. The characters must follow standard CSV format rules with commas separating each column name. The data type may be a string, byte array, or character array. Character/byte arrays allow increased size, where strings are limited to a maximum of 255 bytes. The HEADER parameter is optional. If the HEADER is not parameterized, then no header row is created in the Data log file.
DATA	In/Out	Variant	Pointer to the record data structure, user defined type (UDT), or array. Record data must be located in DB or M memory. The DATA parameter specifies the individual data ele-ments (columns) of a data log record and their data type. Structure data types are limited to a single nesting level. The number of data elements declared should corre-spond to the number of columns specified in the header parameter. The maximum number of data elements you can assign is 253 (with a timestamp) or 255 (without a timestamp). This restriction keeps your record inside the 256 column limit of an Excel sheet.
DONE	OUT	Bool	The DONE bit is TRUE for one scan, after the last re-quest was completed with no error. (Default value: False)
BUSY	OUT	Bool	• 0 - No operation in progress • 1 - Operation on progress
ERROR	OUT	Bool	The ERROR bit is TRUE for one scan, after the last request was terminated with an error. The error code value at the STATUS parameter is valid only during the single scan where ERROR = TRUE.
STATUS	OUT	Word	Execution condition code (Default value: 0)

A data log file is created with a pre-determined fixed sized based on the RECORDS and DATA parameters. The data records are organized as a circular log file. New records are appended to the data log file, until the maximum number of records that is specified by the RECORDS parameter is stored. The next record written will overwrite the oldest record. Another record write operation will overwrite the next oldest data record and so on.

Memory resource usage:

- The data logs consume only load memory.
- There is no set limit for the total number of data logs. The size of all data logs combined is limited by the available resources of load memory. Only eight data logs may be open at one time.
- The maximum possible number for the RECORDS parameter is the limit for an UDint number (4,294,967,295). The actual limit for the RECORD parameter depends on the size of a single record, the size of other data logs, and the available resources of load memory. In addition, Excel limits the number of rows allowed in an Excel sheet.

Note

Data log creation execution must be complete, before starting a data log write operation

- DataLogCreate and DataLogNewFile log file creation operations extend over many program scan cycles. The actual time required for the log file creation depends on the record structure and number of records. Your program logic must monitor and catch the DONE bit's transition to the TRUE state that signals the completion of a log file creation. If a DataLogWrite instruction is executed before a data log creation operation is complete, then the write operation will fail to write a new data log record as expected.
- In certain situations when a very fast program scan is running, data log creation can take an extended time. If the long creation time is too slow, you should ensure that the checkbox for the Enable minimum cycle time for cyclic OBs is active, and the minimum cycle time is set to one ms or greater. Refer to Configuring the cycle time and communication load (Page 101) for more information.

Note

The DataLogNewFile instruction copies an existing data log's record structure

If you want to prevent overwriting any data records, then you can use the DataLogNewFile instruction to create a new data log based on the current data log, after the current data log has stored the maximum number of records. New data records are stored in the new data log file. The old data log file and record data remain stored in flash memory.

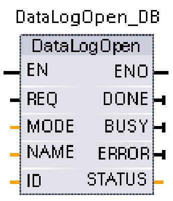
Table 9- 163 Values of ERROR and STATUS

ERROR	STATUS (W#16#....)	Description
0	0000	No error
0	7000	Call with no REQ edge: BUSY = 0, DONE = 0
0	7001	First call with REQ edge (working): BUSY = 1, DONE = 0
0	7002	N^th call (working): BUSY = 1, DONE = 0
1	8070	All internal instance memory is in use.
1	807F	Internal error
1	8090	Invalid file name
1	8091	Name parameter is not a String reference.
1	8093	A data log already exists with that name. Use a different name, make sure the existing data log's .csv file is not open, and then use the File Browser page (Page 810) of the Web Server to delete the existing data log.
1	8097	Requested file length exceeds file system maximum.
1	80B3	Insufficient load memory available.
1	80B4	MC (Memory Cartridge) is write-protected.
1	80C1	Too many open files: No more than eight opened data log files are allowed.
1	8253	Invalid record count
1	8353	Invalid format selection
1	8453	Invalid timestamp selection
1	8B24	Invalid HEADER area assignment: For example, pointing to local memory
1	8B51	Invalid HEADER parameter data type
1	8B52	Too many HEADER parameter data elements
1	8C24	Invalid DATA area assignment: For example, pointing to local memory
1	8C51	Invalid DATA parameter data type
1	8C52	Too many DATA parameter data elements

DataLogOpen (Open data log) instruction

Table 9- 164 DataLogOpen instruction

LAD / FBD	SCL	Description

 DataLogOpen_DB
DataLogOpen
EN
ENO
REQ
DONE
MODE
BUSY
NAME
ERROR
ID
STATUS
	`"DataLogOpen_DB"(`
	`    req:=_bool_in_,`
	`    mode:=_uint_in_,`
	`    done=>_bool_out_,`
	`    busy=>_bool_out_,`
	`    error=>_bool_out_,`
	`    status=>_word_out_,`
	`    name:=_string_inout_,`
	`    ID:=_dword_inout_);`
		Opens a pre-existing data log file. A data log must be opened before you can write new records to the log. Data logs can be opened and closed individually. A maximum of eight data logs can be open at the same time.
		STEP 7 automatically creates the associated instance DB when you insert the instruction.

2 In the SCL example, "DataLogOpen_DB" is the name of the instance DB.

Table 9- 165 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	A low to high (positive edge) signal starts the operation. (Default value: False)
MODE	IN	UInt	Operation mode: • 0 - Append to existing data (Default value) • 1 - Clear all existing records
NAME	IN	Variant	Name of an existing data log: This variant only supports a String data type and can only be located in local, DB, or M memory. (Default value: ' ')
ID	In/Out	DWord	Numeric identifier of a data log. (Default value: 0) **Note:** Symbolic name access for this parameter is not allowed.
DONE	OUT	Bool	The DONE bit is TRUE for one scan, after the last request was completed with no error. (Default value: False)
BUSY	OUT	Bool	• 0 - No operation in progress • 1 - Operation on progress
ERROR	OUT	Bool	The ERROR bit is TRUE for one scan, after the last request was terminated with an error. The error code value at the STATUS parameter is valid only during the single scan where ERROR = TRUE.
STATUS	OUT	Word	Execution condition code (Default value: 0)

You can provide either the NAME or an ID (ID parameter as an input) of a pre-existing data log. If you provide both parameters and a valid ID does correspond to the NAME data log, then the ID is used, and the NAME ignored.

The NAME must be the name of a data log created by the DataLogCreate instruction. If only the NAME is provided and the NAME specifies a valid data log, then the corresponding ID will be returned (ID parameter as an output).

Note

General usage of data log files

- Data log files are automatically opened after the DataLogCreate and DataLogNewFile operations.
- Data log files are automatically closed after a PLC run to stop transition or a PLC power cycle.
- A Data log file must be open before a new DataLogWrite operation is possible.
- A maximum of eight data log files may be open at one time. More than eight data log files may exist, but some of them must be closed so no more than eight are open.

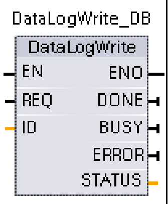
Table 9- 166 Values of ERROR and STATUS

ERROR	STATUS (W#16#)	Description
0	0000	No error
0	0002	Warning: Data log file already open by this application program
0	7000	Call with no REQ edge: BUSY = 0, DONE = 0
0	7001	First call with REQ edge (working): BUSY = 1, DONE = 0
0	7002	Nth call (working): BUSY = 1, DONE = 0
1	8070	All internal instance memory is in use.
1	8090	Data log definition is inconsistent with existing data log file.
1	8091	Name parameter is not a String reference.
1	8092	Data log does not exist.
1	80C0	Data log file is locked.
1	80C1	Too many open files: No more than eight opened data log files are allowed.

DataLogWrite (Write data log) instruction

Table 9- 167 DataLogWrite instruction

LAD / FBD	SCL	Description

 DataLogWrite_DB
DataLogWrite
EN
ENO
REQ
DONE
ID
BUSY
ERROR
STATUS
	`"DataLogWrite_DB"(`
	`req:=_bool_in_,`
	`done=>_bool_out_,`
	`busy=>_bool_out_,`
	`error=>_bool_out_,`
	`status=>_word_out_,`
	`ID:=_dword_inout_);`
		Writes a data record into the specified data log. The pre-existing target data log must be open before a DataLogWrite operation is allowed.
		STEP 7 automatically creates the associated instance DB when you insert the instruction.

2 In the SCL example, "DataLogWrite_DB" is the name of the instance DB.

Table 9- 168 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	A low to high (positive edge) signal starts the operation. (Default value: False)
ID	In/Out	DWord	Numeric data log identifier. Only used as an input for the DataLogWrite instruction. (Default value: 0) **Note:** Symbolic name access for this parameter is not allowed.
DONE	OUT	Bool	The DONE bit is TRUE for one scan, after the last request was completed with no error.
BUSY	OUT	Bool	• 0 - No operation in progress • 1 - Operation on progress
ERROR	OUT	Bool	The ERROR bit is TRUE for one scan, after the last request was terminated with an error. The error code value at the STATUS parameter is valid only during the single scan where ERROR = TRUE.
STATUS	OUT	Word	Execution condition code (Default value: 0)

The memory address and data structure of the record buffer is configured by the DATA parameter of a DataLogCreate instruction. You must programmatically load the record buffer with current run-time process values and then execute the DataLogWrite instruction to move new record data from the buffer to the data log.

The ID parameter identifies a data log and data record configuration. The ID number is generated when a data log is created.

If there are empty records in the circular data log file, then the next available empty record will be written. If all records are full, then the oldest record will be overwritten.

NOTICE

Data log creation operations must be complete, before starting a data log write operation

DataLogCreate and DataLogNewFile log file creation operations extend over many program scan cycles. The actual time required for the log file creation depends on the record structure and number of records. Your program logic must monitor and catch the DONE bit's transition to the TRUE state that signals the completion of a log file creation. If a DataLogWrite instruction is executed before a data log creation operation is complete, then the write operation will fail to write a new data log record as expected.

Note

Effect of data logs on internal CPU memory

Each data log write consumes at a minimum 2 KB of memory. If your program writes small amounts of data frequently, it is consuming at least 2 KB of memory on each write. A better implementation would be to accumulate the small data items in a data block (DB), and to write the data block to the data log at less frequent intervals.

If your program writes many data log entries at a high frequency, consider using a replaceable SD memory card.

NOTICE

Potential for data log data loss during a CPU power failure

If there is a power failure during an incomplete DataLogWrite operation, then the data record being transferred to the data log could be lost.

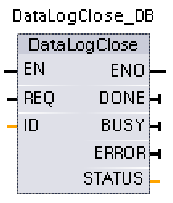
Table 9- 169 Values of ERROR and STATUS

ERROR	STATUS (W#16#)	Description
0	0000	No error
0	0001	Indicates that the data log is full: Each data log is created with a specified maximum number of records. The last record of the maximum number has been written. The next write operation will overwrite the oldest record.
0	7000	Call with no REQ edge: BUSY = 0, DONE = 0
0	7001	First call with REQ edge (working): BUSY = 1, DONE = 0
0	7002	Nth call (working): BUSY = 1, DONE = 0
1	8070	All internal instance memory is in use.
1	8092	Data log does not exist.
1	80B0	Data log file is not open (for explicit open mode only).

DataLogClose (Close data log) instruction

Table 9- 170 DataLogClose instruction

LAD / FBD	SCL	Description

 DataLogClose_DB
DataLogClose
EN
ENO
REQ
DONE
ID
BUSY
ERROR
STATUS
	`"DataLogClose_DB"(`
	`    req:=_bool_in_,`
	`    done=>_bool_out_,`
	`    busy=>_bool_out_,`
	`    error=>_bool_out_,`
	`    status=>_word_out_,`
	`    ID:=_dword_inout_);`
		Closes an open data log file. DataLogWrite operations to a closed data log result in an error. No write operations are allowed to this data log until another DataLogOpen operation is performed.
		A transition to STOP mode will close all open data log files.
		STEP 7 automatically creates the associated instance DB when you insert the instruction.

2 In the SCL example, "DataLogClose_DB" is the name of the instance DB.

Table 9- 171 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	A low to high (positive edge) signal starts the operation. (Default value: False)
ID	In/Out	DWord	Numeric identifier of a data log. Only used as an input for the DataLogClose instruction. (Default value: 0) **Note:** Symbolic name access for this parameter is not allowed.
DONE	OUT	Bool	The DONE bit is TRUE for one scan after the last request was completed with no error.
BUSY	OUT	Bool	• 0 - No operation in progress • 1- Operation on progress
ERROR	OUT	Bool	The ERROR bit is TRUE for one scan, after the last request was terminated with an error. The error code value at the STATUS parameter is valid only during the single scan where ERROR = TRUE.
STATUS	OUT	Word	Execution condition code (Default value: 0)

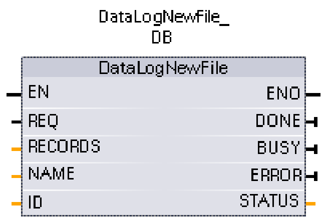
Table 9- 172 Values of ERROR and STATUS

ERROR	STATUS (W#16#)	Description
0	0000	No error
0	0001	Data log not open
0	7000	Call with no REQ edge: BUSY = 0, DONE = 0
0	7001	First call with REQ edge (working): BUSY = 1, DONE = 0
0	7002	Nth call (working): BUSY = 1, DONE = 0
1	8092	Data log does not exist.

DataLogNewFile (Data log in new file) instruction

Table 9- 173 DataLogNewFile instruction

LAD / FBD	SCL	Description

 DataLogNewFile_ DB
DataLogNewFile
EN
ENO
REQ
DONE
RECORDS
BUSY
NAME
ERROR
ID
STATUS
	`"DataLogNewFile_DB"(`
	`req:=_bool_in_,`
	`records=:_udint_in_,`
	`done=>_bool_out_,`
	`busy=>_bool_out_,`
	`error=>_bool_out_,`
	`status=>_word_out_,`
	`name=:_DataLog_out_,`
	`ID:=_dword_inout_);`
		Allows your program to create a new data log file based upon an existing data log file.
		STEP 7 automatically creates the associated instance DB when you insert the instruction.

2 In the SCL example, "DataLogNewFile_DB" is the name of the instance DB.

Table 9- 174 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	A low to high (positive edge) signal starts the operation. (Default value: False)
RECORDS	IN	UDInt	The maximum number of data records the circular data log can contain before overwriting the oldest entry. (Default value: 1) The header record is not included. Sufficient available CPU load memory must exist in order to successfully create the data log.
NAME	IN	Variant	Data log name: You provide the name. This variant only supports a String data type and can only be located in local, DB, or M memory. (Default value: ' ') The string reference is also used as the name of the data log file. The name characters must follow the Windows file system naming restrictions. Characters \ / : * ? " < > | and the space character are not allowed.)
ID	In/Out	DWord	Numeric data log identifier(Default value: 0): • At execution, the ID input identifies a valid data log. The new data log configuration is copied from this data log. • After execution, the ID parameter becomes an output that returns the ID of the newly created data log file. **Note:** Symbolic name access for this parameter is not allowed.
DONE	OUT	Bool	The DONE bit is TRUE for one scan, after the last request was completed with no error.
BUSY	OUT	Bool	• 0 - No operation in progress • 1 - Operation on progress
ERROR	OUT	Bool	The ERROR bit is TRUE for one scan, after the last request was terminated with an error. The error code value at the STATUS parameter is valid only during the single scan where ERROR = TRUE.
STATUS	OUT	Word	Execution condition code (Default value: 0)

You can execute the DataLogNewFile instruction when a data log becomes full or is deemed completed and you do not want to lose any data that is stored in the data log. A new empty data log file can be created based on the structure of the full Data log file. The header record will be duplicated from the original data log with the original data log properties (DATA record buffer, data format, and timestamp settings). The original Data log file is implicitly closed and the new Data log file is implicitly opened.

DataLogWrite parameter trigger: Your program must monitor the ERROR and STATUS parameters of each DataLogWrite operation. When the final record is written and a data log is full, the DataLogWrite ERROR bit = 1 and the DataLogWrite STATUS word = 1. These ERROR and STATUS values are valid for one scan only, so your monitoring logic must use ERROR = 1 as a time gate to capture the STATUS value and then test for STATUS = 1 (the data log is full).

DataLogNewFile operation: When your program logic gets the data log is full signal, this state is used to activate a DataLogNewFile operation. You must execute DataLogNewFile with the ID of an existing (usually full) and open data log, but a new unique NAME parameter. After the DataLogNewFile operation is done, a new data log ID value is returned (as an output parameter) that corresponds to the new data log name. The new data log file is implicitly opened and is ready to store new records. New DataLogWrite operations that are directed to the new data log file, must use the ID value returned by the DataLogNewFile operation.

NOTICE

Data log creation operations must be complete, before starting a data log write operation

DataLogCreate and DataLogNewFile log file creation operations extend over many program scan cycles. The actual time required for the log file creation depends on the record structure and number of records. Your program logic must monitor and catch the DONE bit's transition to the TRUE state that signals the completion of a log file creation. If a DataLogWrite instruction is executed before a data log creation operation is complete, then the write operation will fail to write a new data log record as expected.

Table 9- 175 Values of ERROR and STATUS

ERROR	STATUS (W#16#)	Description
0	0000	No error
0	7000	Call with no REQ edge: BUSY = 0, DONE = 0
0	7001	First call with REQ edge (working): BUSY = 1, DONE = 0
0	7002	Nth call (working): BUSY = 1, DONE = 0
1	8070	All internal instance memory is in use.
1	8090	Invalid file name
1	8091	Name parameter is not a String reference.
1	8092	Data log does not exist.
1	8093	Data log already exists.
1	8097	Requested file length exceeds file system maximum.
1	80B3	Insufficient load memory available.
1	80B4	MC is write-protected.
1	80C1	Too many open files.

9.7.2.3 Working with data logs

The data log files are stored as comma separated value format (*.csv) in persistent flash memory. You can view the data logs by using the PLC Web server feature or by removing the PLC memory card and inserting it in a standard PC card reader.

Viewing data logs with the PLC Web server feature

If the PLC PROFINET port and a PC are connected to a network, then you can use a PC web browser like Microsoft Internet Explorer or Mozilla Firefox to access the built-in PLC Web server. The PLC may be in run mode or stop mode when you operate the PLC Web server. If the PLC is in run mode, then your control program continues to execute while the PLC Web server is transferring log data through the network.

Web server access:

1. Enable the Web server in the Device Configuration for the target CPU (Page 787).
2. Connect your PC to the PLC through the PROFINET network (Page 791).
3. Access the CPU through the built-in Web server (Page 794).
4. Download, edit, and delete data log files with the "File Browser" standard Web page (Page 810).
5. Open the .csv file with a spreadsheet application like Microsoft Excel.

Note

Data log management

Keep no more than 1000 data logs in a file system. Exceeding this number can prevent the Web server from having enough CPU resources to display the data logs.

If you find that the File Browser Web page is not able to display the data logs, then you must place the CPU in STOP mode in order to display and delete data logs.

Manage your data logs to ensure that you only keep the number that you need to maintain, and do not exceed 1000 data logs.

Viewing data logs on a PLC memory card

If the S7-1200 CPU has a "Program" type S7-1200 memory card inserted, then you can remove the memory card and insert the card into a standard SD (Secure Digital) or MMC (MultiMediaCard) card slot on a PC or PG. The PLC is in stop mode when the memory card is removed and your control program is not executed.

Use the Windows file explorer and navigate to the \DataLog directory on the memory card. All your *.csv data log files are located in this directory.

Make a copy of the data log files and put the copies on a local drive of your PC. Then, you can use Excel to open a local copy of a *.csv file and not the original file that is stored on the memory card.

NOTICE

You can copy, but do not modify or delete data log files on a S7-1200 memory card using a PC card reader

The standard Web server File Browser page is the recommended tool for viewing, downloading (copying), and deleting data log files.

Direct browsing of the memory card file system by the Windows Explorer has the risk that you can accidentally delete/modify data log or other system files which can corrupt a file or make the memory card unusable.

NOTICE

Effect of data logs on memory cards

To ensure the overall performance and robustness of your system, limit the data log rate to no faster than every 200 ms.

9.7.2.4 Limit to the size of data log files

Data log files share PLC load memory space with the program, program data, configuration data, user-defined Web pages, and PLC system data. A large program using internal load memory requires a large amount of load memory. There may be insufficient free space for data log files. In this case, you can use a "Program card" (Page 138) to increase the size of load memory. S7-1200 CPUs can use either internal or external load memory, but not both at once.

Maximum size rule for Data log files

The maximum size of one data log file cannot exceed the free load memory size or 500 megabytes, whichever is smaller. The size of 500 megabytes in this case refers to the decimal definition of megabyte, such that the maximum data log file size is 500,000,000 bytes or 500 x 1000^2 bytes.

Table 9- 176 Load memory size

Data area	**CPU 1211C**	**CPU 1212C**	**CPU 1214C**	**CPU 1215C, CPU 1217C**	**Data storage**
Internal load memory flash memory	1 MB	1 MB	4 MB	4 MB	User program and program data, con-figuration data, Data logs, user-defined Web pages, and PLC system data
External load memory Optional "Program card" flash memory	4 MB, 12 MB, 24 MB, 256 MB, 2 GB, or 32 GB depending on the SD card size				

Determining load memory free space

The amount of load memory free space varies during normal operations as the operating system uses and releases memory. Use the following steps to view the load memory memory size.

1. Establish an online connection between STEP 7 and the target S7-1200 PLC.
2. Download the program that controls your data log operations.
3. Create any optional user-defined Web pages that you need. The standard Web pages that access data logs are stored in PLC firmware and do not use load memory.
4. Use the Online and diagnostic tools to view total load memory size and free space (Page 1065).

Calculating the size of a data log file (all data records)

When the data log file is created the maximum memory size is allocated. In addition to the size required for all the data records, you must include storage space for a data log header (if used), time stamp header (if used), record index header, and the minimum block size for memory allocation.

Use the following formula to determine the size of your data log files and ensure you do not violate the maximum size rule.

Data log data bytes = ((data bytes in one record + time stamp bytes + 12 bytes) * number of records)

Header

Data log header bytes = header character bytes + 2 bytes

Header character bytes

- No data header and no timestamps = 7 bytes
- No data header and timestamps (has a timestamp header) = 21 bytes
- Data headers and no timestamps = number of character bytes in all column head text including separator commas
- Data headers and timestamps (has a timestamp header) = number of character bytes in all column head text including separator commas +21 bytes

Data

Data log data bytes = ((data bytes in one record + time stamp bytes + 12 bytes) * number of records)

Data bytes in one data record

The DataLogCreate DATA parameter points to a structure that assigns the number of data fields and the data type of each data field for one data log record.

Multiply the number of occurrences for a given data type by the number of bytes required. Repeat the process for each data type in a record and sum all the data bytes to get the total of all data elements in one record.

Size of individual data elements

Log data is stored as character bytes in the CSV (comma separated values) file format. The following table shows the number of bytes that are required to store each data element.

Data type	Number of bytes (includes data plus one comma byte)
Bool	2
Byte	5
Word	7
DWord	12
Char	4

String	**Example 1**: MyString String[10] The maximum string size is assigned as 10 characters. Text characters + automatic padding with blank characters = 10 bytes Opening and closing double quote + comma characters = 3 bytes 10 + 3 = 13 total bytes **Example 2**: Mystring2 String If no size is assigned with square brackets, then 254 bytes is allocated by default. Text characters + automatic padding with blank characters = 254 bytes Opening and closing double quote + comma characters = 3 bytes 254 + 3 = 257 total bytes
USInt	5
UInt	7
UDInt	12
SInt	5
Int	7
DInt	12
Real	16
LReal	25
Time	15
DTL	24

Number of records in a data log file

The RECORDS parameter of the DataLogCreate instruction sets the maximum number of records in a data log file.

Time stamp bytes in one data record

- No time stamp = 0 bytes
- Time stamp = 20 bytes

9.7.2.5 Data log example program

This Data log example program does not show all the program logic necessary to get sample values from a dynamic process, but does show the key operations of the Data log instructions. The structure and number of log files that you use depends on your process control requirements.

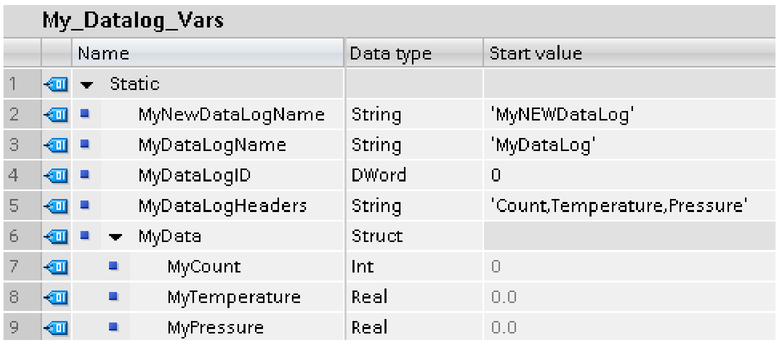
Note

General usage of Data log files

- Data log files are automatically opened after the DataLogCreate and DataLogNew File operations.
- Data log files are automatically closed after a PLC run to stop transition or a PLC power cycle.
- A Data log file must be open before a DataLogWrite operation is possible.
- A maximum of eight data log files may be open at one time. More than eight data log files may exist, but some of them must be closed so no more than eight are open.

Example Data log program

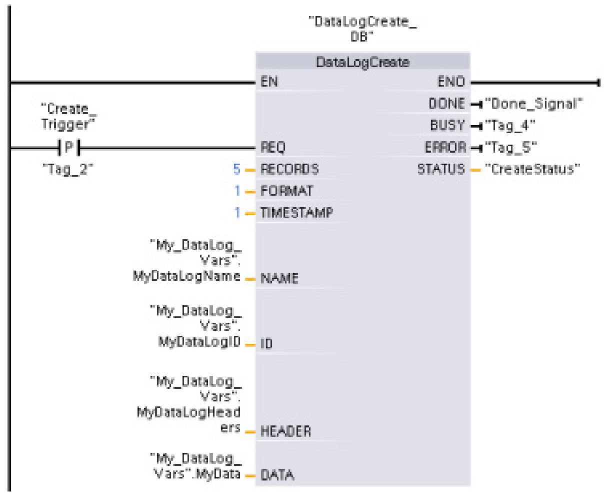
Example data log names, header text, and the MyData structure are created in a data block. The three MyData variables temporarily store new sample values. The process sample values at these DB locations are transferred to a data log file by executing the DataLogWrite instruction.



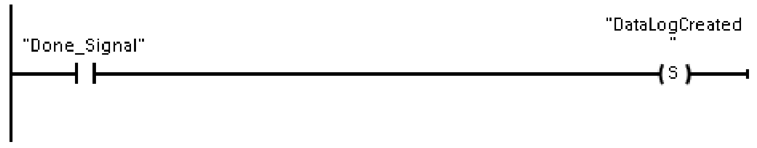
My_Datalog_Vars

	Name	Data type	Start value
1	Static		
2	MyNewDataLogName	String	'MyNEWDataLog'
3	MyDataLogName	String	'MyDataLog'
4	MyDataLogID	DWord	0
5	MyDataLogHeaders	String	'Count,Temperature,Pressure'
6	MyData	Struct	
7	MyCount	Int	0
8	MyTemperature	Real	0.0
9	MyPressure	Real	0.0

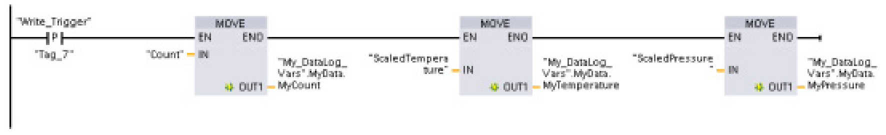
Network 1 REQ rising edge starts the data log creation process.



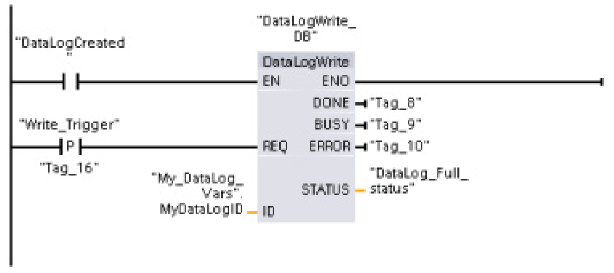
Network 2 Capture the DONE output from DataLogCreate because it is only valid for one scan.



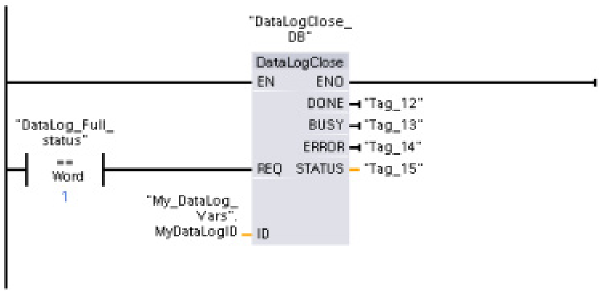
Network 3 A positive edge signal triggers when to store new process values in the MyData structure.


"Write_Trigger"
P
"Tag_7"
MOVE
EN
ENO
"Count" IN
OUT1 "My_DataLog_Vars".MyData.MyCount
MOVE
EN
ENO
"ScaledTemperature" IN
OUT1 "My_DataLog_Vars".MyData.MyTemperature
MOVE
EN
ENO
"ScaledPressure" IN
OUT1 "My_DataLog_Vars".MyData.MyPressure

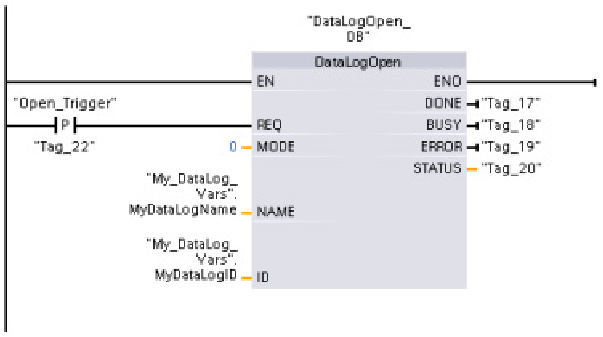
Network 4 The EN input state is based upon when the DataLogCreate operation is complete. A create operation extends over many scan cycles and must be complete before executing a write operation. The positive edge signal on the REQ input is the event that triggers an enabled write operation.



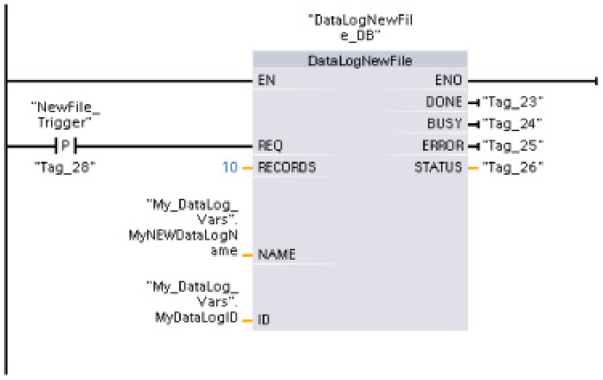
Network 5 Close the data log once the last record has been written. After executing the DataLogWrite operation that writes the last record, the log file full status is signaled when DataLogWrite STATUS output = 1.



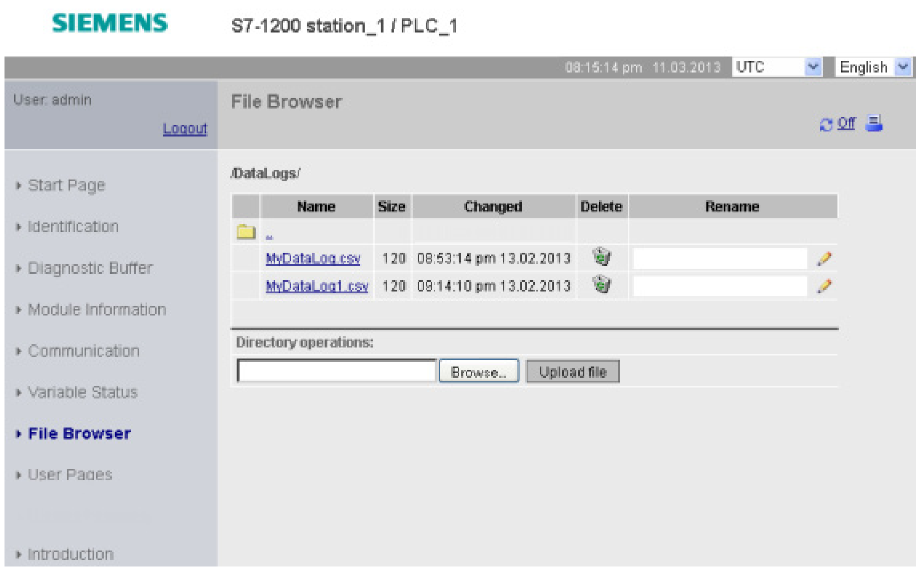
Network 6 A positive signal edge DataLogOpen REQ input simulates the user pushing a button on an HMI that opens a data log file. If you open a Data log file that has all records filled with process data, then the next DataLogWrite operation will overwrite the oldest record. You may want to preserve the old Data log and instead create a new data log, as shown in network 7.



Network 7 The ID parameter is an IN/OUT type. First, you supply the ID value of the existing Data log whose structure you want to copy. After the DataLogNewFile operation is complete, a new and unique ID value for the new Data log is written back to the ID reference location. The required DONE bit = TRUE capture is not shown, refer to networks 1, 2, and 4 for an example of DONE bit logic.



Data log files created by the example program viewed with the S7-1200 CPU Web server



SIEMENS S7-1200 station_1 / PLC_1

08:15:14 pm 11.03.2013 UTC English

User: admin
Logout

- Start Page
- Identification
- Diagnostic Buffer
- Module Information
- Communication
- Variable Status
- **File Browser**
- User Pages
- Introduction

File Browser

/DataLogs/

Name	Size	Changed	Delete	Rename
..				
MyDataLog.csv	120	08:53:14 pm 13.02.2013		
MyDataLog1.csv	120	09:14:10 pm 13.02.2013		

Directory operations:

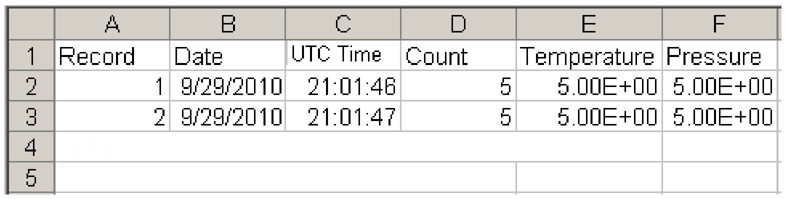
Browse... Upload file

① The "Delete" option is not available if you are not logged in with modify privileges.

② The "Rename" option is not available if you are not logged in with modify privileges.

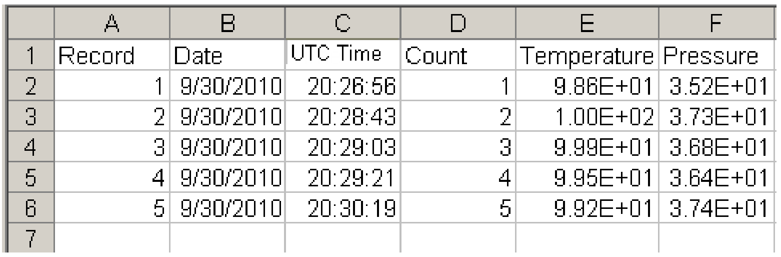
Table 9- 177 Downloaded .csv file examples viewed with Excel

Two records written in a five record maximum file



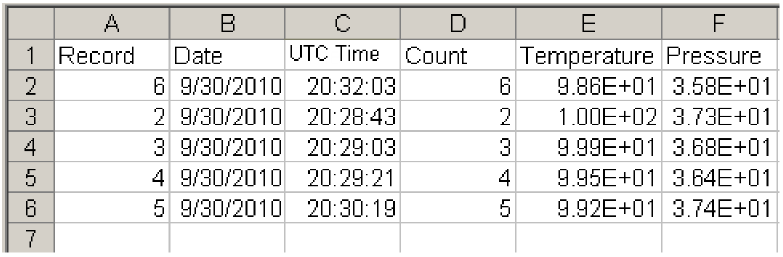
	A	B	C	D	E	F
1	Record	Date	UTC Time	Count	Temperature	Pressure
2	1	9/29/2010	21:01:46	5	5.00E+00	5.00E+00
3	2	9/29/2010	21:01:47	5	5.00E+00	5.00E+00
4						
5						

Five records in a Data log file with a five record maximum



	A	B	C	D	E	F
1	Record	Date	UTC Time	Count	Temperature	Pressure
2	1	9/30/2010	20:26:56	1	9.86E+01	3.52E+01
3	2	9/30/2010	20:28:43	2	1.00E+02	3.73E+01
4	3	9/30/2010	20:29:03	3	9.99E+01	3.68E+01
5	4	9/30/2010	20:29:21	4	9.95E+01	3.64E+01
6	5	9/30/2010	20:30:19	5	9.92E+01	3.74E+01
7						

After one additional record is written to the file above which is full, the sixth write operation overwrites the oldest record one with record six. Another write operation will overwrite record two with record seven and so on.



	A	B	C	D	E	F
1	Record	Date	UTC Time	Count	Temperature	Pressure
2	6	9/30/2010	20:32:03	6	9.86E+01	3.58E+01
3	2	9/30/2010	20:28:43	2	1.00E+02	3.73E+01
4	3	9/30/2010	20:29:03	3	9.99E+01	3.68E+01
5	4	9/30/2010	20:29:21	4	9.95E+01	3.64E+01
6	5	9/30/2010	20:30:19	5	9.92E+01	3.74E+01
7						

Note

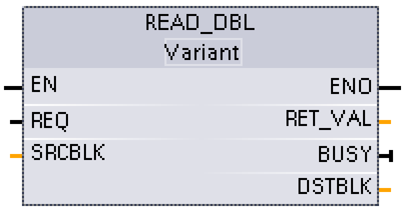
Data logs no longer use an //END marker to mark the end of a data log file that is not full. Prior to V4.1 of the S7-1200 CPU, data logs that were not full included an //END marker.

9.8 Data block control

9.8.1 READ_DBL and WRIT_DBL (Read/write a data block in load memory) instructions

Table 9- 178 READ_DBL and WRIT_DBL instructions

LAD / FBD	SCL	Description

 READ_DBL
Variant
EN
ENO
REQ
RET_VAL
SRCBLK
BUSY
DSTBLK
	`READ_DBL(`
	`    req:=_bool_in_,`
	`    srcblk:=_variant_in_,`
	`    busy=>_bool_out_,`
	`    dstblk=>_variant_out_);`
		Copies DB start values or part of the values, from load memory to a target DB in the work memory.
		The content of load memory is not changed during the copy process.

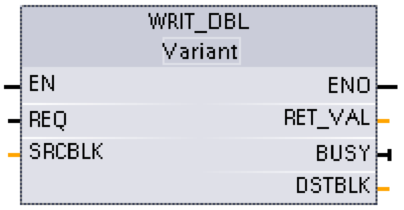
 WRIT_DBL
Variant
EN
ENO
REQ
RET_VAL
SRCBLK
BUSY
DSTBLK
	`WRIT_DBL(`
	`    req:=_bool_in_,`
	`    srcblk:=_variant_in_,`
	`    busy=>_bool_out_,`
	`    dstblk=>_variant_out_);`
		Copies DB current values or part of the values from work memory to a target DB in load memory.
		The content of work memory is not changed during the copy process.

Table 9- 179 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	BOOL	A high signal starts the operation, if BUSY = 0.
SRCBLK	IN	VARIANT	READ_DBL: Pointer to the source data block in load memory WRIT_DBL: Pointer to the source data block in work memory
RET_VAL	OUT	INT	Execution condition code
BUSY	OUT	BOOL	BUSY = 1 signals that the reading/writing process is not complete.
DSTBLK	OUT	VARIANT	READ_DBL: Pointer to the destination data block in work memory WRIT_DBL: Pointer to the destination data block in load memory

Typically, a DB is stored in both load memory (flash) and work memory (RAM). The start values (initial values) are always stored in load memory, and the current values are always stored in work memory. READ_DBL can be used to copy a set of start values from load memory to the current values of a DB in work memory that is referenced by your program. You can use WRIT_DBL to update the start values stored in internal load memory or memory card from current values in work memory.

Note

Effect of WRIT_DBL and READ_DBL instruction on flash memory

The WRIT_DBL instruction performs write operations in flash memory (internal load memory or memory card). To avoid reducing the lifetime of the flash memory, use the WRIT_DBL instrucion for infrequent updates such as recording changes to a production process. For similar reasons, avoid frequent calls to READ_DBL for read operations.

You must create the data blocks for READ_DBL and WRIT_DBL prior to calling these instructions in the STEP 7 program. If you created the source DB as a "standard" type then the destination DB must also be the "standard" type. If you created the source data block as an "optimized" type then the destination data block must also be the "optimized" type.

If the DBs are standard, then you can specify either a tag name or a P# value. The P# value allows you to specify and copy any number of elements of the specified size (Byte, Word, or DWord). Thus, you can copy part or all of a DB. If the DBs are optimized, you can only specify a tag name; you cannot use the P# operator. If you specify a tag name for either standard or optimized DBs (or for other work-memory types), then the instruction copies the data that this tag name references. This could be a user-defined type, an array, or a basic element. You can only use type Struct with these instructions if the DB is standard, not optimized. You must use a user-defined type (UDT) if it is a structure in optimized memory. Only a user-defined type ensures that the "data types" are exactly the same for both the source and destination structures.

Note

Using a structure (data type Struct) in an "optimized" DB

When using a Struct data type with "optimized" DBs, you must first create a user-defined data type (UDT) for the Struct. You then configure both the source and destination DBs with the UDT. The UDT ensures that the data types within the Struct remain consistent for both DBs.

For "standard" DBs, you use the Struct without creating a UDT.

READ_DBL and WRIT_DBL execute asynchronously to the cyclic program scan. The processing extends over multiple READ_DBL and WRIT_DBL calls. You start the DB transfer job by calling with REQ = 1 and then monitor the BUSY and RET_VAL outputs to determine when the data transfer is complete and correct.

Note

Effect of WRIT_DBL and READ_DBL instruction on communication load

When the WRIT_DBL or READ_DBL instruction is continually active, it can consume communication resources to the point that STEP 7 loses communication with the CPU. For this reason, use a positive edge input (Page 215) for the REQ parameter rather than a normally open or closed input (Page 209) that would remain on (signal level high) for multiple scans.

To ensure data consistency, do not modify the destination area during the processing of READ_DBL or the source area during the processing of WRIT_DBL (that is, as long as the BUSY parameter is TRUE).

SRCBLK and DSTBLK parameter restrictions:

- A data block must have been previously created before it can be referenced.
- The length of a VARIANT pointer of type BOOL must be divisible by 8.
- The length of a VARIANT pointer of type STRING must be the same in the source and destination pointers.

Recipes and machine setup information

You can use the READ_DBL and WRIT_DBL instructions to manage recipes or machine setup information. This essentially becomes another method of achieving retentive data for values that do not change often, although you would want to limit the number of writes to prevent wearing out the flash prematurely. This effectively allows you to increase the amount of retentive memory beyond that supported for the normal power-down retentive data, at least for values that do not change often. You could save recipe information or machine-setup information from work memory to load memory using the WRIT_DBL instruction, and you could retrieve such information from load memory into work memory using the READ_DBL instruction.

Table 9- 180 Condition codes

RET_VAL (W#16#...)	Description
0000	No error
0081	Warning: that the source area is smaller than the destination area. The source data is copied completely with the extra bytes in the destination area unchanged.
7000	Call with REQ = 0: BUSY = 0
7001	First call with REQ = 1 (working): BUSY = 1
7002	Nth call (working): BUSY = 1
8051	Data block type error
8081	The source area is larger than the destination area. The destination area is completely filled and the remaining bytes of the source are ignored.
8251	Source data block type error
82B1	Missing source data block
82C0	The source DB is being edited by another statement or a communication function.
8551	Destination data block type error
85B1	Missing destination data block
85C0	The destination DB is being edited by another statement or a communication function.
80C3	More than 50 READ_DBL or 50 WRIT_DBL statements are currently queued for execution.

See also Recipes (Page 411)

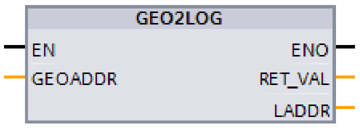
9.9 Address handling

9.9.1 GEO2LOG (Determine the hardware identifier from the slot) instruction

You use the GEO2LOG instruction to determine the hardware identifier based upon slot information.

Table 9- 181 GEO2LOG instruction

LAD / FBD	SCL	Description

	`ret_val := GEO2LOG(`
	`    GEOADDR:=_variant_in_out_,`
	` laddr:=_word_out_);`
		You use the GEO2LOG instruction to determine the hardware identifier based upon slot information.

The GEO2LOG instruction determines the hardware identifier based upon slot information that you define using the GEOADDR system data type:

Depending on the type of hardware you define at the parameter HWTYPE, the following information is evaluated from the other GEOADDR parameters:

- With HWTYPE = 1 (PROFINET IO system):
  - Only IOSYSTEM is evaluated. The other parameters of GEOADDR are not taken into consideration.
  - The hardware identifier of the PROFINET IO system is output.
- With HWTYPE = 2 (PROFINET IO device):
  - IOSYSTEM and STATION are evaluated. The other parameters of GEOADDR are not taken into consideration.
  - The hardware identifier of the PROFINET IO device is output.
- With HWTYPE = 3 (rack):
  - Only IOSYSTEM and STATION are evaluated. The other parameters of GEOADDR are not taken into consideration.
  - The hardware identifier of the rack is output.
- With HWTYPE = 4 (module):
  - IOSYSTEM, STATION, and SLOT are evaluated. The SUBSLOT parameter of GEOADDR is not taken into consideration.
  - The hardware identifier of the module is output.
- With HWTYPE = 5 (submodule):
  - All parameters of GEOADDR are evaluated.
  - The hardware identifier of the submodule is output.

The AREA parameter of the GEOADDR system data type is not evaluated.

Table 9- 182 Data types for the parameters

Parameter and type		Data type	Description
GEOADDR	IN/OUT or IN ?	Variant	Pointer to the structure of the GEOADDR system data type. The GEOADDR system data type contains the slot information from which the hardware ID is determined. Refer to the "GEOADDR system data type (Page 453)" for further information.
RET_VAL	OUT or RETURN ?	Int	Output of error information.
LADDR	OUT	HW_ANY	Hardware identifier of the assembly or the module. The number is automatically assigned and is stored in the properties in the hardware configuration.

For further information on valid data types, refer to "Overview of the valid data types" in the STEP 7 online help.

Table 9- 183 Condition codes

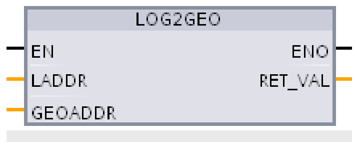
RET_VAL* (W#16#...)	Explanation
0	No error occurred.
8091	Invalid value for in GEOADDR for HWTYPE.
8094	Invalid value for in GEOADDR for IOSYSTEM.
8095	Invalid value for in GEOADDR for STATION.
8096	Invalid value for in GEOADDR for SLOT.
8097	Invalid value for in GEOADDR for SUBSLOT.
* The error codes can be displayed as integer or hexadecimal values in the program editor.	

9.9.2 LOG2GEO (Determine the slot from the hardware identifier) instruction

You use the LOG2GEO instruction to determine the geographical address (module slot) from the logical address belonging to a hardware identifier.

Table 9- 184 LOG2GEO instruction

LAD / FBD	SCL	Description

	`ret_val := LOG2GEO(`
	`    laddr:=_word_in_,`
	
	`GEOADDR:=_variant_in_out_);`
		You use the LOG2GEO instruction to determine the module slot belonging to a hardware identifier.

The LOG2GEO instruction determines the geographic address of a logical address based upon the hardware identifier:

- Use the LADDR parameter to select the logical address based upon the hardware identifier.
- The GEOADDR contains the geographic address of the logical address given at the LADDR input.

Note

In the cases where the HW type does not support a component, a subslot number for a module 0 is returned.

An error is provided if the LADDR input does not address a HW object.

Table 9- 185 Data types for the parameters

Parameter and type		Data type	Description
LADDR	IN	HW_ANY	Hardware identifier of the IO system or the module. The number is assigned automatically and is stored in the properties of the CPU or the interface of the hardware configuration.
RET_VAL	OUT	Int	Error code of the instruction
GEOADDR	IN_OUT	Variant	Pointer to the GEOADDR system data type. The GEOADDR system data type contains the slot information. Refer to the "GEOADDR system data type (Page 453)" for further information.

For further information on valid data types, refer to "Overview of the valid data types" in the STEP 7 online help.

Table 9- 186 Condition codes

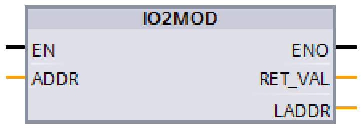
RET_VAL (W#16#...)	Description
0000	No error
8090	The address specified at the LADDR parameter is invalid.
* The error codes can be displayed as integer or hexadecimal values in the program editor.	

9.9.3 IO2MOD (Determine the hardware identifier from an I/O address) instruction

You use the IO2MOD instruction to determine the hardware identifier of the module from an I/O address of a (sub)module.

Table 9- 187 IO2MOD instruction

LAD / FBD	SCL	Description

	`ret_val := IO2MOD(`
	`    ADDR:=_word_in_,`
	`    LADDR:=_word_out_);`
		You use the IO2MOD instruction to determine the module slot belonging to a hardware identifier.

The IO2MOD instruction determines the hardware identifier of the module from an IO address (I, Q, PI, PQ) of a (sub)module.

Enter the IO address at the ADDR parameter. If a series of IO addresses is used at this parameter, only the first address is evaluated to determine the hardware identifier. If the first address is correctly specified, the length for the address specification at the ADDR is of no significance. If an address area is used that encompasses several modules or non-used addresses, the hardware identifier of the first module can also be determined.

If no IO address of a (sub)module is specified at the ADDR parameter, the error code "8090" is output at the RET_VAL parameter.

Note

Input of IO address in SCL

You cannot program using the IO access ID "%QWx:P" in SCL. In this case, use the symbolic tag name or the absolute address in the process image.

Table 9- 188 Data types for the parameters

Parameter	Declaration	Data type	Memory area	Description
ADDR	IN or IN/OUT ?	Variant	I, Q, M, D, L	IO address (I, Q, PI, PQ) within a (sub)module. Make sure that slice access is not used for the parameter ADDR. If this is the case, incorrect values are output at the LADDR parameter.
RET_VAL	OUT or RETURN ?	Int	I, Q, M, D, L	Error code of the instruction.
LADDR	OUT	HW_IO	I, Q, M, D, L	Determined hardware identifier (logical address) of the IO (sub)module.

For further information on valid data types, refer to "Overview of the valid data types" in the STEP 7 online help.

Table 9- 189 Condition codes

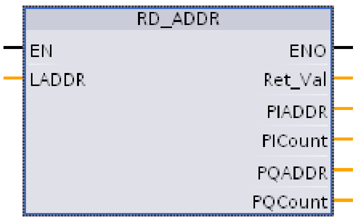
RET_VAL* (W#16#...)	Explanation
0	No error occurred.
8090	IO address specified at ADDR parameter is not used by any hardware component.
* The error codes can be displayed as integer or hexadecimal values in the program editor.	

9.9.4 RD_ADDR (Determine the IO addresses from the hardware identifier) instruction

You use the RD_ADDR instruction to get the I/O addresses of a submodule.

Table 9- 190 RD_ADDR instruction

LAD / FBD	SCL	Description

 RD_ADDR
EN
ENO
LADDR
Ret_Val
PIADDR
PICount
PQADDR
PQCount
	`ret_val := RD_ADDR(`
	`    laddr:=_word_in_,`
	`    PIADDR=>_udint_out_,`
	`    PICount=>_uint_out_,`
	`    PQADDR=>_udint_out_,`
	`    PQCount=>_uint_out_);`
		You use the RD_ADDR instruction to get the I/O addresses of a submodule.

The RD_ADDR instruction determines the length and the start address of the inputs or outputs based on the hardware identifier of a submodule:

- Use the LADDR parameter to select the input or output module based upon the hardware identifier.
- The following output parameters are used depending on whether it is an input module or output module:
  - In the case of an input module, the determined values are output at the PIADDR and PICOUNT parameters.
  - In the case of an output module, the determined values are output at the PQADDR and PQCOUNT parameters.
- The PIADDR and PQADDR parameters each contain the start address of the I/O addresses of the module.
- The PICOUNT and PQCOUNT parameters each contain the number of bytes of the inputs or outputs (1 byte for 8 inputs/outputs, 2 bytes for 16 inputs/outputs).

Table 9- 191 Data types for the parameters

Parameter and type		Data type	Description
LADDR	IN	HW_IO	Hardware identifier of the (sub)module
RET_VAL	OUT	Int	Error code of the instruction
PIADDR	OUT	UDInt	Start address of the input module
PICOUNT	OUT	UInt	Number of bytes of the inputs
PQADDR	OUT	UDInt	Start address of the output module
PQCOUNT	OUT	UInt	Number of bytes of the outputs

For further information on valid data types, refer to "Overview of the valid data types" in the STEP 7 online help.

Table 9- 192 Condition codes

RET_VAL (W#16#...)	Description
0000	No error
8090	Hardware identifier of the module at the LADDR parameter is invalid.
* The error codes can be displayed as integer or hexadecimal values in the program editor.	

9.9.5 GEOADDR system data type

Geographical address

The system data type GEOADDR contains the geographical address of a module (or the slot information).

- Geographical address for PROFINET IO:

  For PROFINET IO, the geographical address is composed of the ID of the PROFINET IO system, the device number, the slot number, and the submodule (if a sub-module is used).
- Geographical address for PROFIBUS DP:

  For PROFIBUS DP, the geographical address consists of the ID of the DP master system, the station number, and the slot number.

The slot information of the modules can be found in the hardware configuration of each module.

Structure of the GEOADDR system data type

The structure GEOADDR is automatically created if you enter "GEOADDR" as the data type in a data block.

Parameter name		Data type	Description
GEOADDR		STRUCT	
	HWTYPE	UINT	Hardware type: • 1: IO system (PROFINET/PROFIBUS) • 2: IO device/DP slave • 3: Rack • 4: Module • 5: Submodule If a hardware type is not supported by the instruction, a HWTYPE "0" is output.
	AREA	UINT	Area ID: • 0 = CPU • 1 = PROFINET IO • 2 = PROFIBUS DP • 3 = AS-i
	IOSYSTEM	UINT	PROFINET IO system (0=central unit in the rack)
	STATION	UINT	• Number of the rack if area identifier AREA = 0 (central module). • Station number if area identifier AREA > 0.
	SLOT	UINT	Slot number
	SUBSLOT	UINT	Number of the submodule. This parameter has the value "0" if no submodule is available or can be plugged.

9.10 Common error codes for the "Extended" instructions

Table 9- 193 Common condition codes for the extended instructions

Condition code (W#16#....)[1]	Description
8x22[2]	Area too small for input
8x23	Area too small for output
8x24	Illegal input area
8x25	Illegal output area
8x28	Illegal input bit assignment
8x29	Illegal output bit assignment
8x30	Output area is a read-only DB.
8x3A	DB does not exist.

[1] If one of these errors occurs when a code block is executed, then the CPU remains in RUN (default) or can be configured to go to STOP. Optionally, you can use the GetError or GetErorID instructions within that code block to handle the error locally (CPU remains in RUN), and create a programmed reaction to the error.

[2] The "x" represents the parameter number with the error. Parameter numbers start with 1.

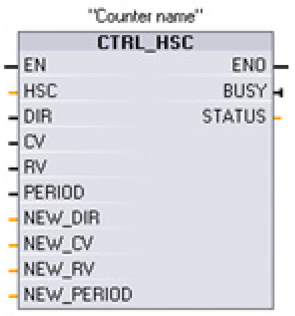
Technology instructions 10

10.1 High-speed counter

10.1.1 CTRL_HSC (Control high-speed counter) instruction

Table 10- 1 CTRL_HSC instruction (For general purpose counting)

LAD / FBD	SCL	Description

 "Counter name"
CTRL_HSC
EN
ENO
HSC
BUSY
DIR
STATUS
CV
RV
PERIOD
NEW_DIR
NEW_CV
NEW_RV
NEW_PERIOD
	`"CTRL_HSC_1_DB" (`
	`hsc:=W#16#0,`
	`dir:=False,`
	`cv:=False,`
	`rv:=False,`
	`period:=False,`
	`new_dir:=0,`
	`new_cv:=L#0,`
	`new_rv:=L#0,`
	`new_period:=0,`
	`busy=>_bool_out_,`
	`status=>_word_out_);`
		Each CTRL_HSC (Control high-speed counter) instruction uses a structure stored in a DB to maintain counter data. You assign the DB when the CTRL_HSC instruction is placed in the editor.

1 When you insert the instruction, STEP 7 displays the "Call Options" dialog for creating the associated DB.

2 In the SCL example, "CTRL_HSC_1_DB" is the name of the instance DB.

Table 10- 2 Data types for the parameters

Parameter and type		Data type	Description
HSC	IN	HW_HSC	HSC identifier
DIR[1, 2]	IN	Bool	1 = Request new direction
CV[1]	IN	Bool	1 = Request to set new counter value
RV[1]	IN	Bool	1= Request to set new reference value
PERIOD[1]	IN	Bool	1 = Request to set new period value (only for frequency measurement mode)
NEW_DIR	IN	Int	New direction: 1= forward, -1= backward
NEW_CV	IN	DInt	New counter value
NEW_RV	IN	DInt	New reference value
NEW_PERIOD	IN	Int	New period value in seconds(only for frequency measure-ment mode): 1= 1 s 2 = .1 s 3 = 0.1 s

Parameter and type		Data type	Description
BUSY[3]	OUT	Bool	Function is busy
STATUS	OUT	Word	Execution condition code

[1] If an update of a parameter value is not requested, then the corresponding input values are ignored.

[2] The DIR parameter is only valid if the configured counting direction is set to "User program (internal direction control)". You determine how to use this parameter in the HSC device configuration.

[3] For an HSC on the CPU or on the SB, the BUSY parameter always has a value of 0.

You configure the parameters for each HSC in the device configuration for the CPU for counting/frequency function, reset options, interrupt event configuration, hardware I/O, and count value address.

Some of the parameters for the HSC can be modified by your user program to provide program control of the counting process:

- Set the counting direction to a NEW_DIR value
- Set the current count value to a NEW_CV value
- Set the reference value to a NEW_RV value
- Set the period value (for frequency measurement mode) to a NEW_PERIOD value

If the following Boolean flag values are set to 1 when the CTRL_HSC instruction is executed, the corresponding NEW_xxx value is loaded to the counter. Multiple requests (more than one flag is set at the same time) are processed in a single execution of the CTRL_HSC instruction.

- DIR = 1 is a request to load a NEW_DIR value, 0 = no change
- CV = 1 is a request to load a NEW_CV value, 0 = no change
- RV = 1 is a request to load a NEW_RV value, 0 = no change
- PERIOD = 1 is a request to load a NEW_PERIOD value, 0 = no change

The CTRL_HSC instruction is typically placed in a hardware interrupt OB that is executed when the counter hardware interrupt event is triggered. For example, if a CV=RV event triggers the counter interrupt, then a hardware interrupt OB code block executes the CTRL_HSC instruction and can change the reference value by loading a NEW_RV value.

The current count value is not available in the CTRL_HSC parameters. The process image address that stores the current count value is assigned during the hardware configuration of the high-speed counter. You may use program logic to directly read the count value. The value returned to your program will be a correct count for the instant in which the counter was read. The counter will continue to count high-speed events. Therefore, the actual count value could change before your program completes a process using an old count value.

HSC current count value: Program access, value range, and rollover behavior

The CPU stores the current value of each HSC in an input (I) address. The following table shows the default addresses assigned to the current value for each HSC. You can change the I address for the current value by modifying the properties of the CPU in the Device Configuration.

High-speed counters use a DInt value to store the current count value. A DInt count value has a range of -2147483648 to +2147483647. The counter rolls over from the maximum positive value to the maximum negative value when counting up, and from the maximum negative value to the maximum positive value when counting down.

HSC	Current value data type	Default current value address
HSC1	DInt	ID1000
HSC2	DInt	ID1004
HSC3	DInt	ID1008
HSC4	DInt	ID1012
HSC5	DInt	ID1016
HSC6	DInt	ID1020

If an error occurs, ENO is set to 0 and the STATUS output indicates a condition code.

Table 10- 3 Execution condition codes

STATUS (W#16#)	Description
0	No error
80A1	HSC identifier does not address a HSC
80B1	Illegal value in NEW_DIR
80B2	Illegal value in NEW_CV
80B3	Illegal value in NEW_RV
80B4	Illegal value in NEW_PERIOD
80C0	Multiple access to the high-speed counter
80D0	High-speed counter (HSC) not enabled in CPU hardware configuration

10.1.2 CTRL_HSC_EXT (Control high-speed counter (extended)) instruction

The instruction CTRL_HSC_EXT provides program access to the number of input pulses to a designated HSC over a specified time period. This instruction allows the program to determine the period between input pulses with a fine nano-second resolution.

To use CTRL_HSC_EXT instruction, follow the steps below:

1. Configure the associated HSC for Period mode. Choose the desired Operating phase. If you select internal direction contro, you release the direction input for other uses.
2. Drop the CTRL_HSC_EXT into the ladder network, which also creates an instance datablock CTRL_HSC_EXT_DB.
3. Create a User Global_DB=ex: "MYDB", which is an input parameter to CTRL_HSC_EXT. This DB contains the information needed by the SFB.
4. Within MYDB, locate an empty row and add the variable Name=Ex: "My period".
5. Add the data type by typing in "HSC_Period" <enter> (the dropdown control does not currently contain this option). You must type this name exactly as shown.
6. Verify that the variable "MyPeriod" is now an expandable communication data structure.
7. Attach on ladder instruction CTRL_HSC_EXT: "CTRL" pin the DB variable "MYDB".MyPeriod.

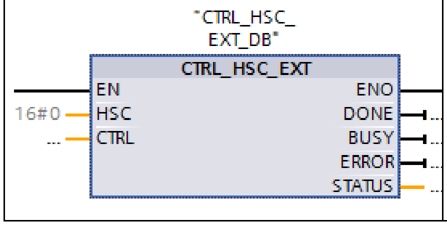
Note

Digital input filter time setting

For HSC digital inputs, use the smallest expected pulse width for the associated digital input filter setting.

Table 10- 4 CTRL_HSC_EXT instruction

LAD / FBD	SCL	Description

 "CTRL_HSC_
EXT_DB"
CTRL_HSC_EXT
EN
ENO
16#0
HSC
DONE
...
CTRL
BUSY
ERROR
STATUS
	`"CTRL_HSC_1_DB" (`
	`    hsc:=_hw_hsc_in_,`
	`    done:=_done_out_,`
	`    busy:=_busy_out_,`
	`    error:=_error_out_,`
	`    status:=_status_out_,`
	`    ctrl:=MYDB.MyPeriod);`
		Each CTRL_HSC_EXT (Control high-speed counter) instruction uses a system defined data structure stored in a user defined Global DB to store counter data. The HSC_Period data type is assigned as an input parameter to CTRL_HSC_EXT.

1 STEP 7 automatically creates the DB when you insert the instruction.

2 In the SCL example, "CTRL_HSC_1_DB" is the name of the instance DB.

Table 10- 5 Data types for the parameters

Parameter and type		Data type	Description
HSC	IN	HW_HSC	HSC identifier
CTRL	IN	HSC_Period	SFB input and return data (see table "HSC_Period structure")
DONE	OUT	Bool	1= Indicates SFB is finished. Always 1 because SFB is synchronous
BUSY	OUT	Bool	Always 0, function is never busy
ERROR	OUT	Bool	1 = Indicates an error
STATUS	OUT	Word	Execution condition code (see table "Execution condition codes")

Table 10- 6 HSC_Period structure

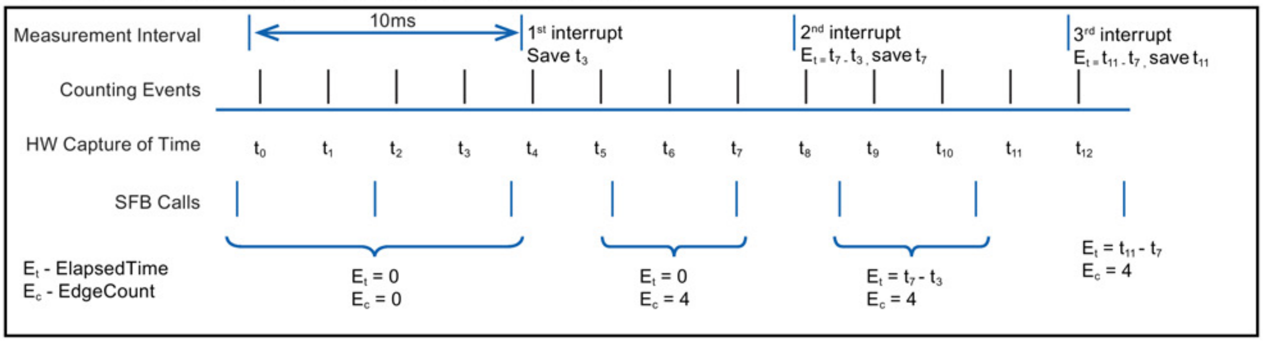
Structure element		Data type	Description
ElapsedTime	OUT	UDINT	Elapsed time between last input pulses of sequential intervals.
EdgeCount	OUT	UDINT	Returns the number of input pulses of the most recent completed period.
EnHSC	IN	Bool	Enables HSC: 1 = Enables SFB, 0 = Disables SFB
EnPeriod	IN	Bool	Enables updating the Period: 1 = Changes SFB period to NewPeriod
NewPeriod	IN	INT	NewPeriod specifies the period measurement interval (time taken to perform a period measurement). The only allowed values are 10, 100, or 1000 milliseconds.

HSC_Period structure element definitions

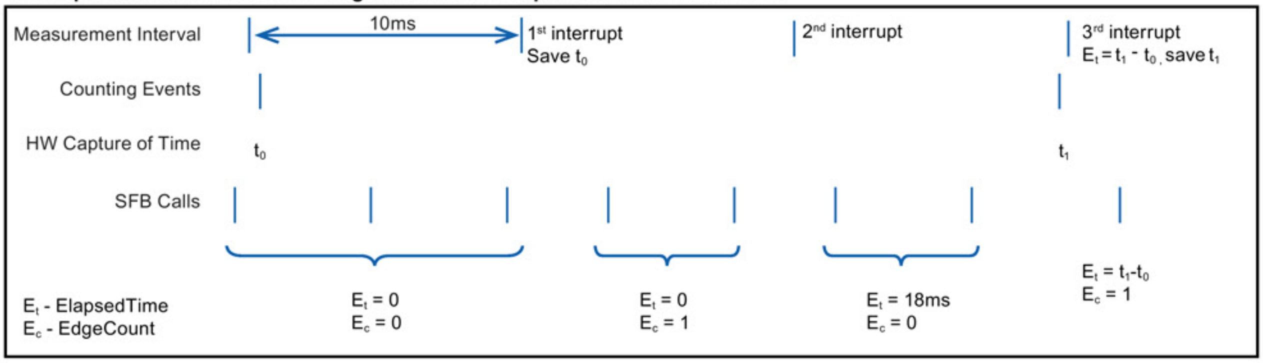
- ElapsedTime returns the elapsed time in nanoseconds between the last pulse in the current period measurement interval and the last pulse in the previous period measurement interval.
  If EdgeCount = 0, then the ElapsedTime is the cumulative time since the last pulse.
  ElapsedTime has a range from 0 to 4,294,967,280 nanoseconds (0x0000 0000 to 0xFFFF FFF0). Period overflow is indicated by the return value 4,294,96,295 (0xFFFF FFFF. The values from 0xFFFF FFF1 to 0xFFFF FFFE are reserved.
  If ElapsedTime is either 0 (no input pulses received) or 0xFFFF FFFF (Period overflow), then EdgeCount is not valid.
- EdgeCount returns the number of input pulses of the most recent measurement interval. The EdgeCount must be "1" or more in order to be able to calculate the period between pulses. The period is calculated using the forumula: Period = ElapsedTime/EdgeCount.

The following examples show the instruction makes period measurements.

Example 1: Multiple Counting Events in a Measurement Interval


Measurement Interval
10ms
1st interrupt
Save t_3
2nd interrupt
$E_t = t_7 - t_3$, save t_7
3rd interrupt
$E_t = t_{11} - t_7$, save t_{11}
Counting Events
HW Capture of Time
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12}
SFB Calls
E_t - ElapsedTime
E_c - EdgeCount
$E_t = 0$, $E_c = 0$
$E_t = 0$, $E_c = 4$
$E_t = t_7 - t_3$, $E_c = 4$
$E_t = t_{11} - t_7$, $E_c = 4$

Example 2: Zero and One Counting Events in a Multiple Measurement Interval



Rules:
1. If $E1_t = 0$, the period is invalid
2. Else, Period = E_t / E_c

The CPU stores the current value of each HSC in an input (I) address. The following table shows the default addresses assigned to the current value for each HSC. You can change the input (I) address for the current value by modifying the properties of the CPU in the Device Configuration.

High-speed counters use a DInt value to store the current count value. A DInt count value has a range of -2147483648 to +2147483647. The counter rolls over from the maximum positive value to the maximum negative value when counting up, and from the maximum negative value to the maximum positive value when counting down.

Table 10- 7 Default addresses assigned to current value of each HSC

HSC	Current value data type	Default current value address
HSC1	DInt	ID1000
HSC2	DInt	ID1004
HSC3	DInt	ID1008
HSC4	DInt	ID1012
HSC5	DInt	ID1016
HSC6	DInt	ID1020

If an error occurs, ENO is set to 0 and the STATUS output indicates the condition code.

Table 10- 8 Execution condition codes

STATUS (W#16#)	Description
0	No error
80A1	HSC identifier does not address an HSC
80D0	SBF 124 not available
80B5	Invalid value for NewPeriod

10.1.3 Operation of the high-speed counter

High-speed counters (HSC) can count events that occur faster than the cyclic OB execution rate. If the events to be counted occur slower than the execution rate of the OB, you can use CTU, CTD, or CTUD standard counter instructions. If the events occur faster than the OB execution rate, then use the faster HSC device. The CTRL_HSC instruction allows your program to programmatically change some of the HSC parameters.

For example: You can use the HSC as an input for 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 HSC.

The HSC is loaded with the first of several presets, and the outputs are activated for the time period where the current count is less than the current preset. The HSC provides an interrupt when the current count is equal to preset, when reset occurs, and also when there is a direction change.

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 interrupts occur at a much lower rate than the counting rate of the HSC, precise control of high-speed operations can be implemented with relatively minor impact to the scan cycle of the CPU. 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 channel selection

Use the following table and ensure that the CPU and SB input channels that you connect can support the maximum pulse rates in your process signals.

Note

CPU and SB input channels (V4 or later firmware) have configurable input filter times

Earlier firmware versions had fixed HSC input channels and fixed filter times that could not be changed.

V4 or later versions allow you to assign input channels and filter times. The default input filter setting of 6.4 ms may be too slow for your process signals. You must optimize the digital input filter times (Page 164) for the HSC inputs for your HSC application.

Table 10- 9 CPU input: maximum frequency

CPU	CPU Input channel	1 or 2 phase mode	A/B Quadrature phase mode
1211C	Ia.0 to Ia.5	100 kHz	80 kHz
1212C	Ia.0 to Ia.5	100 kHz	80 kHz
	Ia.6, Ia.7	30 kHz	20 kHz
1214C and 1215C	Ia.0 to Ia.5	100kHz	80kHz
	Ia.6 to Ib.5	30 kHz	20 kHz
1217C	Ia.0 to Ia.5	100 kHz	80 kHz
	Ia.6 to Ib.1	30 kHz	20 kHz
	Ib.2 to Ib.5 (.2+, .2- to .5+, .5-)	1 MHz	1 MHz

Table 10- 10 SB signal board input: maximum frequency (optional board)

SB signal board	SB input channel	1 or 2 phase mode	A/B Quadrature phase mode
SB 1221, 200 kHz	Ie.0 to Ie.3	200kHz	160 kHz
SB 1223, 200 kHz	Ie.0, Ie.1	200kHz	160 kHz
SB 1223	Ie.0, Ie.1	30 kHz	20 kHz

Selecting the functionality for the HSC

All HSCs function the same way for the same counter mode of operation. Counter mode, direction control, and initial direction are assigned in the CPU device configuration for HSC function properties.

There are four basic types of HSC:

- Single-phase counter with internal direction control
- Single-phase counter with external direction control
- Two-phase counter with 2 clock inputs
- A/B phase quadrature counter

You can use each HSC type with or without a reset input. When you activate the reset input (with some restrictions, see the following table), the current value is cleared and held clear until you deactivate the reset input.

- Frequency function: Some HSC modes allow the HSC to be configured (Type of counting) to report the frequency instead of a current count of pulses. Three different frequency measuring periods are available: 0.01, 0.1, or 1.0 seconds.

  The frequency measuring period determines how often the HSC calculates and reports a new frequency value. The reported frequency is an average value determined by the total number of counts in the last measuring period. If the frequency is rapidly changing, the reported value will be an intermediate between the highest and lowest frequency occurring during the measuring period. The frequency is always reported in Hertz (pulses per second) regardless of the frequency measuring period setting.
- Counter modes and inputs: The following table shows the inputs used for the clock, direction control, and reset functions associated with the HSC.
- Period measurement function: Period measurement is provided over the configured measurement interval (10ms, 100ms, or 1000ms). The HSC_Period SDT returns period measurements and provides the period measurements as two values: ElapsedTime and EdgeCount. HSC inputs ID1000 to ID1020 are not affected by period measurements:
  - ElapsedTime is an unsigned double integer value in nanoseconds representing the time from the first counting event to the last counting event in the measurement interval. If the EdgeCount = 0, then the ElapsedTime is the time since the last counting event in a prior interval. ElapsedTime has a range from 0 to 4,294,967,280 ns (0x0000 0000 to 0xFFFF FFF0). Overflow is indicated by the value 4,294,967,295 (0xFFFF FFFF). The values from 0xFFFF FFF1 to 0xFFFF FFFE are reserved.
  - EdgeCount is an unsigned double integer value representing the number of counting events in the measurement interval.

  The same input cannot be used for two different functions, but any input not being used by the present mode of its HSC can be used for another purpose. For example, if HSC1 is in a mode that uses two built-in inputs but does not use the third external reset input (default assignment at I0.3), then I0.3 can be used for edge interrupts or for HSC 2.

Table 10- 11 Counting modes for HSC

Type	Input 1	Input 2	Input 3	Function
Single-phase counter with internal direction control	Clock	-	-	Count or frequency
			Reset	Count
Single-phase counter with external direction control	Clock	Direction	-	Count or frequency
			Reset	Count
Two-phase counter with 2 clock inputs	Clock up	Clock down	-	Count or frequency
			Reset	Count
A/B-phase quadrature counter	Phase A	Phase B	-	Count or frequency
			Reset[1]	Count

[1] For an encoder: Phase Z, Home

Input addresses for the HSC

When you configure the CPU, you have the option to enable and configure the "Hardware inputs" for each HSC.

All HSC inputs must be connected to terminals on the CPU module or optional signal board that plugs into the front of the CPU module.

Note

As shown in the following tables, the default assignments for the optional signals for the different HSCs overlap. For example, the optional external reset for HSC 1 uses the same input as one of the inputs for HSC 2. For

For V4 CPUs or later, you can reassign the HSC inputs during the CPU configuration. You do not have to use the default input assignments.

Always ensure that you have configured your HSCs so that any one input is **not** being used by two HSCs.

The following tables show the HSC input default assignments for the on-board I/O of CPUs and an optional SB. (If the SB model selected has only 2 inputs, only 4.0 and 4.1 inputs are available.)

HSC input table definitions

- **Single-phase: C is Clock** input, **[d] is direction** input (optional), and **[R] is external reset** input (optional)
  (Reset is available only for "Counting" mode.)
- **Two-phase: CU is Clock Up** input, **CD is Clock Down** input, and **[R] is external reset** input.(optional)
  (Reset is available only for "Counting" mode.)
- **AB-phase quadrature: A is the Clock A** input, **B is the Clock B** input, and **[R] is external reset** input (optional). (Reset is available only for "Counting" mode.)

Table 10- 12 CPU 1211C: HSC default address assignments

HSC counter mode		**CPU on-board input (default 0.x)**						**Optional SB input (default 4.x)** [1]			
		0	**1**	**2**	**3**	**4**	**5**	**0**	**1**	**2**	**3**
HSC 1	1-phase	C	[d]		[R]			C	[d]		[R]
	2-phase	CU	CD		[R]			CU	CD		[R]
	AB-phase	A	B		[R]			A	B		[R]
HSC 2	1-phase		[R]	C	[d]				[R]	C	[d]
	2-phase		[R]	CU	CD				[R]	CU	CD
	AB-phase		[R]	A	B				[R]	A	B
HSC 3	1-phase					C	[d]	C	[d]		R]
	2-phase										
	AB-phase										
HSC4	1-phase					C	[d]	C	[d]		R]
	2-phase					CU	CD				
	AB-phase					A	B				
HSC 5	1-phase							C	[d]		[R]
	2-phase							CU	CD		[R]
	AB-phase							A	B		[R]
HSC 6	1-phase								[R]	C	[d]
	2-phase								[R]	CU	CD
	AB-phase								[R]	A	B

[1] An SB with only 2 digital inputs provides only the 4.0 and 4.1 inputs.

Table 10- 13 CPU 1212C: HSC default address assignments

HSC counter mode		**CPU on-board input (default 0.x)**								**Optional SB input (default 4.x)** [1]			
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**0**	**1**	**2**	**3**
HSC 1	1-phase	C	[d]		[R]					C	[d]		[R]
	2-phase	CU	CD		[R]					CU	CD		[R]
	AB-phase	A	B		[R]					A	B		[R]
HSC 2	1-phase		[R]	C	[d]						[R]	C	[d]
	2-phase		[R]	CU	CD						[R]	CU	CD
	AB-phase		[R]	A	B						[R]	A	B
HSC 3	1-phase					C	[d]		[R]	C	[d]		[R]
	2-phase					CU	CD		[R]				
	AB-phase					A	B		[R]				
HSC 4	1-phase						[R]	C	[d]	C	[d]		[R]
	2-phase						[R]	CU	CD				
	AB-phase						[R]	A	B				

HSC counter mode		CPU on-board input (default 0.x)								Optional SB input (default 4.x) [1]			
		0	1	2	3	4	5	6	7	0	1	2	3
HSC 5	1-phase									C	[d]		[R]
	2-phase									CU	CD		[R]
	AB-phase									A	B		[R]
HSC 6	1-phase										[R]	C	[d]
	2-phase										[R]	CU	CD
	AB-phase										[R]	A	B

1 An SB with only 2 digital inputs provides only the 4.0 and 4.1 inputs.

Table 10- 14 CPU 1214C, CPU 1215C, and CPU1217C:
HSC default address assignments
(on-board inputs only, see next table for optional SB addresses)

HSC counter mode		Digital input byte 0 (default: 0.x)								Digital input byte 1 (default: 1.x)					
		0	1	2	3	4	5	6	7	0	1	2	3	4	5
HSC 1	1-phase	C	[d]		[R]										
	2-phase	CU	CD		[R]										
	AB-phase	A	B		[R]										
HSC 2	1-phase		[R]	C	[d]										
	2-phase		[R]	CU	CD										
	AB-phase		[R]	A	B										
HSC 3	1-phase					C	[d]		[R]						
	2-phase					CU	CD		[R]						
	AB-phase					A	B		[R]						
HSC 4	1-phase						[R]	C	[d]						
	2-phase						[R]	CU	CD						
	AB-phase						[R]	A	B						
HSC 5	1-phase									C	[d]	[R]			
	2-phase									CU	CD	[R]			
	AB-phase									A	B	[R]			
HSC 6	1-phase												C	[d]	[R]
	2-phase												CU	CD	[R]
	AB-phase												A	B	[R]

Table 10- 15 Optional SB in CPUs in above table: HSC default address assignments

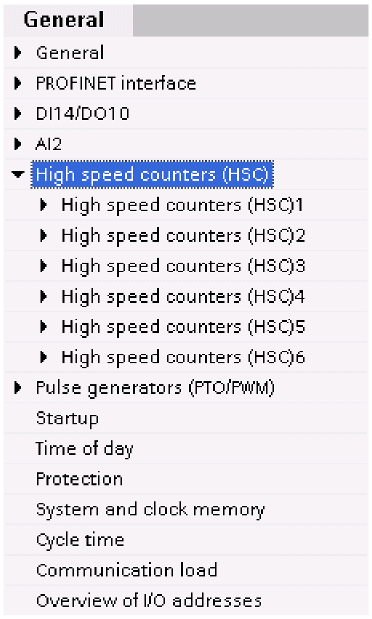
HSC		**Optional SB inputs (default: 4.x)** [1]			
		0	**1**	**2**	**3**
HSC 1	1-phase	C	[d]		[R]
	2-phase	CU	CD		[R]
	AB-phase	A	B		[R]
HSC 2	1-phase		[R]	C	[d]
	2-phase		[R]	CU	CD
	AB-phase		[R]	A	B
HSC 5	1-phase	C	[d]		[R]
	2-phase	CU	CD		[R]
	AB-phase	A	B		[R]
HSC 6	1-phase		[R]	C	[d]
	2-phase		[R]	CU	CD
	AB-phase		[R]	A	B

[1] An SB with only 2 digital inputs provides only the 4.0 and 4.1 inputs.

Note

The digital I/O points used by high-speed counter devices are assigned during CPU device configuration. When digital I/O point addresses are assigned to HSC devices, the values of the assigned I/O point addresses cannot be modified by the force function in a watch table.

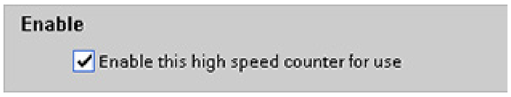
10.1.4 Configuration of the HSC


General
General
PROFINET interface
DI14/DO10
AI2
High speed counters (HSC)
High speed counters (HSC)1
High speed counters (HSC)2
High speed counters (HSC)3
High speed counters (HSC)4
High speed counters (HSC)5
High speed counters (HSC)6
Pulse generators (PTO/PWM)
Startup
Time of day
Protection
System and clock memory
Cycle time
Communication load
Overview of I/O addresses

You may configure up to 6 high-speed counters. Edit the CPU device configuration and assign the HSC properties of each individual HSC.

Enable an HSC by selecting the "Enable" option for that HSC

Use the CTRL_HSC and/or CTRL_HSC_EXT instructions in your user program to control the operation of the HSC.



Enable

✓ Enable this high speed counter for use

WARNING

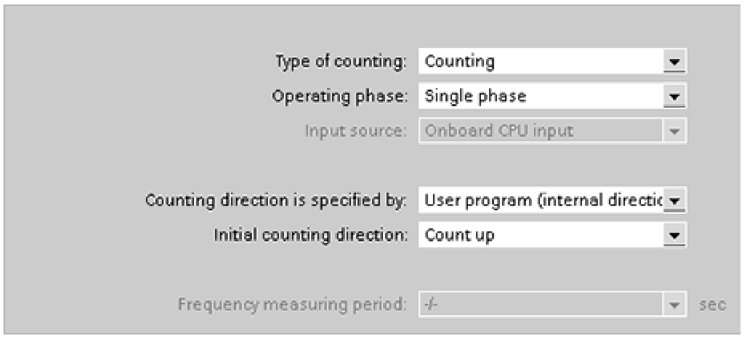
Risks with changes to filter time setting for digital input channels

If the filter time for a digital input channel is changed from a previous setting, a new "0" level input value might need to be presented for up to 20.0 ms accumulated duration before the filter becomes fully responsive to new inputs. During this time, short "0" pulse events of duration less than 20.0 ms may not be detected or counted.

This changing of filter times can result in unexpected machine or process operation, which can cause death or serious injury to personnel, and/or damage to equipment.

To ensure that a new filter time goes immediately into effect, power cycle the CPU.

After enabling the HSC, configure the other parameters, such as counter function, initial values, reset options and interrupt events.



Type of counting: Counting
Operating phase: Single phase
Input source: Onboard CPU input

Counting direction is specified by: User program (internal directic
Initial counting direction: Count up

Frequency measuring period: -/- sec

For additional information about configuring the HSC, refer to the section on configuring the CPU (Page 162).

10.2 PID control

STEP 7 provides the following PID instructions for the S7-1200 CPU:

- The PID_Compact instruction is used to control technical processes with continuous input- and output variables.
- The PID_3Step instruction is used to control motor-actuated devices, such as valves that require discrete signals for open- and close actuation.
- The PID_Temp instruction provides a universal PID controller that allows handling of the specific requirements of temperature control.

Note

Changes that you make to the PID configuration and download in RUN do not take effect until the CPU transitions from STOP to RUN mode. Changes that you make in the "PID parameters" dialog using the "Start value control" take effect immediately.

All three PID instructions (PID_Compact, PID_3Step, and PID_Temp) can calculate the P-, I-, and D-components during startup (if configured for "pre-tuning"). You can also configure the instruction for "fine tuning" to allow you to optimize the parameters. You do not need to manually determine the parameters.

Note

Execute the PID instruction at constant intervals of the sampling time (preferably in a cyclic OB).

Because the PID loop needs a certain time to respond to changes of the control value, do not calculate the output value in every cycle. Do not execute the PID instruction in the main program cycle OB (such as OB 1).

The sampling time of the PID algorithm represents the time between two calculations of the output value (control value). The output value is calculated during self-tuning and rounded to a multiple of the cycle time. All other functions of PID instruction are executed at every call.

PID algorithm

The PID (Proportional/Integral/Derivative) controller measures the time interval between two calls and then evaluates the results for monitoring the sampling time. A mean value of the sampling time is generated at each mode changeover and during initial startup. This value is used as reference for the monitoring function and is used for calculation. Monitoring includes the current measuring time between two calls and the mean value of the defined controller sampling time.

The output value for the PID controller consists of three components:

- P (proportional): When calculated with the "P" component, the output value is proportional to the difference between the setpoint and the process value (input value).
- I (integral): When calculated with the "I" component, the output value increases in proportion to the duration of the difference between the setpoint and the process value (input value) to finally correct the difference.
- D (derivative): When calculated with the "D" component, the output value increases as a function of the increasing rate of change of the difference between the setpoint and the process value (input value). The output value is corrected to the setpoint as quickly as possible.

The PID controller uses the following formula to calculate the output value for the PID_Compact instruction.

$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s}(w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1}(c \cdot w - x) \right]$$

y	Output value	x	Process value
w	Setpoint value	s	Laplace operator
K_p	Proportional gain (P component)	a	Derivative delay coefficient (D component)
T_1	Integral action time (I component)	b	Proportional action weighting (P component)
T_D	Derivative action time (D component)	c	Derivative action weighting (D component)

The PID controller uses the following formula to calculate the output value for the PID_3Step instruction.

$$\Delta y = K_p \cdot s \cdot \left[(b \cdot w - x) + \frac{1}{T_I \cdot s}(w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1}(c \cdot w - x) \right]$$

y	Output value	x	Process value
w	Setpoint value	s	Laplace operator
K_p	Proportional gain (P component)	a	Derivative delay coefficient (D component)
T_1	Integral action time (I component)	b	Proportional action weighting (P component)
T_D	Derivative action time (D component)	c	Derivative action weighting (D component)

10.2.1 Inserting the PID instruction and technology object

STEP 7 provides two instructions for PID control:

- The PID_Compact instruction and its associated technology object provide a universal PID controller with tuning. The technology object contains all of the settings for the control loop.
- The PID_3Step instruction and its associated technology object provide a PID controller with specific settings for motor-activated valves. The technology object contains all of the settings for the control loop. The PID_3Step controller provides two additional Boolean outputs.

After creating the technology object, you must configure the parameters (Page 505). You also adjust the autotuning parameters ("pre-tuning" during startup or manual "fine tuning") to commission the operation of the PID controller (Page 522).

Table 10- 16 Inserting the PID instruction and the technology object

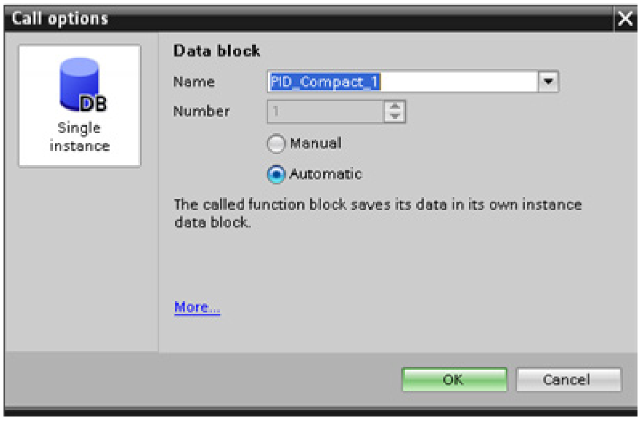
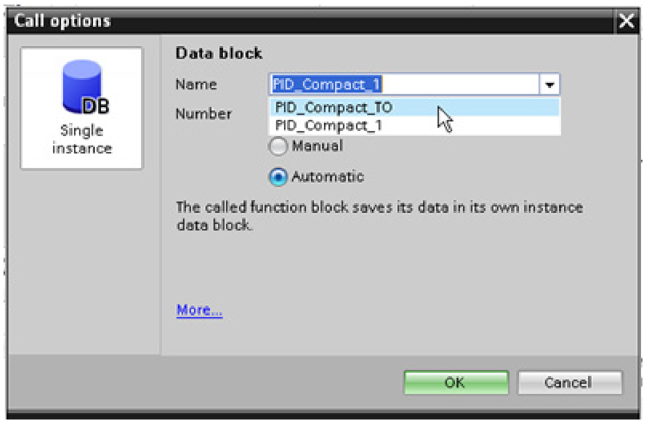
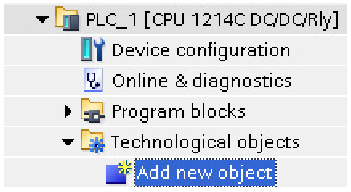
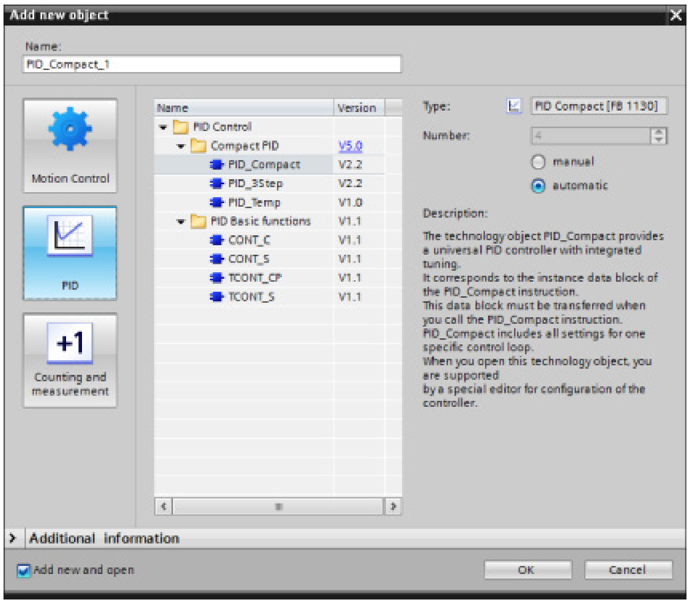
When you insert a PID instruction into your user program, STEP 7 automatically creates a technology object and an instance DB for the instruction. The instance DB contains all of the parameters that are used by the PID instruction. Each PID instruction must have its own unique instance DB to operate properly. After inserting the PID instruction and creating the technology object and instance DB, you configure the parameters for the technology object (Page 505).	 Call options Data block Name: PID_Compact_1 Number: 1 Manual [✓ Automatic] The called function block saves its data in its own instance data block.

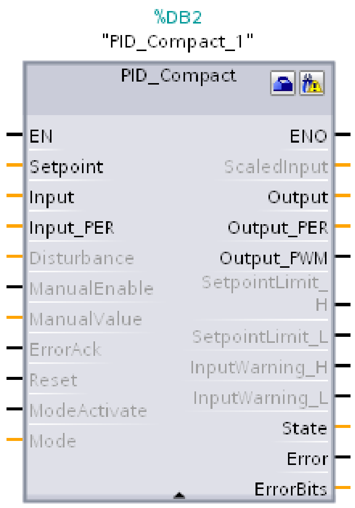
Table 10- 17 (Optional) Creating a technology object from the project navigator

You can also create technology objects for your project **before** inserting the PID instruction. By creating the technology object before inserting a PID instruction into your user program, you can then select the technology object when you insert the PID instruction.	 Call options Data block Name: PID_Compact_1 PID_Compact_TO PID_Compact_1 Number Manual [✓ Automatic] The called function block saves its data in its own instance data block. More...
To create a technology object, double-click the "Add new object" icon in the project navigator.	 PLC_1 [CPU 1214C DC/DC/Rly] Device configuration Online & diagnostics Program blocks Technological objects Add new object
Click the "Control" icon and select the technology object for the type of PID controller (PID_Compact or PID_3Step). You can create an optional name for the technology object. Click "OK" to create the technology object.	 Add new object Name: PID_Compact_1 Name / Version: PID Control; Compact PID V5.0; PID_Compact V2.2; PID_3Step V2.2; PID_Temp V1.0; PID Basic functions V1.1; CONT_C V1.1; CONT_S V1.1; TCONT_CP V1.1; TCONT_S V1.1 Type: PID Compact [FB 1130] Number: 4 manual [✓ automatic] Description: The technology object PID_Compact provides a universal PID controller with integrated tuning. It corresponds to the instance data block of the PID_Compact instruction. This data block must be transferred when you call the PID_Compact instruction. PID_Compact includes all settings for one specific control loop. When you open this technology object, you are supported by a special editor for configuration of the controller. Additional information Add new and open

10.2.2 PID_Compact instruction

The PID_Compact instruction provides a universal PID controller with integrated self-tuning for automatic and manual mode.

Table 10- 18 PID_Compact instruction

LAD / FBD	SCL	Description
 %DB2 "PID_Compact_1" PID_Compact EN ENO Setpoint ScaledInput Input Output Input_PER Output_PER Disturbance Output_PWM ManualEnable SetpointLimit_H ManualValue SetpointLimit_L ErrorAck InputWarning_H Reset InputWarning_L ModeActivate State Mode Error ErrorBits	`"PID_Compact_1"(Setpoint:=_real_in_, Input:=_real_in_, Input_PER:=_word_in_, Disturbance:=_real_in_, ManualEnable:=_bool_in_, ManualValue:=_real_in_, ErrorAck:=_bool_in_, Reset:=_bool_in_, ModeActivate:=_bool_in_, Mode:=_int_in_, ScaledInput=>_real_out_, Output=>_real_out_, Output_PER=>_word_out_, Output_PWM=>_bool_out_, SetpointLimit_H=>_bool_out_, SetpointLimit_L=>_bool_out_, InputWarning_H=>_bool_out_, InputWarning_L=>_bool_out_, State=>_int_out_, Error=>_bool_out_, ErrorBits=>_dword_out_);`	PID_Compact provides a PID controller with self-tuning for automatic and manual mode. PID_Compact is a PID T1 controller with anti-windup and weighting of the P- and D-component.

1 STEP 7 automatically creates the technology object and instance DB when you insert the instruction. The instance DB contains the parameters of the technology object.

2 In the SCL example, "PID_Compact_1" is the name of the instance DB.

Table 10- 19 Data types for the parameters

Parameter and type		Data type	Description
Setpoint	IN	Real	Setpoint of the PID controller in automatic mode. (Default value: 0.0)
Input	IN	Real	A tag of the user program is used as the source of the process value. (Default value: 0.0) If you are using the Input parameter, you must set Config.InputPerOn = FALSE.
Input_PER	IN	Word	An Analog input is used as the source of the process value. (Default value: W#16#0) If you are using the Input_PER parameter, you must set Config.InputPerOn = TRUE.
Disturbance	IN	Real	Disturbance variable or pre-control value
ManualEnable	IN	Bool	Enables or disables the manual operation mode. (Default value: FALSE): • A FALSE to TRUE edge activates "manual mode", while State = 4, Mode remains unchanged. As long as ManualEnable = TRUE, you cannot change the operating mode using a rising edge at ModeActivate or use the commissioning dialog. • A TRUE to FALSE edge activates the operating mode that is assigned by Mode. Note: We recommend that you change the operating mode using ModeActivate only.
ManualValue	IN	Real	Output value for manual operation. (Default value: 0.0) You can use values from Config.OutputLowerLimit to Config.OutputUpperLimit.
ErrorAck	IN	Bool	Resets the ErrorBits and warning outputs. FALSE to TRUE edge
Reset	IN	Bool	Restarts the controller. (Default value: FALSE): • FALSE to TRUE edge: – Switches to "inactive" mode – Resets the ErrorBits and warning outputs – Clears Integral action – Maintains PID parameters • As long as Reset = TRUE, PID_Compact remains in "Inactive" mode (State = 0). • TRUE to FALSE edge: – PID_Compact switches to the operating mode that is saved in the Mode parameter.
ModeActivate	IN	Bool	The PID_Compact switches to the operating mode that is saved in the Mode parameter. FALSE to TRUE edge:
Mode	IN	Int	The desired PID mode; Activated on the leading edge of the Mode Activate input.
ScaledInput	OUT	Real	Scaled process value. (Default value: 0.0)
Output[1]	OUT	Real	Output value in REAL format. (Default value: 0.0)
Output_PER[1]	OUT	Word	Analog output value. (Default value: W#16#0)

Parameter and type		Data type	Description
Output_PWM[1]	OUT	Bool	Output value for pulse width modulation. (Default value: FALSE) On and Off times form the output value.
SetpointLimit_H	OUT	Bool	Setpoint high limit. (Default value: FALSE) If SetpointLimit_H = TRUE, the absolute setpoint upper limit is reached (Setpoint ≥ Config.SetpointUpperLimit). The setpoint is limited to Config.SetpointUpperLimit.
SetpointLimit_L	OUT	Bool	Setpoint low limit. (Default value: FALSE) If SetpointLimit_L = TRUE, the absolute setpoint lower limit is reached (Setpoint ≤ Config.SetpointLowerLimit). The setpoint is limited to Config.SetpointLowerLimit.
InputWarning_H	OUT	Bool	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit. (Default value: FALSE)
InputWarning_L	OUT	Bool	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit. (Default value: FALSE)
State	OUT	Int	Current operating mode of the PID controller. (Default value: 0) You can change the operating mode using the Mode input parameter and a rising edge at ModeActivate: • State = 0: Inactive • State = 1: Pre-tuning • State = 2: Manual fine tuning • State = 3: Automatic mode • State = 4: Manual mode • State = 5: Substitute output value with error monitoring
Error	OUT	Bool	If Error = TRUE, at least one error message is pending in this cycle. (Default value: FALSE) Note: The Error parameter in V1.x PID was the ErrorBits field that contained the error codes. It is now a Boolean flag indicating that an error has occurred.
ErrorBits	OUT	DWord	The PID_Compact instruction ErrorBits parameters table (Page 480) defines the error messages that are pending. (Default value: DW#16#0000 (no error)). ErrorBits is retentive and is reset upon a rising edge at Reset or ErrorAck. Note: In V1.x, the ErrorBits parameter was defined as the Error parameter and did not exist.

[1] You can use the outputs of the Output, Output_PER, and Output_PWM parameters in parallel.

Operation of the PID_Compact controller

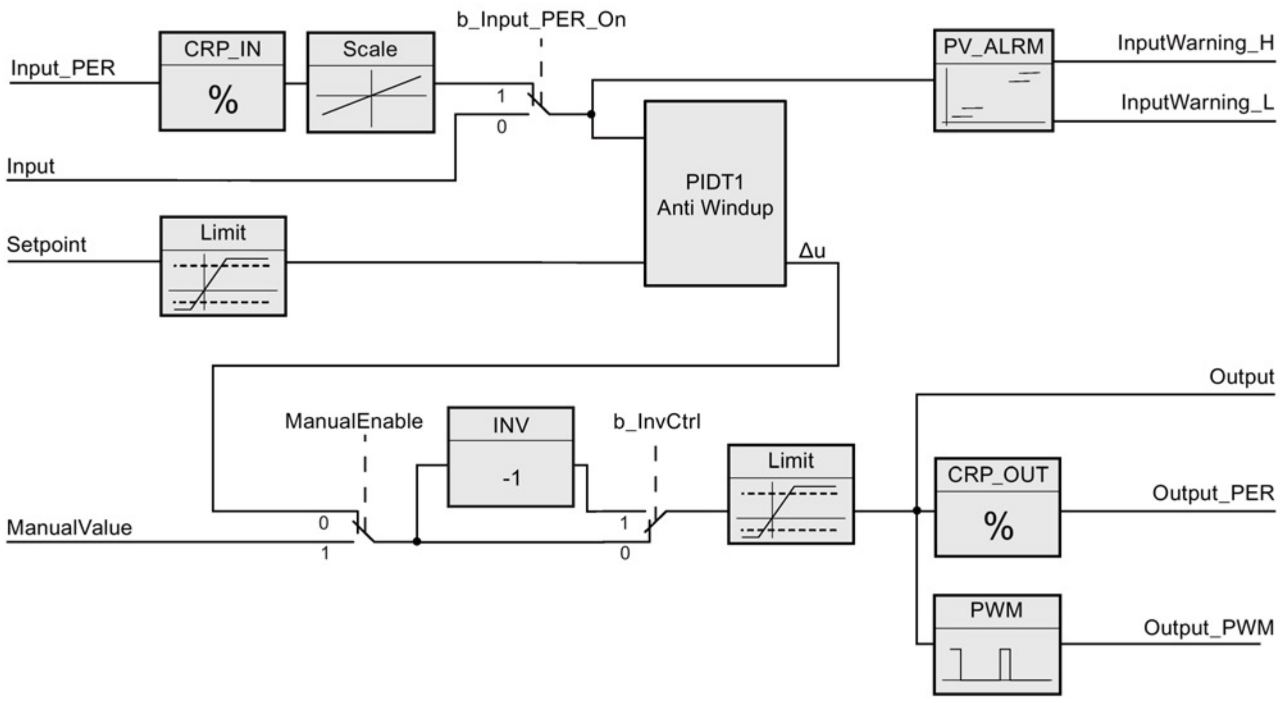


Figure 10-1 Operation of the PID_Compact controller

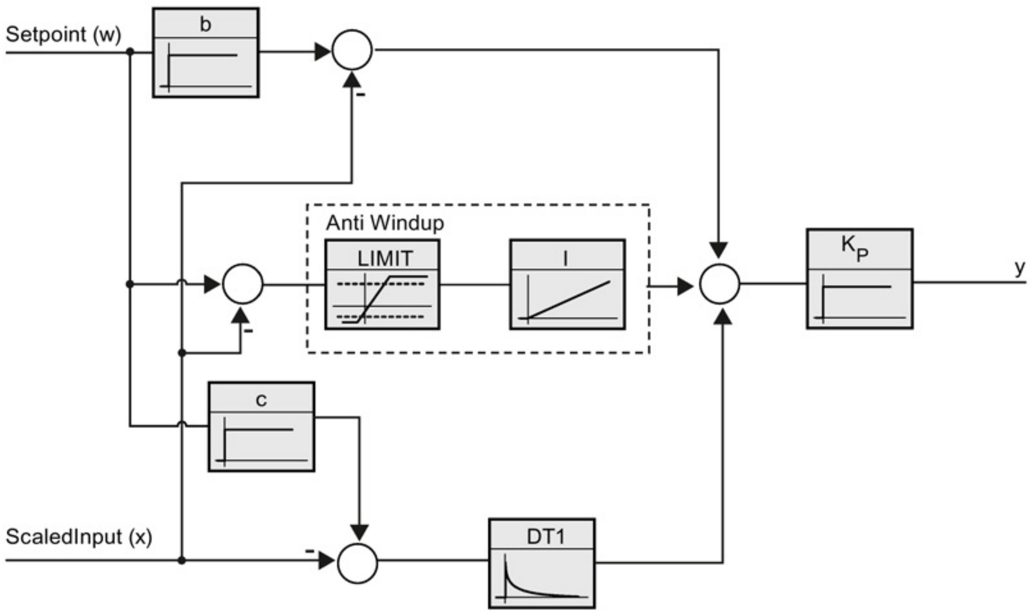


Figure 10-2 Operation of the PID_Compact controller as a PIDT1 controller with anti-windup

10.2.3 PID_Compact instruction ErrorBit parameters

If several errors are pending, the values of the error codes are displayed by means of binary addition. The display of error code 0003, for example, indicates that the errors 0001 and 0002 are also pending.

Table 10- 20 PID_Compact instruction ErrorBit parameters

ErrorBit (DW#16#...)	Description
0000	No error
0001 [1, 2]	The Input parameter is outside the process value limits. Input > Config.InputUpperLimit Input < Config.InputLowerLimit
0002 [2, 3]	Invalid value at the Input_PER parameter. Check whether an error is pending at the analog input.
0004 [4]	Error during fine tuning. Oscillation of the process value could not be maintained.
0008 [4]	Error at start of pre-tuning. The process value is too close to the setpoint. Start fine tuning.
0010 [4]	The setpoint was changed during tuning. Note: You can set the permitted fluctuation on the setpoint at the CancelTuningLevel tag.
0020	Pre-tuning is not permitted during fine tuning. Note: If ActivateRecoverMode = TRUE before the error occurred, PID_Compact remains in fine tuning mode.
0080 [4]	Error during pre-tuning. Incorrect configuration of output value limits. Check whether the limits of the output value are configured correctly and match the control logic.
0100 [4]	Error during fine tuning resulted in invalid parameters.
0200 [2, 3]	Invalid value at the Input parameter: Value has an invalid number format.
0400 [2, 3]	Calculation of the output value failed. Check the PID parameters.
0800 [1, 2]	Sampling time error: PID_Compact is not called within the sampling time of the cyclic interrupt OB.
1000 [2, 3]	Invalid value at the Setpoint parameter: Value has an invalid number format.
10000	Invalid value at the ManualValue parameter: Value has an invalid number format. Note: If ActivateRecoverMode = TRUE before the error occurred, PID_Compact uses SubstituteOutput as the output value. As soon as you assign a valid value in the ManualValue parameter, PID_Compact uses it as the output value.

ErrorBit (DW#16#...)	Description
20000	Invalid value at the SubstituteValue tag: Value has an invalid number format. PID_Compact uses the output value low limit as the output value. Note: If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_Compact switches back to automatic mode.
40000	Invalid value at the Disturbance parameter: Value has an invalid number format. Note: If automatic mode was active and ActivateRecoverMode = FALSE before the error occurred, Disturbance is set to zero. PID_Compact remains in automatic mode. Note: If pre-tuning or fine tuning mode was active and ActivateRecoverMode = TRUE before the error occurred, PID_Compact switches to the operating mode that is saved in the Mode parameter. If Disturbance in the current phase has no effect on the output value, tuning is not canceled.

1 Note: If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact remains in automatic mode.

2 Note: If pre-tuning or fine tuning mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact switches to the operating mode that is saved in the Mode parameter.

3 Note: If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact outputs the configured substitute output value. As soon as the error is no longer pending, PID_Compact switches back to automatic mode.

4 Note: If ActivateRecoverMode = TRUE before the error occurred, PID_Compact cancels the tuning and switches to the operating mode that is saved in the Mode parameter.

10.2.4 PID_Compact instruction Warning parameters

If the PID controller has several warnings pending, it displays the values of the error codes by means of binary addition. The display of error code 0003, for example, indicates that the errors 0001 and 0002 are pending.

Table 10- 21 PID_Compact instruction Warning parameters

Warning (DW#16#...)	Description
0000	No warning pending.
0001 [1]	The point of inflection was not found during pretuning.
0002	Oscillation was enforced during “tuning in run”. (The "Warning" parameter suppresses this warning and is only visible in the “WarningInternal” parameter for diagnostic purposes.)
0004 [1]	The setpoint was limited to the configured limits.
0008 [1]	Not all the necessary controlled system properties were defined for the selected method of calculation. Instead, the PID parameters were calculated using the TIR.TuneRuleHeat / TIR.TuneRuleCool = 3 method.
0010	The operating mode could not be changed because Reset = TRUE or ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040 [1]	The process value exceeded one of its warning limits.
0080	Invalid value at Mode. The operating mode is not switched.
0100 [1]	The manual value was limited to the limits of the controller output.
0200	The specified rule for tuning is not supported. No PID parameters are calculated.
1000	The substitute output value cannot be reached because it is outside the output value limits.

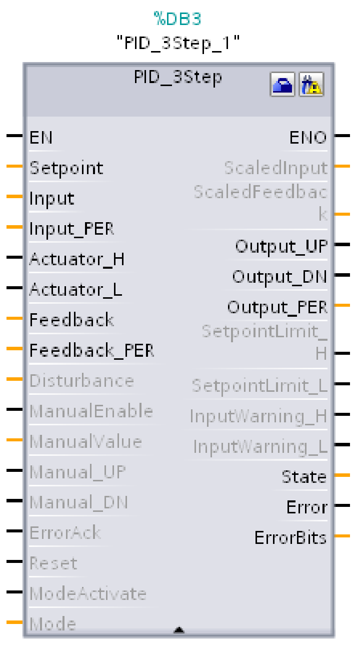
[1] Note: The PID controller deleted the following warnings automatically as soon as the cause is eliminated or the user action repeated with valid parameters: 0001, 0004, 0008, 0040, and 0100.

10.2.5 PID_3Step instruction

The PID_3Step instruction configures a PID controller with self-tuning capabilities that has been optimized for motor-controlled valves and actuators.

Table 10- 22 PID_3Step instruction

LAD / FBD	SCL	Description

 %DB3
"PID_3Step_1"
PID_3Step
EN
Setpoint
Input
Input_PER
Actuator_H
Actuator_L
Feedback
Feedback_PER
Disturbance
ManualEnable
ManualValue
Manual_UP
Manual_DN
ErrorAck
Reset
ModeActivate
Mode
ENO
ScaledInput
ScaledFeedback
Output_UP
Output_DN
Output_PER
SetpointLimit_H
SetpointLimit_L
InputWarning_H
InputWarning_L
State
Error
ErrorBits
	`"PID_3Step_1"(`
	`SetpoInt:=_real_in_,`
	`Input:=_real_in_,`
	`ManualValue:=_real_in_,`
	`Feedback:=_real_in_,`
	`InputPer:=_word_in_,`
	`FeedbackPer:=_word_in_,`
	`Disturbance:=_real_in_,`
	`ManualEnable:=_bool_in_,`
	`ManualUP:=_bool_in_,`
	`ManualDN:=_bool_in_,`
	`ActuatorH:=_bool_in_,`
	`ActuatorL:=_bool_in_,`
	`ErrorAck:=_bool_in_,`
	`Reset:=_bool_in_,`
	`ModeActivate:=_bool_in_,`
	`Mode:=_int_in_,`
	`ScaledInput=>_real_out_,`
	`ScaledFeedback=>_real_out_,`
	`ErrorBits=>_dword_out_,`
	`OutputPer=>_word_out_,`
	`State=>_int_out_,`
	`OutputUP=>_bool_out_,`
	`OutputDN=>_bool_out_,`
	`SetpoIntLimitH=>_bool_out_,`
	`SetpoIntLimitL=>_bool_out_,`
	`InputWarningH=>_bool_out_,`
	`InputWarningL=>_bool_out_,`
	`Error=>_bool_out_,`
	`ErrorBits=>_dword_out_);`
		PID_3Step configures a PID controller with self-tuning capabilities that has been optimized for motor-controlled valves and actuators. It provides two Boolean outputs.
		PID_3Step is a PID T1controller with anti-windup and weighting of the P- and D-components.

1 STEP 7 automatically creates the technology object and instance DB when you insert the instruction. The instance DB contains the parameters of the technology object.

2 In the SCL example, "PID_3Step_1" is the name of the instance DB.

Table 10- 23 Data types for the parameters

Parameter and type		Data type	Description
Setpoint	IN	Real	Setpoint of the PID controller in automatic mode. (Default value: 0.0)
Input	IN	Real	A tag of the user program is used as the source of the process value. (Default value: 0.0) If you are using the Input parameter, you must set Config.InputPerOn = FALSE.
Input_PER	IN	Word	An Analog input is used as the source of the process value. (Default value: W#16#0) If you are using the Input_PER parameter, you must set Config.InputPerOn = TRUE.
Actuator_H	IN	Bool	Digital position feedback of the valve for the high end stop If Actuator_H = TRUE, the valve is at the high end stop and is no longer moved in this direction. (Default value: FALSE)
Actuator_L	IN	Bool	Digital position feedback of the valve for the low end stop If Actuator_L = TRUE, the valve is at the low end stop and is no longer moved in this direction. (Default value: FALSE)
Feedback	IN	Real	Position feedback of the valve. (Default value: 0.0) If you are using the Feedback parameter, you must set Config.FeedbackPerOn = FALSE.
Feedback_PER	IN	Int	Analog feedback of the valve position. (Default value: W#16#0) If you are using the Feedback_PER parameter, you must set Config.FeedbackPerOn = TRUE. Feedback_PER is scaled, based upon the following tags: • Config.FeedbackScaling.LowerPointIn • Config.FeedbackScaling.UpperPointIn • Config.FeedbackScaling.LowerPointOut • Config.FeedbackScaling.UpperPointOut
Disturbance	IN	Real	Disturbance variable or pre-control value
ManualEnable	IN	Bool	Enables or disables the manual operation mode. (Default value: FALSE): • A FALSE to TRUE edge activates "manual mode", while State = 4, Mode remains unchanged. As long as ManualEnable = TRUE, you cannot change the operating mode using a rising edge at ModeActivate or use the commissioning dialog. • A TRUE to FALSE edge activates the operating mode that is assigned by Mode. Note: We recommend that you change the operating mode using ModeActivate only.
ManualValue	IN	Real	Process value for manual operation. (Default value: 0.0) In manual mode, you specify the absolute position of the valve. ManualValue is evaluated only if you are using OutputPer, **or** if position feedback is available.

Parameter and type		Data type	Description
ManualUP	IN	Bool	• Manual_UP = TRUE: – The valve is opened even if you use Output_PER or a position feedback. The valve is no longer moved if the high end stop has been reached. – See also Config.VirtualActuatorLimit • Manual_UP = FALSE: – If you use Output_PER or a position feedback, the valve is moved to ManualValue. Otherwise, the valve is no longer moved. Note: If Manual_UP and Manual_DN are set to TRUE simultaneously, the valve is not moved.
ManualDN	IN	Bool	• Manual_DN = TRUE: – The valve is opened even if you use Output_PER or a position feedback. The valve is no longer moved if the high end stop has been reached. – See also Config.VirtualActuatorLimit • Manual_DN = FALSE: – If you use Output_PER or a position feedback, the valve is moved to ManualValue. Otherwise, the valve is no longer moved.
ErrorAck	IN	Bool	Resets the ErrorBits and warning outputs. FALSE to TRUE edge
Reset	IN	Bool	Restarts the controller. (Default value: FALSE): • FALSE to TRUE edge: – Switches to "inactive" mode – Resets the ErrorBits and warning outputs – Clears Integral action – Maintains PID parameters • As long as Reset = TRUE, PID_3Step remains in "Inactive" mode (State = 0). • TRUE to FALSE edge: – PID_3Step switches to the operating mode that is saved in the Mode parameter.
ModeActivate	IN	Bool	The PID_3Step switches to the mode that is saved in the Mode parameter. FALSE to TRUE edge:
Mode	IN	Int	The desired PID mode; Activated on the leading edge of the Mode Activate input.
ScaledInput	OUT	Real	Scaled process value
ScaledFeedback	OUT	Real	Scaled valve position feedback Note: For an actuator without position feedback, the position of the actuator indicated by ScaledFeedback is very imprecise. ScaledFeedback can only be used for rough estimation of the current position in this case.
Output_UP	OUT	Bool	Digital output value for opening the valve. (Default value: FALSE) If Config.OutputPerOn = FALSE, the parameter Output_UP is used.

Parameter and type		Data type	Description
Output_DN	OUT	Bool	Digital output value for closing the valve. (Default value: FALSE) If Config.OutputPerOn = FALSE, the parameter Output_DN is used.
Output_PER	OUT	Word	Analog output value. If Config.OutputPerOn = TRUE, the parameter Output_PER is used.
SetpointLimitH	OUT	Bool	Setpoint high limit. (Default value: FALSE) If SetpointLimitH = TRUE, the absolute upper limit of the setpoint is reached (Setpoint ≥ Config.SetpointUpperLimit). Note: The setpoint is limited to (Setpoint ≥ Config.SetpointUpperLimit).
SetpointLimitL	OUT	Bool	Setpoint low limit. (Default value: FALSE) If SetpointLimitL = TRUE, the absolute lower limit of the setpoint is reached (Setpoint ≥ Config.SetpointLowerLimit). Note: The setpoint is limited to (Setpoint ≥ Config.SetpointLowerLimit).
InputWarningH	OUT	Bool	If InputWarningH = TRUE, the input value has reached or exceeded the warning high limit. (Default value: FALSE)
InputWarningL	OUT	Bool	If InputWarningL = TRUE, the input value has reached or exceeded the warning low limit. (Default value: FALSE)
State	OUT	Int	Current operating mode of the PID controller. (Default value: 0) You can change the operating mode using the Mode input parameter and a rising edge at ModeActivate: • State = 0: Inactive • State = 1: Pre-tuning • State = 2: Manual fine tuning • State = 3: Automatic mode • State = 4: Manual mode • State = 5: Substitute output value approach • State = 6: Transition time measurement • State = 7: Error monitoring • State = 8: Substitute output value approach with error monitoring\ • State = 10: Manual mode without end stop signals
Error	OUT	Bool	If Error = TRUE, at least one error message is pending. (Default value: FALSE) Note: The Error parameter in V1.x PID was the ErrorBits field that contained the error codes. It is now a Boolean flag indicating that an error has occurred.
ErrorBits	OUT	DWord	The PID_3Step instruction ErrorBits parameters table (Page 490) defines the error messages that are pending. (Default value: DW#16#0000 (no error)). ErrorBits is retentive and is reset upon a rising edge at Reset or ErrorAck. Note: In V1.x, the ErrorBits parameter was defined as the Error parameter and did not exist.

Operation of the PID_3Step controller

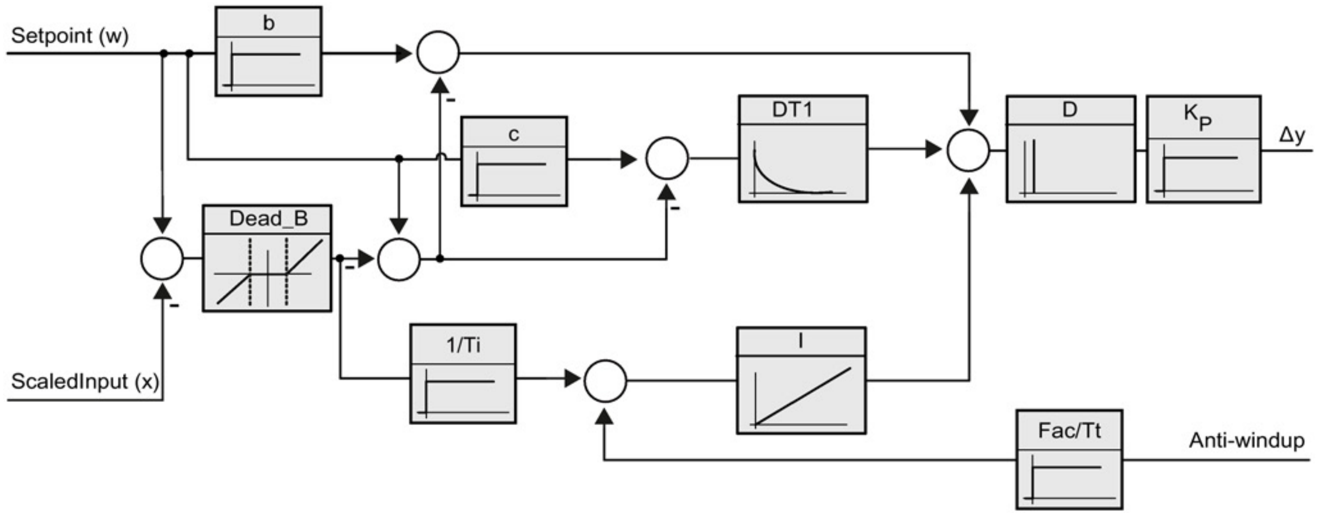


Figure 10-3 Operation of the PID_3Step controller as a PID T1 controller with anti-windup

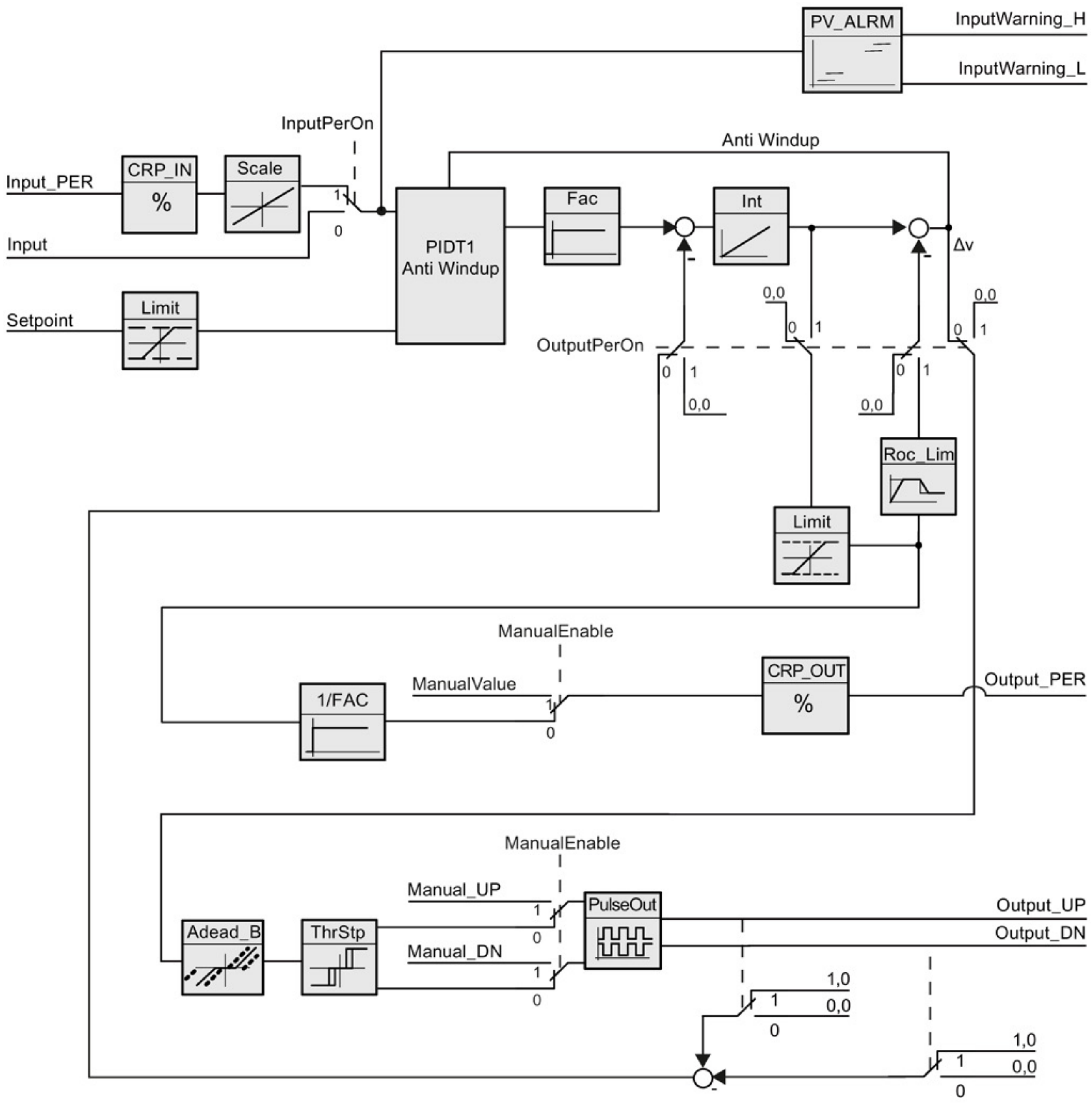


Figure 10-4 Operation of the PID_3Step controller without position feedback

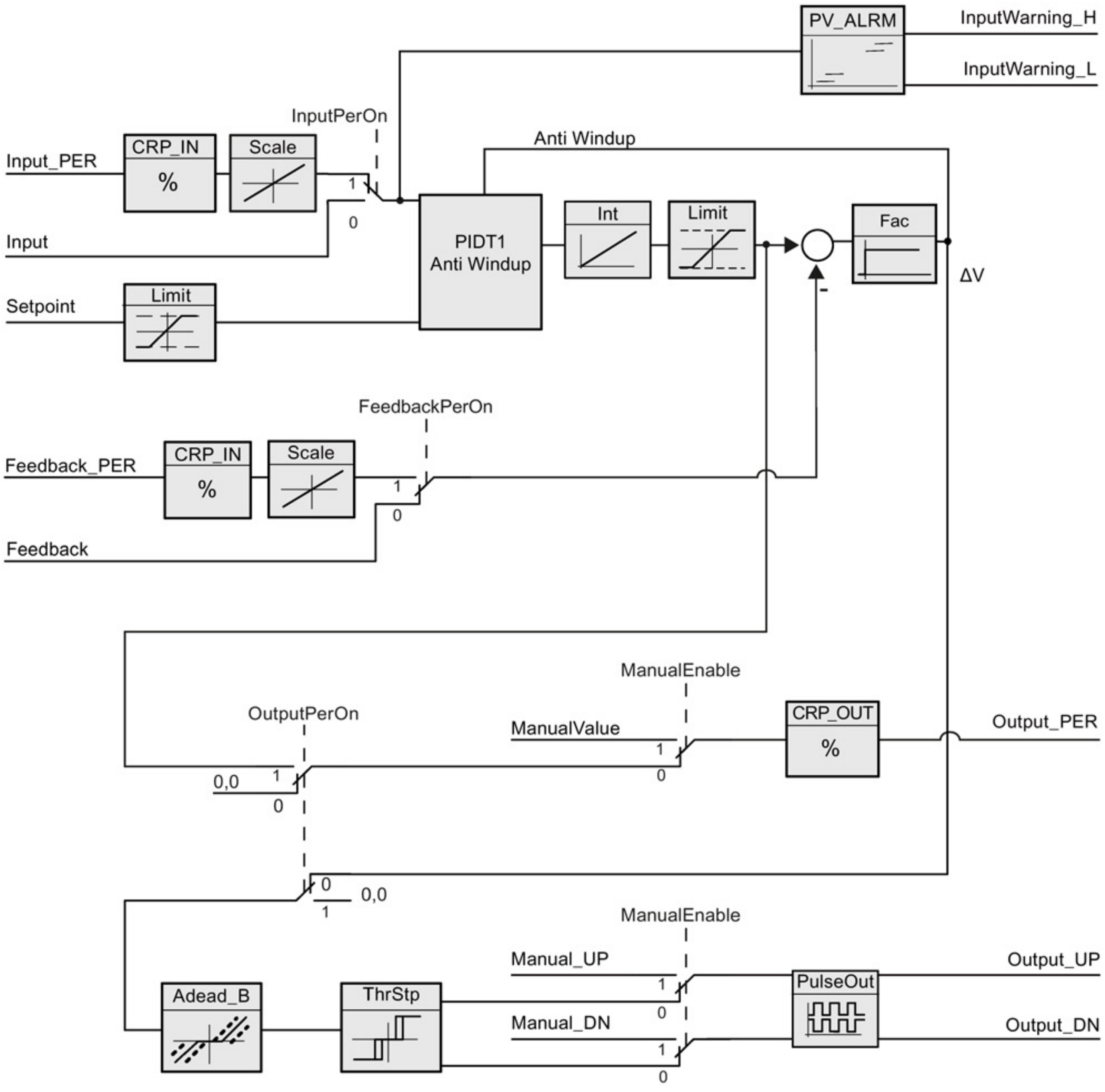


Figure 10-5 Operation of the PID_3Step controller with position feedback enabled

10.2.6 PID_3Step instruction ErrorBit parameters

If several errors are pending, the values of the error codes are displayed by means of binary addition. The display of error code 0003, for example, indicates that the errors 0001 and 0002 are also pending.

Table 10- 24 PID_3STEP instruction ErrorBit parameters

ErrorBit (DW#16#...)	Description
0000	No error
0001 [1, 2]	The Input parameter is outside the process value limits. Input > Config.InputUpperLimit Input < Config.InputLowerLimit
0002 [2, 3]	Invalid value at the Input_PER parameter. Check whether an error is pending at the analog input.
0004 [4]	Error during fine tuning. Oscillation of the process value could not be maintained.
0010 [4]	The setpoint was changed during tuning. Note: You can set the permitted fluctuation on the setpoint at the CancelTuningLevel tag.
0020	Pre-tuning is not permitted during fine tuning. Note: If ActivateRecoverMode = TRUE before the error occurred, PID_3Step remains in fine tuning mode.
0080 [4]	Error during pre-tuning. Incorrect configuration of output value limits. Check whether the limits of the output value are configured correctly and match the control logic.
0100 [4]	Error during fine tuning resulted in invalid parameters.
0200 [2, 3]	Invalid value at the Input parameter: Value has an invalid number format.
0400 [2, 3]	Calculating the output value failed. Check the PID parameters.
0800 [1, 2]	Sampling time error: PID_3Step is not called within the sampling time of the cyclic interrupt OB.
1000 [2, 3]	Invalid value at the Setpoint parameter: Value has an invalid number format.
2000 [1, 2, 5]	Invalid value at the Feedback_PER parameter. Check whether an error is pending at the analog input.
4000 [1, 2, 5]	Invalid value at the Feedback parameter: Value has an invalid number format.
8000 [1, 2]	Error during digital position feedback. Actuator_H = TRUE and Actuator_L = TRUE. The actuator cannot be moved to the substitute output value and remains in its current position. Manual mode is not possible in this state. In order to move the actuator from this state, you must deactivate the "Actuator end stop" (Config.ActuatorEndStopOn = FALSE) or switch to manual mode without end stop signals (Mode = 10).

ErrorBit (DW#16#...)	Description
10000	Invalid value at the ManualValue parameter: Value has an invalid number format. The actuator cannot be moved to the manual value and remains in its current position. Assign a valid value in ManualValue or move the actuator in manual mode with Manual_UP and Manual_DN.
20000	Invalid value at the SavePosition tag: Value has an invalid number format. The actuator cannot be moved to the substitute output value and remains in its current position.
40000	Invalid value at the Disturbance parameter: Value has an invalid number format. Note: If automatic mode was active and ActivateRecoverMode = FALSE before the error occurred, Disturbance is set to zero. PID_3Step remains in automatic mode. Note: If pre-tuning or fine tuning mode was active and ActivateRecoverMode = TRUE before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter. If Disturbance in the current phase has no effect on the output value, tuning is not canceled. The error has no effect during transition time measurement.

1 Note: If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step remains in automatic mode.

2 Note: If pre-tuning, fine tuning, or transition time measurement mode were active and ActivateRecoverMode = TRUE before the error occurred, PID_3Step switches to the operating mode that is saved in the Mode parameter.

3 Note: If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_3Step switches to "Approach substitute output value with error monitoring" or "Error monitoring" mode. As soon as the error is no longer pending, PID_3Step switches back to automatic mode.

4 Note: If ActivateRecoverMode = TRUE before the error occurred, PID_3Step cancels the tuning and switches to the operating mode that is saved in the Mode parameter.

5 The actuator cannot be moved to the substitute output value and remains in its current position. In manual mode, you can change the position of the actuator only with Manual_UP and Manual_DN, and not with ManualValue.

10.2.7 PID_3Step instruction Warning parameters

If the PID controller has several warnings pending, it displays the values of the error codes by means of binary addition. The display of error code 0003, for example, indicates that the errors 0001 and 0002 are pending.

Table 10- 25 PID_Compact instruction Warning parameters

Warning (DW#16#...)	Description
0000	No warning pending.
0001 [1]	The point of inflection was not found during pretuning.
0002	Oscillation was enforced during “tuning in run”. (The "Warning" parameter suppresses this warning and is only visible in the “WarningInternal” parameter for diagnostic purposes.)
0004 [1]	The setpoint was limited to the configured limits.
0008 [1]	Not all the necessary controlled system properties were defined for the selected method of calculation. Instead, the PID parameters were calculated using the TIR.TuneRuleHeat / TIR.TuneRuleCool = 3 method.
0010	The operating mode could not be changed because Reset = TRUE or ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040 [1]	The process value exceeded one of its warning limits.
0080	Invalid value at Mode. The operating mode is not switched.
0100 [1]	The manual value was limited to the limits of the controller output.
0200	The specified rule for tuning is not supported. No PID parameters are calculated.
1000	The substitute output value cannot be reached because it is outside the output value limits.

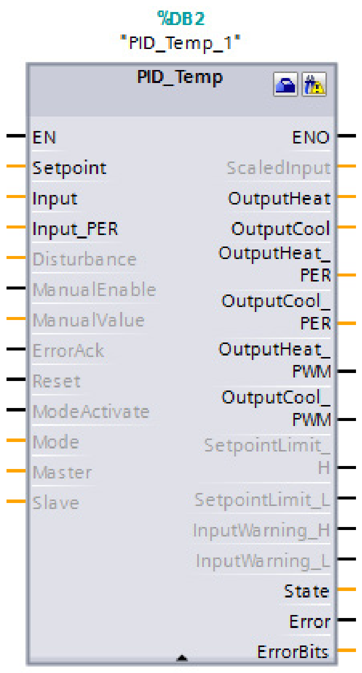
[1] Note: The PID controller deleted the following warnings automatically as soon as the cause is eliminated or the user action repeated with valid parameters: 0001, 0004, 0008, 0040, and 0100.

10.2.8 PID_Temp instruction

The PID_Temp instruction provides a universal PID controller that allows handling of the specific requirements of temperature control.

Table 10- 26 PID_Temp instruction

LAD / FBD	SCL	Description

 %DB2
"PID_Temp_1"
PID_Temp
EN
Setpoint
Input
Input_PER
Disturbance
ManualEnable
ManualValue
ErrorAck
Reset
ModeActivate
Mode
Master
Slave
ENO
ScaledInput
OutputHeat
OutputCool
OutputHeat_PER
OutputCool_PER
OutputHeat_PWM
OutputCool_PWM
SetpointLimit_H
SetpointLimit_L
InputWarning_H
InputWarning_L
State
Error
ErrorBits
	`"PID_Temp_1"(`
	`Setpoint:=_real_in_,`
	`Input:=_real_in_,`
	`Input_PER:=_int_in_,`
	`Disturbance:=_real_in_,`
	`ManualEnable:=_bool_in_,`
	`ManualValue:=_real_in_,`
	`ErrorAck:=_bool_in_,`
	`Reset:=_bool_in_,`
	`ModeActivate:=_bool_in_,`
	`Mode:=_int_in_,`
	`Master:=_dword_in`
	`Save:=_dword_in`
	`ScaledInput=>_real_out_,`
	`OutputHeat=>_real_out_,`
	`OutputCool=>_real_out_,`
	`OutputHeat_PER=>_int_out_,`
	`OutputCool_PER=>_int_out_,`
	`Out-`
	`putHeat_PWM=>_bool_out_,`
	`Out-`
	`putCool_PWM=>_bool_out_,`
	`SetpointLimit_H=>_bool_out_,`
	`SetpointLimit_L=>_bool_out_,`
	`InputWarn-`
	`ing_H=>_bool_out_,`
	`InputWarn-`
	`ing_L=>_bool_out_,`
	`State=>_int_out_,`
	`Error=>_bool_out_,`
	`ErrorBits=>_dword_out_);`
		PID_Temp provides these capabilities:
		• Heating and cooling of the process with different actuators
		• Integrated autotuning to handle temper-ature processes
		• Cascading to process more than one temperature that depends on the same actuator

1 STEP 7 automatically creates the technology object and instance DB when you insert the instruction. The instance DB contains the parameters of the technology object.

2 In the SCL example, "PID_Temp_1" is the name of the instance DB.

Table 10- 27 Data types for the parameters

Parameter and type		Data type	Description
Setpoint	IN	Real	Setpoint of the PID controller in automatic mode. (Default value: 0.0)
Input	IN	Real	A tag of the user program is used as the source of the process value. (Default value: 0.0) If you are using the Input parameter, you must set Config.InputPerOn = FALSE.
Input_PER	IN	Int	An Analog input is used as the source of the process value. (Default value: 0) If you are using the Input_PER parameter, you must set Config.InputPerOn = TRUE.
Disturbance	IN	Real	Disturbance variable or pre-control value
ManualEnable	IN	Bool	Enables or disables the manual operation mode. (Default value: FALSE): • A FALSE to TRUE edge activates Manual mode, while State = 4, Mode remains unchanged. As long as ManualEnable = TRUE, you cannot change the operating mode using a rising edge at ModeActivate or use the commissioning dialog. • A TRUE to FALSE edge activates the operating mode that is assigned by Mode. Note: We recommend that you change the operating mode using ModeActivate only.
ManualValue	IN	Real	Output value for manual operation. (Default value: 0.0) You can use values from Config.OutputLowerLimit to Config.OutputUpperLimit.
ErrorAck	IN	Bool	Resets the ErrorBits and warning outputs with a FALSE to TRUE edge. (Default value: FALSE)
Reset	IN	Bool	Restarts the controller. (Default value: FALSE): • FALSE to TRUE edge: – Switches to "inactive" mode – Resets the ErrorBits and warning outputs – Clears Integral action – Maintains PID parameters • As long as Reset = TRUE, PID_Temp remains in Inactive mode (State = 0). • TRUE to FALSE edge: – PID_Temp switches to the operating mode that is saved in the Mode parameter.
ModeActivate	IN	Bool	The PID_Temp switches to the operating mode that is saved in the Mode parameter with a FALSE to TRUE edge. (Default value: FALSE)

Parameter and type		Data type	Description
Mode	IN/OUT	Int	Activated on the leading edge of the Mode Activate input. Operating mode selection (Default value: 0.0): • Mode = 0: Inactive • Mode = 1: Pretuning • Mode = 2: Fine tuning • Mode = 3: Automatic mode • Mode = 4: Manual mode “Substitute output value with error monitoring” (State = 5). This cannot be activated by the user; it is only an automatic error reaction.
Master	IN/OUT	DWord	Cascade connection to master (AntiWindUp and tuning conditions). (Default value: DW#16#0000) • Bits 0 - 15: Not used in PID_Temp instruction • Bits 16 - 23: Limit counter: A slave increments this value if it reaches its limitation. The number of slaves in limitation is processed for Anti-Windup-functionality (Refer to the Config.Cascade.AntiWindUpMode parameter. • Bit 24: IsAutomatic: This bit is set to "1" if all slaves of this controller are in Automatic mode and are processed to check conditions for tuning in a cascade. This bit is identical to the AllSlaveAutomaticState parameter. • Bit 25: "IsReplacement-Setpoint": This bit is set to "1" if a slave of this controller has the "Replacement Setpoint" activated and is processed to check conditions for tuning in a cascade. The inverted value is stored in the NoSlaveReplacementSetpoint parameter.
Slave	IN/OUT	DWord	
ScaledInput	OUT	Real	Scaled process value. (Default value: 0.0)
OutputHeat[1]	OUT	Real	Output value for heating in REAL format. (Default value: 0.0) This output value is calculated, independent from the output selection, using the Config.Output.Heat.Select parameter.
OutputCool[1]	OUT	Real	Output value for cooling in REAL format. (Default value: 0.0) This output value is calculated, independent from the output selection, using the Config.Output.Cool.Select parameter.
OutputHeat_PER[1]	OUT	Int	Output value for heating in peripheral format (Default value: 0) This output value is only calculated if selected using the Config.Output.Heat.Select = 2 parameter. If not selected, this output is always "0".
OutputCool_PER[1]	OUT	Int	Output value for cooling in peripheral format (Default value: 0) This output value is only calculated if selected using the Config.Output.Cool.Select = 2 parameter. If not selected, this output is always "0".
OutputHeat_PWM[1]	OUT	Bool	Pulse-width-modulated output value for heating. (Default value: FALSE) This output value is only calculated if selected using the Config.Output.Heat.Select = 1 (default value) parameter. If not selected, this output is always FALSE.

Parameter and type		Data type	Description
OutputCool_PWM[1]	OUT	Bool	Pulse-width-modulated output value for cooling. (Default value: FALSE) This output value is only calculated if selected using the Config.Output.Cool.Select = 1 (default value) parameter. If not selected, this output is always FALSE.
SetpointLimit_H	OUT	Bool	Setpoint high limit. (Default value: FALSE) If SetpointLimit_H = TRUE, the absolute setpoint upper limit is reached (Setpoint ≥ Config.SetpointUpperLimit). The setpoint is limited to Config.SetpointUpperLimit.
SetpointLimit_L	OUT	Bool	Setpoint low limit. (Default value: FALSE) If SetpointLimit_L = TRUE, the absolute setpoint lower limit is reached (Setpoint ≤ Config.SetpointLowerLimit). The setpoint is limited to Config.SetpointLowerLimit.
InputWarning_H	OUT	Bool	If InputWarning_H = TRUE, the process value has reached or exceeded the warning high limit. (Default value: FALSE)
InputWarning_L	OUT	Bool	If InputWarning_L = TRUE, the process value has reached or fallen below the warning low limit. (Default value: FALSE)
State	OUT	Int	Current operating mode of the PID controller. (Default value: 0) You can change the operating mode using the Mode input parameter and a rising edge at ModeActivate: • State = 0: Inactive • State = 1: Pre-tuning • State = 2: Fine tuning • State = 3: Automatic mode • State = 4: Manual mode • State = 5: Substitute output value with error monitoring
Error	OUT	Bool	If Error = TRUE, at least one error message is pending in this cycle. (Default value: FALSE) Note: The Error parameter in V1.x PID was the ErrorBits field that contained the error codes. It is now a Boolean flag indicating that an error has occurred.
ErrorBits	OUT	DWord	The PID_Temp instruction, ErrorBits parameters table (Page 502)defines the error messages that are pending. (Default value: DW#16#0000 (no error)). ErrorBits is retentive and is reset upon a rising edge at Reset or ErrorAck. Note: In V1.x, the ErrorBits parameter was defined as the Error parameter and did not exist.
Warning	OUT	DWord	The PID_Temp instruction, Warning parameters table (Page 504) defines the user-relevant warning messages that are pending. (Default value: DW#16#0000 (no warning)).
WarningInternal	OUT	DWord	The PID_Temp instruction, WarningInternal parameters table defines the warning internal messages that are pending (includes all warnings). (Default value: DW#16#0000 (no warning internal)).

[1] You can use the outputs of the Output, Output_PER, and Output_PWM parameters in parallel.

10.2.8.1 Operation of the PID_Temp controller

Selecting heating and/or cooling control

You must first select if you need a cooling device in addition to the heating output at parameter "ActivateCooling". Afterwards, you must define if you want to use two PID-parameter-sets (advanced mode) or only one PID-parameter-set with an additional heating/cooling-factor at parameter "AdvancedCooling".

Using CoolFactor

In case you want to apply a heating/cooling-factor, you must define the value manually. You have to identify the value from the technical data of your application (ratio of proportional gain of the actuators (for example, the ratio of maximum heating- and cooling-power of the actuators) and assign it to parameter "CoolFactor". A heating/cooling-factor of 2.0 means that the heating device is two times more effective than the cooling device. If you use cooling factor, PID_Temp calculates the output signal and, depending on its sign, multiply the output signal with the heating/cooling-factor (when sign is negative) or not (when sign is positive).

Using two PID-parameter-sets

Different PID-parameter-sets for heating and cooling can be automatically detected during commissioning. You can expect a better control performance compared to heating/cooling-factor because, in addition to different proportional gains, you can consider different delay times with two parameters-sets. However, the disadvantage is that this can take more time for the tuning process. If PID-parameter switchover is activated (Config.AdvancedCooling = TRUE), the PID_Temp controller detects in "Automatic mode" (controlling is active) if heating or cooling is necessary at that time and uses PID-parameter-sets for control.

ControlZone

With the PID_Temp controller, you can define a control zone for each parameter-set at parameter "ControlZone". If the control deviation (setpoint – input) is within the control zone, PID_Temp uses the PID-algorithm to calculate the output signals. However, if the control deviation leaves the defined range, the output is set to the maximum heating or maximum cooling output value (cooling output activated) / minimum heating output value (cooling output deactivated). You can use this functionality to reach the desired setpoint faster, especially for initial heating-up of slow temperature processes.

DeadZone

In the "DeadZone" parameter, you can define a width of control deviation for heating and cooling that is neglected by the PID-algorithm. This means a control deviation within this range is suppressed, and the PID_Temp controller behaves like the setpoint and process values are identical. Thus, you can reduce unnecessary intervention by the controller around the setpoint and conserve the actuator. If you want to apply a DeadZone, you must define the value manually. Auto tuning does not automatically set the DeadZone value. DeadZone is symmetric (between -Retain.CtrlParams.Heat.DeadZone and +Retain.CtrlParams.Heat.DeadZone) for heating controllers without cooling or heating/cooling controllers using CoolFactor. DeadZone can be asymmetric (between -Retain.CtrlParams.Cool.DeadZone and +Retain.CtrlParams.Heat.DeadZone) for heating/cooling controllers using two PID-parameter sets.

PID_Temp controller operations

The following block diagrams illustrate the PID_Temp instruction standard and cascade operations:

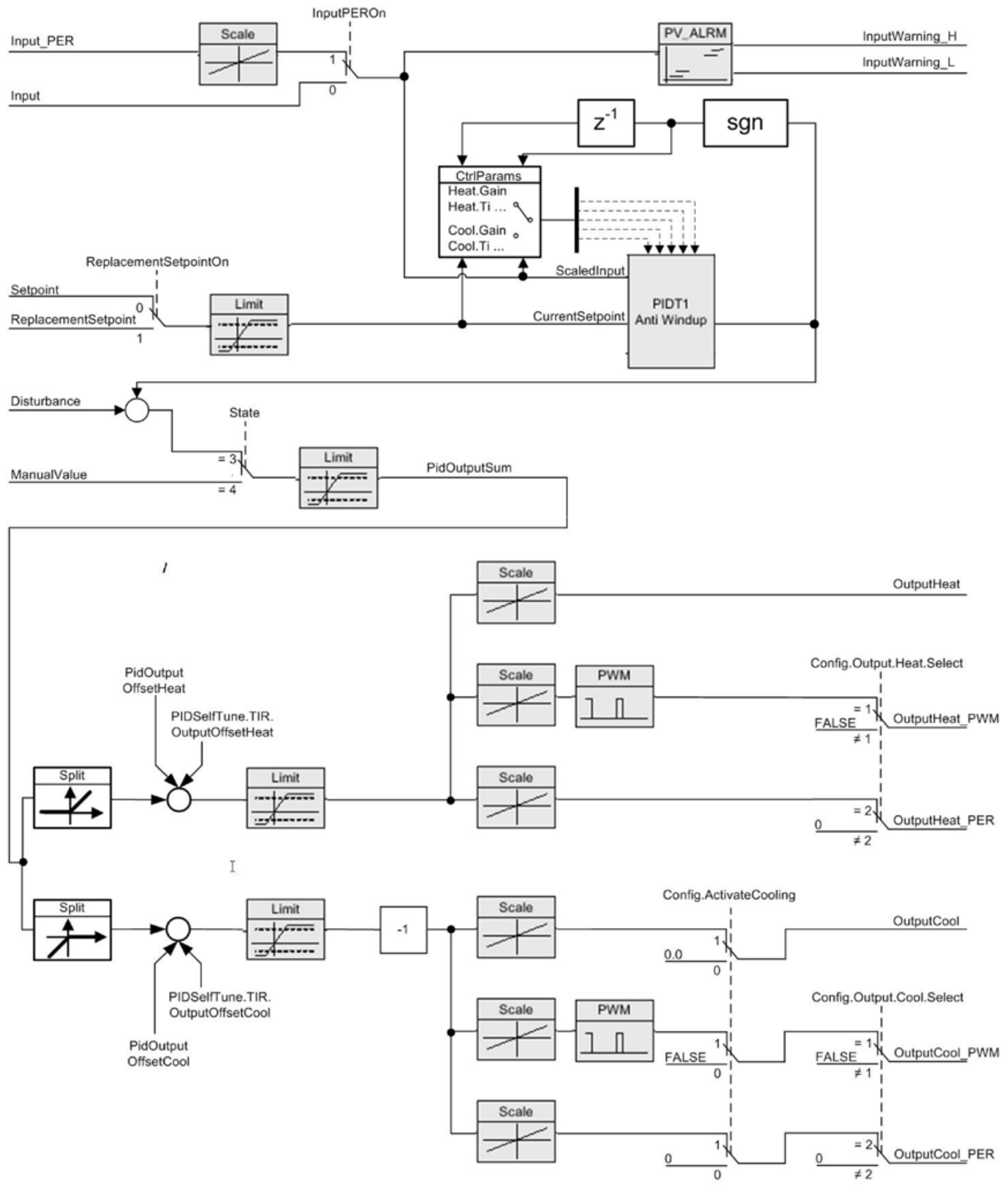


Figure 10-6 PID_Temp_Operation_Block_Diagram

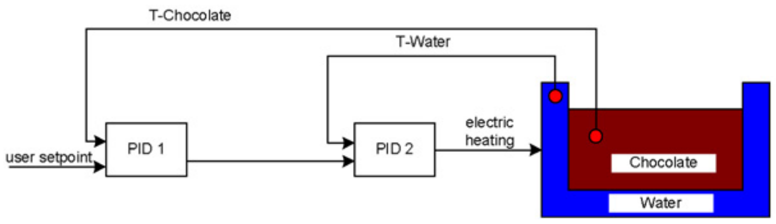


Figure 10-7 PID_Temp_Cascade_Operation_Block_Diagram

10.2.8.2 Cascading controllers

You can cascade temperature PID controllers to process more than one temperature that depend on the same actuator.

Call order

You must call cascaded PID controllers in the same OB cycle. First, you must call the master, then, the next slave(s) in the control signal flow, and finally on to the last slave in the cascade. The PID_Temp instruction does not make an automatic check of call order.

Communication connections

When cascading controllers, you must connect the master and slave so that they can share information with each other. You must connect a slave’s “Master” IN/OUT parameter to its master’s “Slave” IN/OUT parameter in the signal flow direction.

This shows a connection of PID_Temp controllers in a cascade with two sub-cascades: "PID_Temp1" provides the setpoint. The configuration connects the outputs of "PID_Temp2", "PID_Temp3", "PID_Temp5", "PID_Temp6", and "PID_Temp8" to the process:

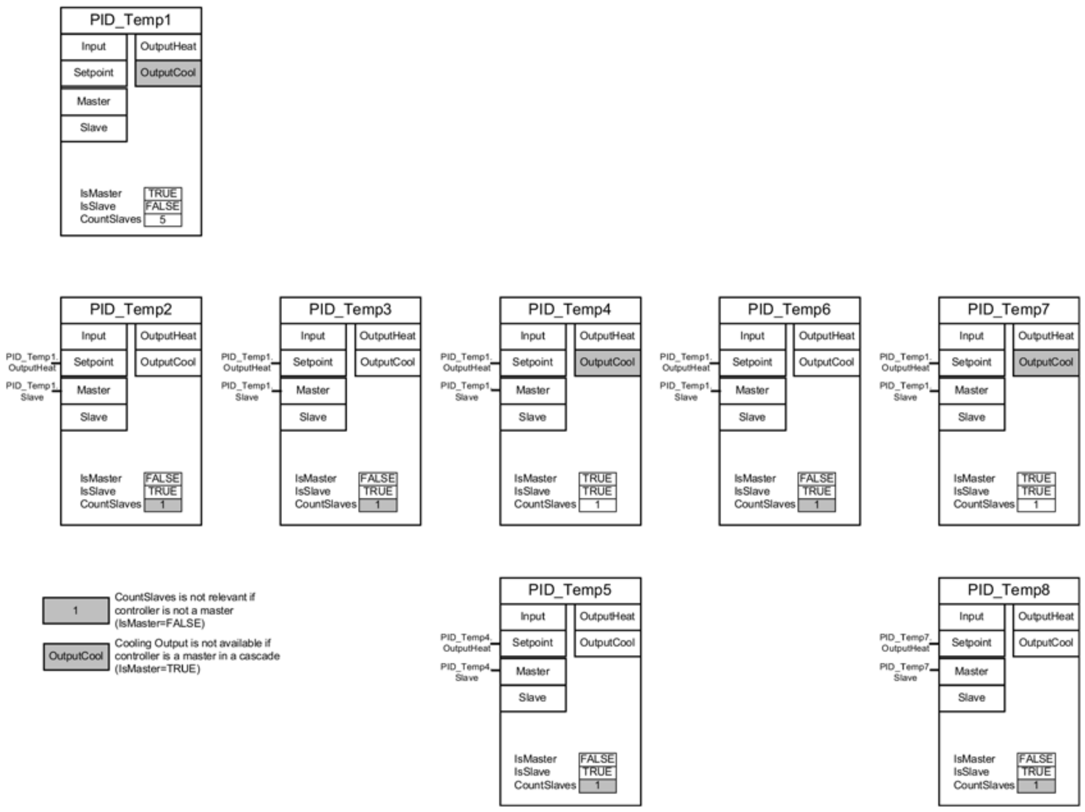


Figure 10-8 PID_Temp_Cascading_communication_connection

Replacement setpoint

The PID_Temp instruction provides a second setpoint input at the “ReplacementSetpoint” parameter that you can activate by setting the parameter “ReplacementSetpointOn” = TRUE. You can use “ReplacementSetpoint” as your setpoint input during commissioning or tuning of a slave controller without having to disconnect the output-to-setpoint connection between master and slave. This connection is necessary for normal operation of the cascade.

In this way, you do not have to change your program and download it if you want to temporarily separate a slave from its master. You only have to activate the "ReplacementSetpoint" and deactivate it again when you finish.The setpoint value is effective for the PID algorithm when you can see the value at the “CurrentSetpoint” parameter.

Autotuning

An autotuning for a cascaded master controller must meet these requirements:

- Be commissioned from its inner slave to the first master.
- All slaves of the master have to be in “Automatic mode".
- The output of the master must be the setpoint for the slaves.

PID_Temp instruction will provide the following support for autotuning in the cascade:

- If you start autotuning for a master controller, the master checks to see if all slaves are in “Automatic mode” and for the deactivation of the Replacement-Setpoint-functionality for all slaves (“ReplacementSetpointOn” = FALSE). If you do not meet these conditions, you cannot autotune the master. The master cancels the tuning, goes to “Inactive” mode" (if “ActivateRecoverMode” = FALSE), or back to the mode stored in the “Mode” parameter (if “ActivateRecoverMode” = TRUE). The master displays the error message 200000hex ("Error with master in the cascade. Slaves are not in automatic mode or have a substitute setpoint enabled and are preventing tuning of the master.").
- When all slaves are in “Automatic mode”, the system sets the parameter “AllSlaveAutomaticState” = TRUE. You can apply this parameter in your programs or localize the cause of error 200000hex.
- When the "ReplacementSetpoint" is deactivated for all slaves, the system sets the parameter “NoSlaveReplacementSetpoint” = TRUE. You can apply this parameter in their programs or localize the cause of error 200000hex.

When the PID_Temp instruction commissioning dialog is used, you have further support for cascade tuning (Page 524).

Operation modes and error handling

The PID_Temp controller does not allow switching of the operating mode by its master or slaves. This means that a master inside the cascade stays in its current mode when a slave raises an error. This is an advantage if two or more parallel slaves operate with this master controller; an error in one chain does not shut down the parallel chain.

Similarly, a slave inside the cascade stays in its current operation mode, if its master has an error. However, further operation of the slave then depends on the configuration of the master because the slave’s setpoint is the the master’s output. This means that if you configure the master with “ActivateRecoverMode” = TRUE and an error occurs, the master outputs the last valid or a substitute output value as setpoint for the slave. If you configure the master with “ActivateRecoverMode” = FALSE, the master switchs to “Inactive mode" and sets all outputs to "0.0" so that the slave uses "0.0" as its setpoint.

Because only the slave controllers have direct access to the actuators and these stay in their operating mode in case of a master error, you can avoid damage to the process. For example, for plastics processing devices, it is fatal for the slaves to stop working, shut down the actuators, and allow the plastic to harden inside the device solely because the master controller had an error.

Anti-windup

A slave in a cascade gets its setpoint from the output of his master. If the slave reaches its own output limits while the master still sees a control deviation (setpoint – input), the master freezes or reduces its integration contribution to prevent a so-called "WindUp". In case of a "WindUp", the master increases its integration contribution to a very large value and must reduce it first, before the controller can again have a normal reaction. Such a "WindUp" affects the dynamic of the control negatively. The PID_Temp provides ways to prevent this effect in a cascade by configuring the parameter “Config.Cascade.AntiWindUpMode” of the master controller:

Value	Description
0	Deactivates Anti-Windup functionality.
1	Reduces the integration contribution of the master controller at the ratio “slaves in limitation” to “existing slaves” (parameter “CountSlaves”).
2	Freezes the integration contribution of the master as soon as one slave reaches its limitation. Only relevant if "Config.Cascade.IsMaster" = TRUE.

10.2.9 PID_Temp instruction ErrorBit parameters

If the PID controller has several warnings pending, it displays the values of the error codes by means of binary addition. The display of error code 0003, for example, indicates that the errors 0001 and 0002 are pending.

Table 10- 28 PID_Temp instruction ErrorBit parameters

ErrorBit (DW#16#...)	Description
0000	No error
0001 [1, 2]	The Input parameter is outside the process value limits. Input > Config.InputUpperLimit Input < Config.InputLowerLimit
0002 [2, 3]	Invalid value at the Input_PER parameter. Check whether an error is pending at the analog input.
0004 [4]	Error during fine tuning. Oscillation of the process value could not be maintained.
0008 [4]	Error at start of pre-tuning. The process value is too close to the setpoint. Start fine tuning.
0010 [4]	The setpoint was changed during tuning. Note: You can set the permitted fluctuation on the setpoint at the CancelTuningLevel tag.
0020	Pre-tuning is not permitted during fine tuning. Note: If ActivateRecoverMode = TRUE before the error occurred, PID_Temp remains in fine tuning mode.
0040 [4]	Error during pretuning. The cooling could not reduce the process value.
0080 [4]	Error during pre-tuning. Incorrect configuration of output value limits. Check whether the limits of the output value are configured correctly and match the control logic.

ErrorBit (DW#16#...)	Description
0100 [4]	Error during fine tuning resulted in invalid parameters.
0200 [2, 3]	Invalid value at the Input parameter: Value has an invalid number format.
0400 [2, 3]	Calculation of the output value failed. Check the PID parameters.
0800 [1, 2]	Sampling time error: PID_Temp is not called within the sampling time of the cyclic interrupt OB.
1000 [2, 3]	Invalid value at the Setpoint parameter: Value has an invalid number format.
10000	Invalid value at the ManualValue parameter: Value has an invalid number format. Note: If ActivateRecoverMode = TRUE before the error occurred, PID_Temp uses SubstituteOutput as the output value. As soon as you assign a valid value in the ManualValue parameter, PID_Temp uses it as the output value.
20000	Invalid value at the SubstituteValue tag: Value has an invalid number format. PID_Temp uses the output value low limit as the output value. Note: If automatic mode was active before the error occurred, ActivateRecoverMode = TRUE, and the error is no longer pending, PID_Temp switches back to automatic mode.
40000	Invalid value at the Disturbance parameter: Value has an invalid number format. Note: If automatic mode was active and ActivateRecoverMode = FALSE before the error occurred, Disturbance is set to zero. PID_Temp remains in automatic mode. Note: If pre-tuning or fine tuning mode was active and ActivateRecoverMode = TRUE before the error occurred, PID_Temp switches to the operating mode that is saved in the Mode parameter. If Disturbance in the current phase has no effect on the output value, tuning is not canceled.
200000	Error with master in the cascade. Slaves are not in automatic mode or have a substitute setpoint enabled, preventing tuning of the master.
400000	The PID controller does not permit pretuning for heating while cooling is active.
800000	The process value must be close to the setpoint in order to start pretuning for cooling.
1000000	Error starting tuning. “Heat.EnableTuning” and “Cool.EnableTuning” are not set or do not match the configuration.
2000000	Pretuning for cooling requires successful pretuning for heating.

ErrorBit (DW#16#...)	Description
4000000	Error starting fine tuning. “Heat.EnableTuning” and “Cool.EnableTuning” cannot be set at the same time.
8000000	Error during PID parameter calculation resulted in invalid parameters (for example, negative Gain; the current PID parameters remain unchanged and tuning has no effect).

[1] Note: If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Temp remains in automatic mode.

[2] Note: If pre-tuning or fine tuning mode was active before the error occurred and ActivateRecover-Mode = TRUE, PID_Temp switches to the operating mode that is saved in the Mode parameter.

[3] Note: If automatic mode was active before the error occurred and ActivateRecoverMode = TRUE, PID_Compact outputs the configured substitute output value. As soon as the error is no longer pending, PID_Temp switches back to automatic mode.

[4] Note: If ActivateRecoverMode = TRUE before the error occurred, PID_Temp cancels the tuning and switches to the operating mode that is saved in the Mode parameter.

10.2.10 PID_Temp instruction Warning parameters

If the PID controller has several warnings pending, it displays the values of the error codes by means of binary addition. The display of error code 0003, for example, indicates that the errors 0001 and 0002 are pending.

Table 10- 29 PID_Temp instruction Warning parameters

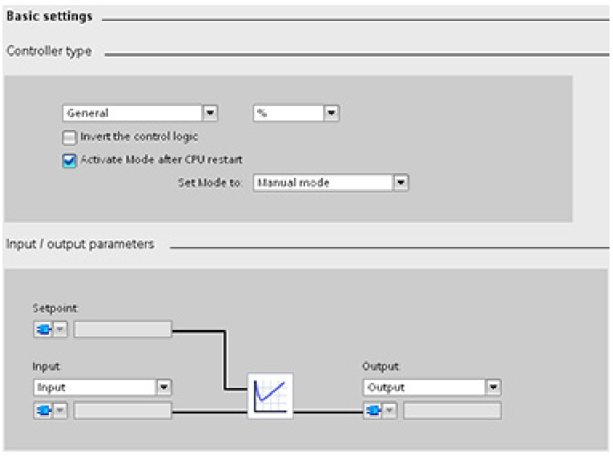
Warning (DW#16#...)	Description
0000	No warning pending.
0001 [1]	The point of inflection was not found during pretuning.
0002	Oscillation was enforced during “tuning in run”. (The "Warning" parameter suppresses this warning and is only visible in the “WarningInternal” parameter for diagnostic purposes.)
0004 [1]	The setpoint was limited to the configured limits.
0008 [1]	Not all the necessary controlled system properties were defined for the selected method of calculation. Instead, the PID parameters were calculated using the TIR.TuneRuleHeat / TIR.TuneRuleCool = 3 method.
0010	The operating mode could not be changed because Reset = TRUE or ManualEnable = TRUE.
0020	The cycle time of the calling OB limits the sampling time of the PID algorithm. Improve results by using shorter OB cycle times.
0040 [1]	The process value exceeded one of its warning limits.
0080	Invalid value at Mode. The operating mode is not switched.
0100 [1]	The manual value was limited to the limits of the controller output.
0200	The specified rule for tuning is not supported. No PID parameters are calculated.
1000	The substitute output value cannot be reached because it is outside the output value limits.

Warning (DW#16#...)	Description
4000	The specified output selection for heating and/or cooling is not supported. Only OutputHeat and OutputCool are active.
8000	The specified value for the PIDSelfTune.SUT.AdaptDelayTime parameter is not supported, so the default value "0" is used.
10000	The specified value for the PIDSelfTune.SUT.CoolingMode parameter is not supported, so the default value "0" is used.

[1] Note: The PID controller deleted the following warnings automatically as soon as the cause is eliminated or the user action repeated with valid parameters: 0001, 0004, 0008, 0040, and 0100.

10.2.11 Configuring the PID_Compact and PID_3Step controllers

The parameters of the technology object determine the operation of the PID controller. Use the icon to open the configuration editor.



Basic settings

Controller type

General | %

Invert the control logic

Activate Mode after CPU restart

Set Mode to: Manual mode

Input / output parameters

Table 10- 30 Example configuration settings for the PID_Compact instruction

Settings		Description
Basic	Controller type	Selects the engineering units.
	Invert the control logic	Allows selection of a reverse-acting PID loop. • If not selected, the PID loop is in direct-acting mode and the output of PID loop increases if input value < setpoint. • If selected, the output of the PID loop increases if the input value > setpoint.
	Enable last mode after CPU restart	Restarts the PID loop after it is reset or if an input limit has been exceeded and returned to the valid range.
	Input	Selects either the Input parameter or the Input_PER parameter (for analog) for the process value. Input_PER can come directly from an analog input module.
	Output	Selects either the Output parameter or the Output_PER parameter (for analog) for the output value. Output_PER can go directly to an analog output module.

Settings		Description
Process value	Scales both the range and the limits for the process value. If the process value goes below the low limit or above the high limit, the PID loop goes to inactive mode and sets the output value to 0. To use Input_PER, you **must** scale the analog process value (input value).	

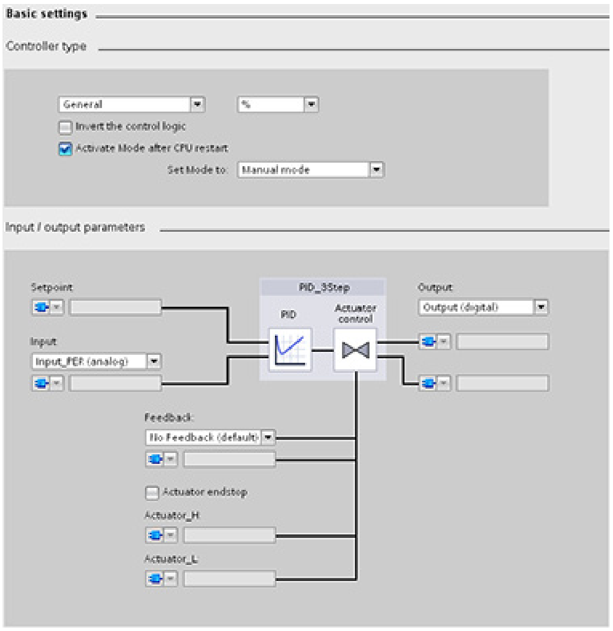

Basic settings
Controller type
General %
Invert the control logic
Activate Mode after CPU restart
Set Mode to: Manual mode
Input / output parameters
Setpoint
Input: Input_PER (analog)
Output: Output (digital)
Feedback: No Feedback (default)
Actuator endstop
Actuator_H
Actuator_L

Table 10- 31 Example configuration settings for the PID_3Step instruction

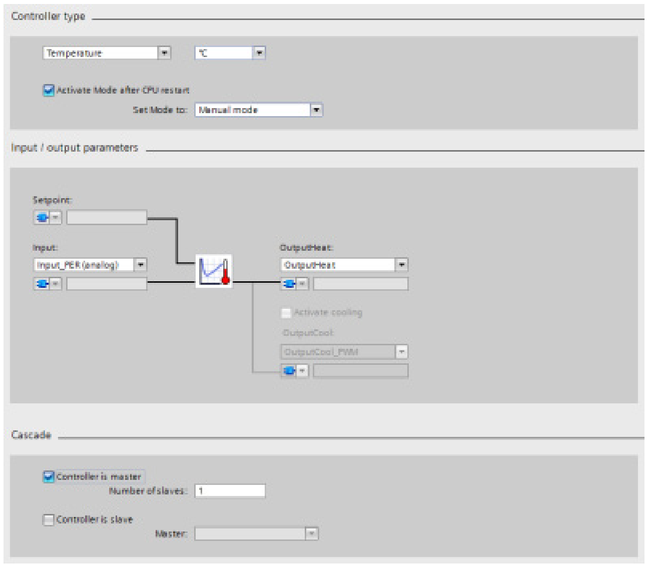
Settings		Description
Basic	Controller type	Selects the engineering units.
	Invert the control logic	Allows selection of a reverse-acting PID loop. • If not selected, the PID loop is in direct-acting mode, and the output of PID loop increases if the input value < setpoint). • If selected, the output of the PID loop increases if the input value > setpoint.
	Activate mode after CPU restart	Restarts the PID loop after it is reset or if an input limit has been exceeded and returned to the valid range. Set Mode to: Defines the mode that the user wants the PID to go to after restart.
	Input	Selects either the Input parameter or the Input_PER parameter (for analog) for the process value. Input_PER can come directly from an analog input module.
	Output	Selects either to use the digital outputs (Output_UP and Output_DN) or to use the analog output (Output_PER) for the output value.
	Feedback	Selects the type of device status returned to the PID loop: • No feedback (default) • Feedback • Feedback_PER
Process value	Scales both the range and the limits for the process value. If the process value goes below the low limit or above the high limit, the PID loop goes to inactive mode and sets the output value to 0. To use Input_PER, you **must** scale the analog process value (input value).	

Settings		Description
Actuator	Motor transition time	Sets the time from open to close for the valve. (Locate this value on the data sheet or the faceplate of the valve.)
	Minimum ON time	Sets the minimum movement time for the valve. (Locate this value on the data sheet or the faceplate of the valve.)
	Minimum OFF time	Sets the minimum pause time for the valve. (Locate this value on the data sheet or the faceplate of the valve.)
	Reaction to error	Defines the behavior of the valve when an error is detected or when the PID loop is reset. If you select to use a substitute position, enter the "Safety position". For analog feedback or analog output, select a value between the upper or lower limit for the output. For digital outputs, you can choose only 0% (off) or 100% (on).
	Scale Position Feedback[1]	• "High end stop" and "Lower end stop" define the maximum positive position (full-open) and the maximum negative position (full-closed). "High end stop" must be greater than "Lower end stop". • "High limit process value" and "Low limit process value" define the upper and lower positions of the valve during tuning and automatic mode. • "FeedbackPER" ("Low" and "High") defines the analog feedback of the valve position. "FeedbackPER High" must be greater than "FeedbackPER Low".
Advanced	Monitoring process value	Sets the warning high and low limits for the process value.
	PID parameters	If the user wishes, he can enter his own PID tuning parameters in this window. The “Enable Manual Entry” check box must be checked to allow this.

[1] "Scale Position Feedback" is editable only if you enabled "Feedback" in the "Basic" settings.

10.2.12 Configuring the PID_Temp controller

The parameters of the technology object determine the operation of the PID controller. Use the icon to open the configuration editor.



Controller type

Temperature ▾ | °C ▾

☑ Activate Mode after CPU restart

Set Mode to: Manual mode

Input / output parameters

Setpoint:

Input: Input_PER (analog)

OutputHeat: OutputHeat

☐ Activate cooling

OutputCool: OutputCool_PWM

Cascade

☑ Controller is master

Number of slaves: 1

☐ Controller is slave

Master:

Table 10- 32 Example configuration settings for the PID_Temp instruction

Settings		Description
Basic	Controller type	Selects the engineering units.
	Activate mode after CPU restart	Restarts the PID loop after it is reset or if an input limit has been exceeded and returned to the valid range. Set Mode to: Defines the mode that the user wants the PID to go to after restart.
	Input	Selects either the Input parameter or the Input_PER parameter (for analog) for the process value. Input_PER can come directly from an analog input module.
	Output Heat	Selects either to use the digital outputs (OutputHeat and OutputHeat_PWM) or to use the analog output (OutputHeat_PER (analog)) for the output value.
	Output Cool	Selects either to use the digital outputs (OutputCool and OutputCool_PWM) or to use the analog output (OutputCool_PER (analog)) for the output value.
Process value	Scales both the range and the limits for the process value. If the process value goes below the low limit or above the high limit, the PID loop goes to inactive mode and sets the output value to 0. To use Input_PER, you **must** scale the analog process value (input value).	
Cascade	Controller is master	Sets the controller as a master and selects the number of slaves.
	Controller is slave	Sets the controller as a slave and selects the number of masters.

Controller type

Setting	TO-DB parameter	Data type	Value range	Description
Physical quantity	"PhysicalQuantity"	Int (Enum)	• General • Temperature (=default)	Pre-selection for physical unit value No multi-value control and not editable in online mode of functional view.
Unit of meas-urement	"PhysicalUnit	Int (Enum)	• General: Units = % • Temperature: Units (possible selections) = – °C (=default) – °F – K	User unit selection is set back to “0” if you change the physical quantity.
Activate mode after CPU restart	"RunModeByStartup"	Bool	Checkbox	If set to TRUE (=default), the controller switches to the state that is stored in the “Mode” variable after a powercycle (Power on - off - on) or PLC STOP-to-RUN transition. Other-wise, the PID_Temp remains in “Inactive” mode.
Set mode to	"Mode"	Int (Enum)	Modes (possible selections): • 0: Inactive • 1: Pretuning • 2: Fine tuning • 3: Automatic mode • 4: Manual mode (=default)	The engineering station (ES) sets the start value of the“Mode” variable according to user selec-tion.The default value of Mode (stored inTO-DB) is Manual Mode.

Input / output parameters

Setting	TO-DB parameter	Data type	Value range	Description
Setpoint	Setpoint	Real)	Real	Only accessible in Property Page. No multi value control in online mode of functional view.
Selection input	"Config.InputPerOn"	Bool (Enum)	Bool	Selects which kind of input to use. Possible selections: • FALSE: "Input" (Real) • TRUE: "Input_PER (analog)"
Input	Input or Input_PER	Real or Int	Real or Int	Only accessible in Properties page. No multi value control in online mode of functional view.
Selection Output (heating)	"Config.Output.Heat.Select"	Int (Enum)	2 >= Config.Output. Heat.Select >= 0	Selects which kind of output to use for heating. Possible selections: • "OutputHeat" (Real) • "OutputHeat_PWM" (Bool) (=default) • "OutputHeat_PER (analog)" (Word) Is set to "OutputHeat" once, if "This controller is a master" checkbox in the "Cascade" section is activated by user.
Output (heating)	OutputHeat, OutputHeat_PER, or OutputHeat_PWM	Real or Int or Bool	Real, Int, or Bool	Only accessible in Properties page. No multi value control in online mode of functional view.

Setting	TO-DB parameter	Data type	Value range	Description
Activate output (cooling)	"Config.ActivateCooling"	Bool	Bool	Checking this checkbox: • Sets the "Con-fig.Output. Heat.PidLowerLimit = 0.0 once. • Sets the "Con-fig.ActivateCooling" parameter to TRUE, instead of FALSE if unchecked (=default). • Activates all other “Output (cooling)” controls (in “Basic settings” and other views). • Changes the line from the PID symbol to the controls from gray to black. • “This controller is a master” checkbox in the “Cascade” sec-tion is disabled. Note: Only available if you do not configure the controller as a master for a cascade (“This control-ler is a master”checkbox in the “Cascade” section is deactivated; "Con-fig.Cascade.IsMaster" = FALSE).

Setting	TO-DB parameter	Data type	Value range	Description
Selection Output (cooling)	"Config.Output.Cool.Select"	Int (Enum)	2 >= Config.Output.Heat.Select >= 0	Selects which kind of output to use for cooling. Possible selections: • “OutputCool” (Real) • “OutputCool_PWM” (Bool) (=default) • “OutputCool_PER (analog)” (Word) Only available if you check “Activate output (cooling)”; (Config.ActivateCooling = TRUE).
Output (cooling)	OutputCool, OutputCool_PER, or OutputCool_PWM	Real or Int or Bool	Real, Int, or Bool	Only accessible in Properties page. No multi value control in online mode of functional view.

Cascade parameters

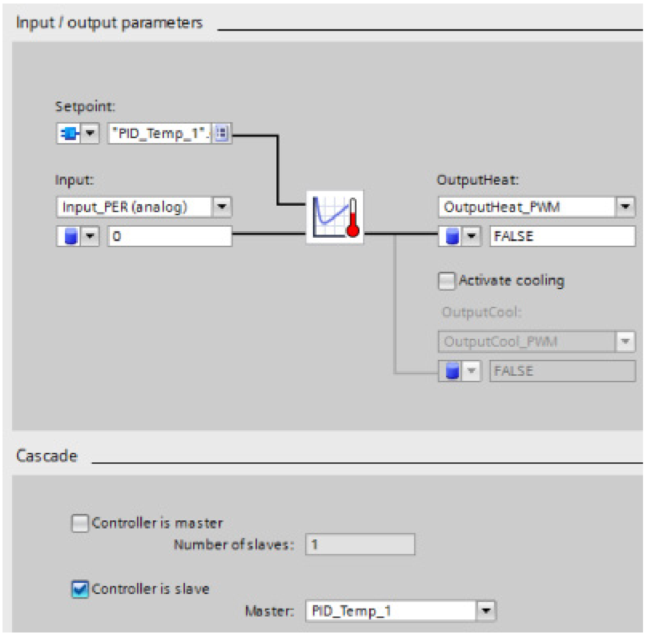
The following parameters enable you to select controllers as masters or slaves and to determine the number of slave copntrollers that receive their setpoint directly from the master controller:

Setting	TO-DB parameter	Data type	Value range	Description
This controller is a master	"Config.Cascade.IsMaster"	Bool	Bool	Shows if this controller is a master in a cascade. When you check this checkbox, you perform the following: • Set the parameter "Con-fig.Cascade.IsMaster" to TRUE, instead of FALSE if unchecked (=default). • Set “Selection Output (heating)” in “Input / output parameters” section to “Out-putHeat” once (Con-fig.Output.Heat.Select = 0). • Enable “Number of Slaves” input field. • Disable “Activate output (cooling)” checkbox in “Input / output parameters” section. Note: Only available if cooling output of this controller is deactivated (“Activate output (cool-ing)” checkbox in “Input / output parameters” sec-tion deactivated (Con-fig.ActivateCooling = FALSE).

Setting	TO-DB parameter	Data type	Value range	Description
Number of slaves	"Config.Cascade.CountSlaves"	Int	255 >= Con-fig.Cascade. CountSlaves >= 1	Number of slave control-lers that get their set-point directly from this master controller. The PID_Temp instruction processes this value, along with others, for anti-windup-handling."Number of slaves is only available if “This controller is a mas-ter” checkbox is activat-ed (Con- (Con-fig.Cascade.IsMaster = TRUE).
This controller is a slave	"Config.Cascade.IsSlave"	Bool	Bool	Shows if this controller is a slave in a cascade. When you check this checkbox, you set the parameter "Con-fig.Cascade.IsSlave" to TRUE, instead of FALSE if unchecked (=default). You must check this checkbox in the property page to enable the “Se-lectionMaster” dropdown list.

Example: Cascading controllers

In the “Basic settings” dialog below, you see the “Input / output parameters” section and the “Cascade” section for slave controller "PID_Temp_2" after selecting "PID_Temp_1" as master. You make the connections between master and slave controller:


Input / output parameters

Setpoint: "PID_Temp_1".

Input: Input_PER (analog)

0

OutputHeat: OutputHeat_PWM

FALSE

Activate cooling

OutputCool: OutputCool_PWM

FALSE

Cascade

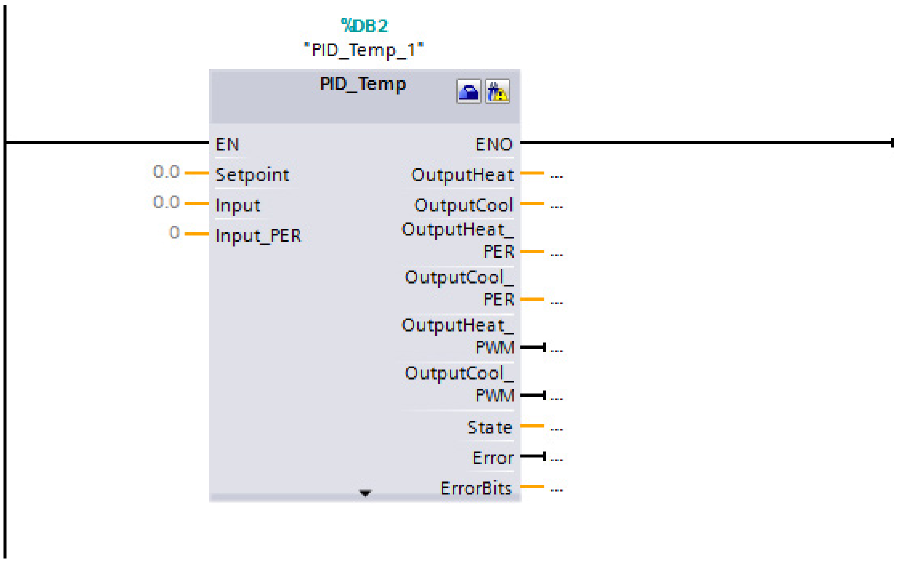
Controller is master

Number of slaves: 1

Controller is slave

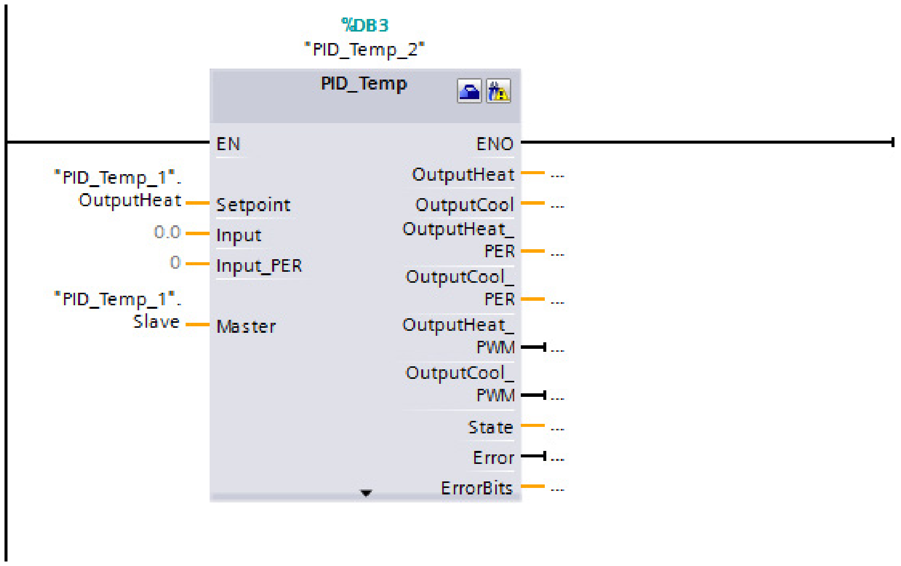
Master: PID_Temp_1

Network 1: In these networks, you make the connection between the "PID_Temp_1" master and the "PID_Temp_2" slave in the programming editor:


%DB2
"PID_Temp_1"
PID_Temp

EN — ENO
0.0 — Setpoint — OutputHeat — ...
0.0 — Input — OutputCool — ...
0 — Input_PER — OutputHeat_PER — ...
OutputCool_PER — ...
OutputHeat_PWM — ...
OutputCool_PWM — ...
State — ...
Error — ...
ErrorBits — ...

Network 2: You make the connection between the "PID_Temp_1" master's “OutputHeat” and "Slave" parameters to the "PID_Temp_2" slave's “Setpoint” and "Master" parameters, respectively:



Autotuning of temperature processes

The PID_Temp instruction provides two modes for auto tuning:

- “Pretuning” (parameter “Mode” = 1)
- “Finetuning” (parameter “Mode” = 2)

Depending on the controller configuration, different variants of these tuning methods are available:

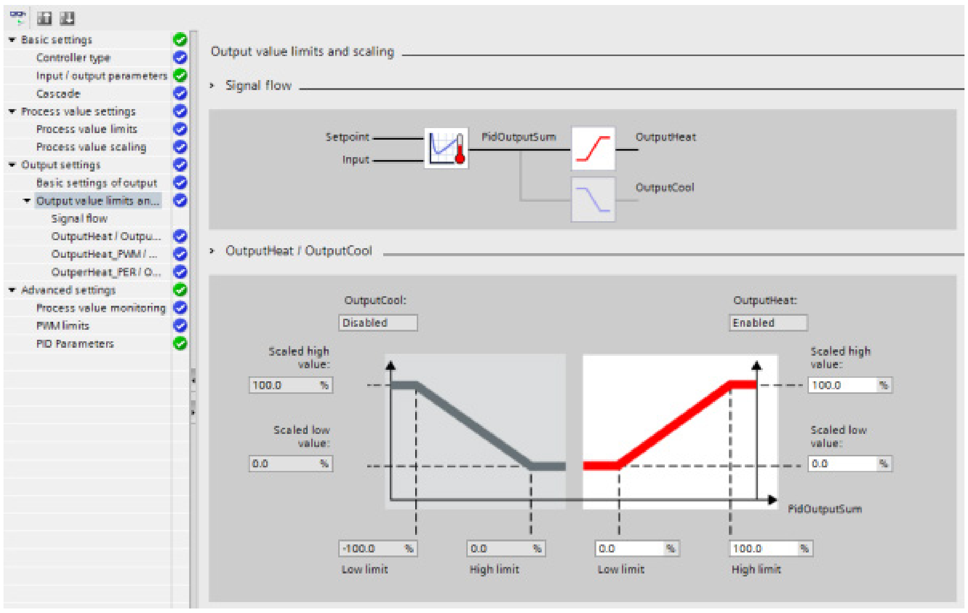
Configuration	**Controller with heating output**	**Controller with heating and cooling output using cooling factor**	**Controller with heating and cooling output using two sets of PID parameters**
Associated TO-DB values	• Config.ActivateCooling = FALSE • Config.AdvancedCooling = irrelevant	• Config.ActivateCooling = TRUE • Config.AdvancedCooling = FALSE	• Config.ActivateCooling = TRUE • Config.AdvancedCooling = TRUE
Available tuning methods	• “Pretuning heating” • “Fine tuning heating" (cooling offset cannot be used)	• “Pretuning heating” • “Fine tuning heating" (cooling offset can be used)	• “Pretuning heating and cooling” • "Pretuning heating” • “Pretuning cooling” • “Fine tuning heating” (cooling offset can be used) • “Fine tuning cooling” (heating offset can be used)

Output value limits and scaling

Cooling activation disabled

If you configure the PID_Temp instruction as master for a cascade “Activate output (cooling)” checkbox in “Basic settings” view is unchecked and disabled, all settings in the “Output settings” view that depend on cooling activation are disabled, too.

The figure below shows the "Output value limits and scaling” section in the “Output settings” view with cooling deactivated (OutputHeat_PWM selected in “Input / output parameters” view and OutputHeat always enabled):



Output value limits and scaling

Signal flow

OutputHeat / OutputCool

OutputCool: Disabled

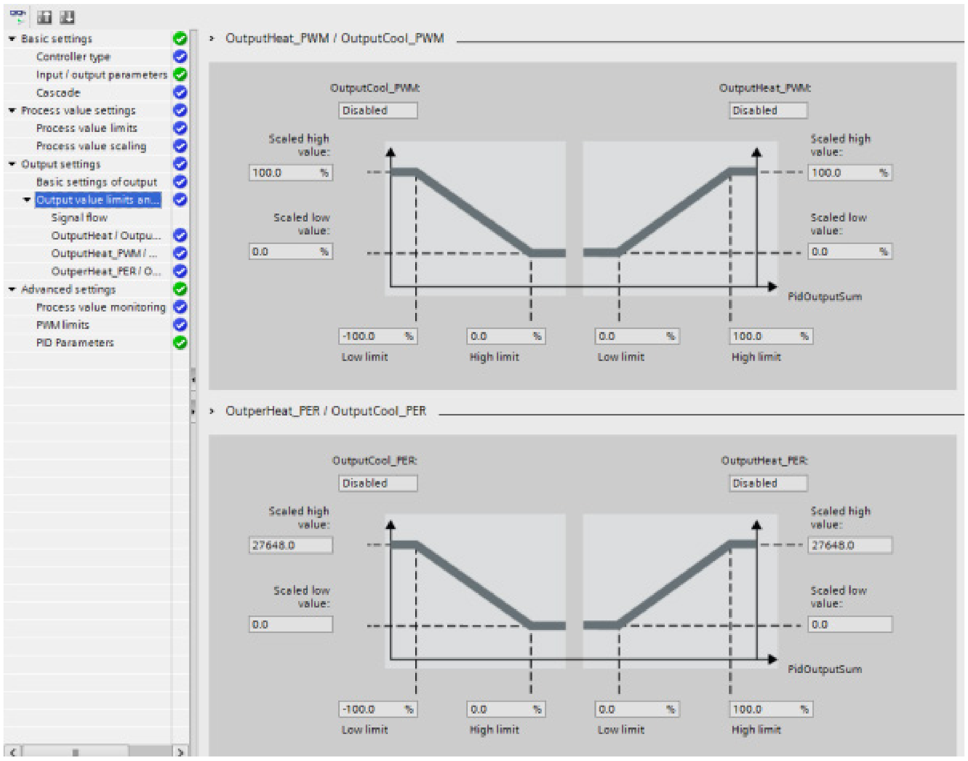
OutputHeat: Enabled

Scaled high value: 100.0 % (OutputCool); 100.0 % (OutputHeat)

Scaled low value: 0.0 % (OutputCool); 0.0 % (OutputHeat)

PidOutputSum

-100.0 % Low limit; 0.0 % High limit; 0.0 % Low limit; 100.0 % High limit



OutputHeat_PWM / OutputCool_PWM

OutputCool_PWM: Disabled

OutputHeat_PWM: Disabled

Scaled high value: 100.0 % (OutputCool_PWM); 100.0 % (OutputHeat_PWM)

Scaled low value: 0.0 % (OutputCool_PWM); 0.0 % (OutputHeat_PWM)

-100.0 % Low limit; 0.0 % High limit; 0.0 % Low limit; 100.0 % High limit

OutperHeat_PER / OutputCool_PER

OutputCool_PER: Disabled

OutputHeat_PER: Disabled

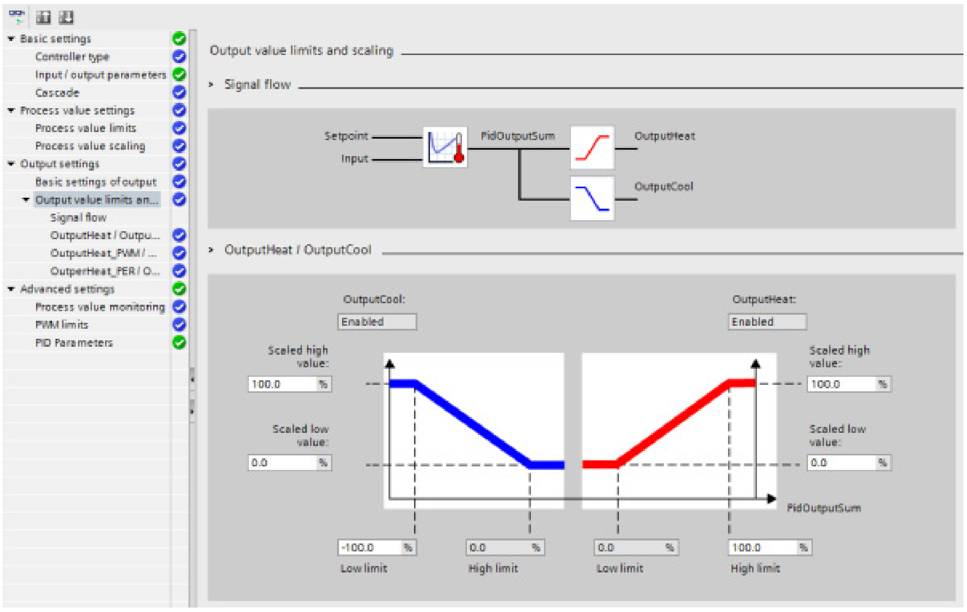
Scaled high value: 27648.0 (OutputCool_PER); 27648.0 (OutputHeat_PER)

Scaled low value: 0.0 (OutputCool_PER); 0.0 (OutputHeat_PER)

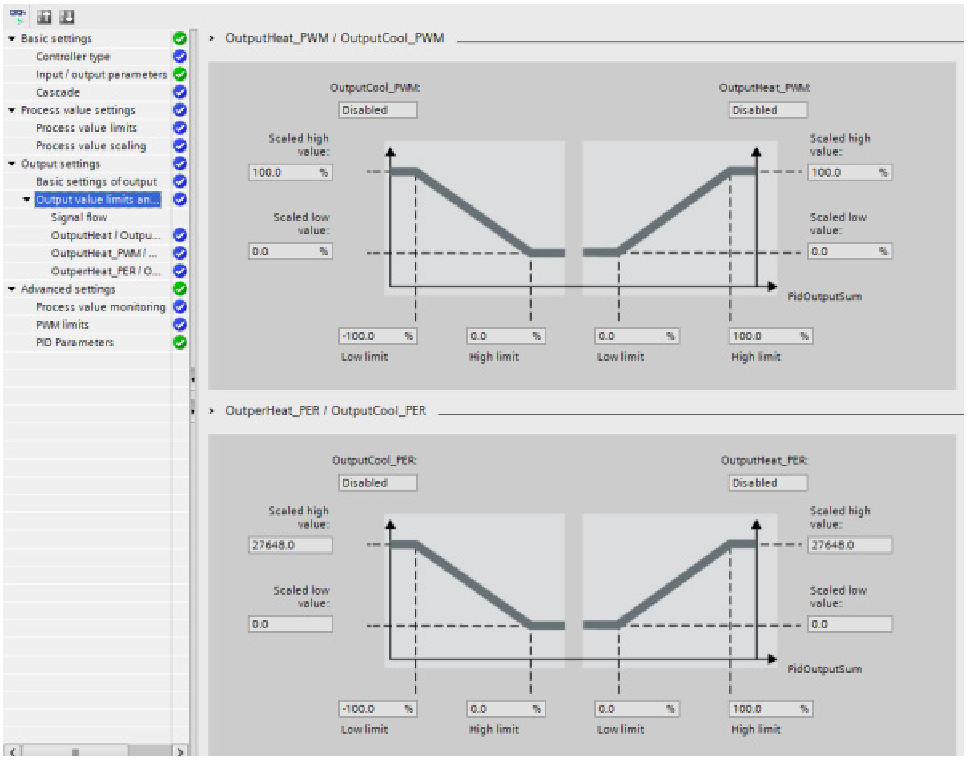
-100.0 % Low limit; 0.0 % High limit; 0.0 % Low limit; 100.0 % High limit

Cooling activation enabled

The figure below shows the ”Output value limits and scaling” section in “Output settings” view with cooling activated (OutputCool_PER and OutputHeat_PWM selected in “Input / output parameters” view; OutputCool and OutputHeat always enabled):



OutputCool: Enabled
OutputHeat: Enabled
Scaled high value: 100.0 % (OutputCool); 100.0 % (OutputHeat)
Scaled low value: 0.0 % (OutputCool); 0.0 % (OutputHeat)
Low limit: -100.0 %; High limit: 0.0 %; Low limit: 0.0 %; High limit: 100.0 %



OutputCool_PWM: Disabled
OutputHeat_PWM: Disabled
Scaled high value: 100.0 %; 100.0 %
Scaled low value: 0.0 %; 0.0 %
Low limit: -100.0 %; High limit: 0.0 %; Low limit: 0.0 %; High limit: 100.0 %

OutputCool_PER: Disabled
OutputHeat_PER: Disabled
Scaled high value: 27648.0; 27648.0
Scaled low value: 0.0; 0.0
Low limit: -100.0 %; High limit: 0.0 %; Low limit: 0.0 %; High limit: 100.0 %

Operation modes

To change the mode of operation manually, the user needs to set the “Mode” in-out parameter of the controller and activate it by setting “ModeActivate” from FALSE to TRUE (rising edge triggered). You must reset “ModeActivate” before the next mode change; it does not reset automatically.

Output parameter “State” shows the current operating mode and is set to the requested “Mode” if possible. The “State” parameter cannot be changed directly; it is only changed through the “Mode” parameter or automatic operating mode changes by the controller.

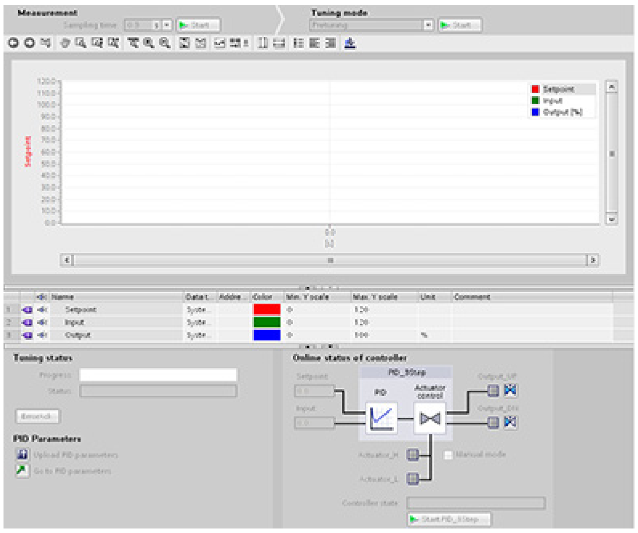
"Mode" / "State"	Name	Description
0	Inactive	The PID_Temp instruction: • Deactivates the PID-algorithm and pulse width modulation • Sets to "0" (FALSE) all controller outputs (OutputHeat, OutputCool, OutputHeat_PWM,OutputCool_PWM, OutputHeat_PER, OutputCool_PER), regardless of configured output limits or offsets. You can reach this mode by setting “Mode” = 0, “Reset” = TRUE, or by error.
1	Pretuning (startup tuning / SUT)	This mode determines the parameters during first start up of the controller. Unlike the PID_Compact, for the PID_Temp, you must select if you require heating tuning, cooling tuning, or both with the “Heat.EnableTuning” and “Cool.EnableTuning” parameters. You can activate "Pretuning" from Inactive, Automatic mode, or Manual mode. If tuning is successful, PID_Temp switches to Automatic mode. If tuning is unsuccessful, the switchover of the operating mode depends on “ActivateRecoverMode”.
2	Fine tuning (tuning in run / TIR)	This mode determines the optimum parameterization of the PID controller at the setpoint. Unlike the PID_Compact, for the PID_Temp, you must select if you require heating tuning or cooling tuning with the “Heat.EnableTuning” and “Cool.EnableTuning” parameters. You can activate "Finetuning" from Inactive, Automatic mode, or Manual mode. If tuning is successful, PID_Temp switches to Automatic mode. If tuning is not successful, the switchover of the operating mode depends on “ActivateRecoverMode”.
3	Automatic mode	In Automatic mode (the standard PID control mode), the result of the PID-algorithm determines the output values. PID_Temp switches to Inactive if an error occurs and “ActivateRecoverMode” = FALSE. If an error occurs and “ActivateRecoverMode” = TRUE, the switchover of the operating mode depends on the error. Refer to PID_Temp instruction ErrorBit parameters (Page 502) for further information.

"Mode" / "State"	Name	Description
4	Manual mode	In this mode, the PID controller scales, limits, and transfers the value of parameter “ManualValue” to the outputs. The PID controller assigns “ManualValue” in the scaling of the PID-algorithm (like“PidOutputSum”), so its value decides if it is effective at the heating or cooling outputs. You can reach this mode by setting “Mode” = 4 or “ManualEnable”= TRUE.
5	Substitute output value with error monitoring (Recover mode)	You can activate this mode by setting "Mode" = 5. The mode is an automatic error reaction of the controller if Automatic mode is active at the moment the error occurs: • SetSubstituteOutput = FALSE (Last valid output value) • SetSubstituteOutput = TRUE (Value stored in parameter "SubstituteOutput") When PID_Temp is in “Automatic mode” and the “ActivateRecoverMode” parameter = TRUE, PID_Temp changes to this mode in the case of the following errors: • “Invalid value at "Input_PER" parameter. Check for an error at the analog input (for example, wire broken)." (ErrorBits = DW#16#0002) • “Invalid value at "Input" parameter. Value is not a number.” (ErrorBits = DW#16#0200) • “Calculation of output value failed. Check the PID parameters.” (ErrorBits = DW#16#0400) • “Invalid value at "Setpoint" parameter. Value is not a number.” (ErrorBits = DW#16#1000) If the error is no longer pending, PID_Temp will switch back to Automatic mode automatically.

10.2.13 Commissioning the PID_Compact and PID_3Step controllers

Use the commissioning editor to configure the PID controller for autotuning at startup and for autotuning during operation. To open the commissioning editor, click the icon on either the instruction or the project navigator.

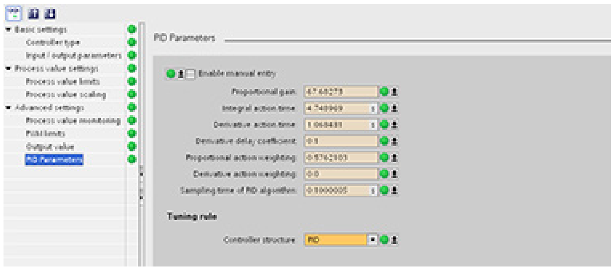
Table 10- 33 Sample commissioning screen (PID_3Step)

 Measurement Sampling time Start Tuning mode Pretuning Start Setpoint Setpoint Input Output [%] Name Data t... Addre... Color Min. Y scale Max. Y scale Unit Comment Setpoint Input Output Tuning status Progress Status Errorack PID Parameters Upload PID parameters Go to PID parameters Online status of controller PID_3Step Setpoint Input PID Actuator control Output_UP Output_DN Actuator_H Actuator_L Manual mode Controller state Start PID_3Step	• Measurement: To display the setpoint, the process value (input value) and the output value in a real-time trend, enter the sample time and click the "Start" button. • Tuning mode: To tune the PID loop, select either "Pre-tuning" or "Fine tuning" (manual) and click the "Start" button. The PID controller runs through multiple phases to calculate system response and update times. The appropriate tuning parameters are calculated from these values. After the completion of the tuning process, you can store the new parameters by clicking the "Upload PID parameters" button in the "PID Parameters" section of the commissioning editor. If an error occurs during tuning, the output value of the PID goes to 0. The PID mode then is set to "inactive" mode. The status indicates the error.

PID start value control

You can edit the actual values of the PID configuration parameters so that the behavior of the PID controller can be optimized in online mode.

Open the "Technology objects" for your PID controller and its "Configuration" object. To access the start value control, click the "eyeglasses icon" in the upper left corner of the dialog:



PID Parameters

Enable manual entry

Proportional gain	47.68273
Integral action time	4.740969
Derivative action time	1.068431
Derivative delay coefficient	0.1
Proportional action weighting	0.5762103
Derivative action weighting	0.0
Sampling time of PID algorithm	0.1000005

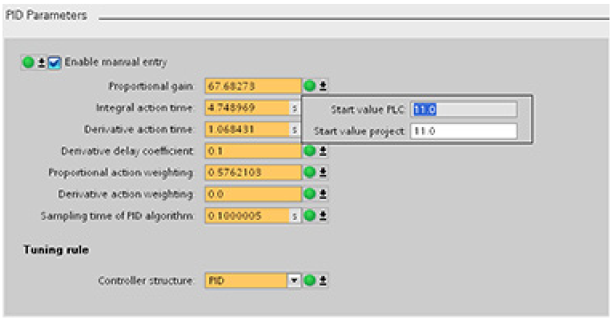
Tuning rule

Controller structure	PID

You can now change the value of any of your PID controller configuration parameters as shown in the figure below.

You can compare the actual value to the project (offline) start value and the PLC (online) start value of each parameter. This is necessary to compare online/offline differences of the

Technology object data block (TO-DB) and to be informed about the values that will be used as current values on the next Stop-to-Start transition of the PLC. In addition, a compare icon gives a visual indication to help easily identify online/offline differences:



PID Parameters

Enable manual entry

Proportional gain:	67.68273
Integral action time:	4.748969 s
Derivative action time:	1.068431 s
Derivative delay coefficient:	0.1
Proportional action weighting:	0.5762103
Derivative action weighting:	0.0
Sampling time of PID algorithm:	0.1000005 s

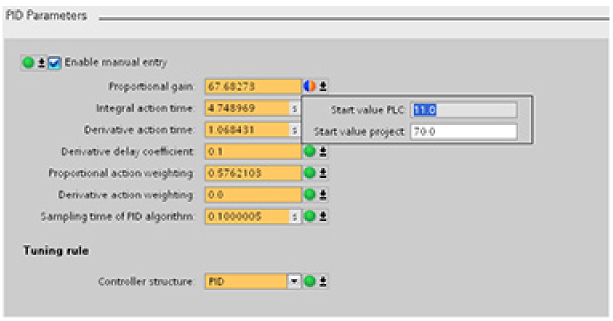
Start value PLC: 11.0
Start value project: 11.0

Tuning rule

Controller structure: PID

The figure above shows the PID parameter screen with compare icons showing which values are different between online and offline projects. A green icon indicates that the values are the same; a blue/orange icon indicates that the values are different.

Additionally, click the parameter button with the downward arrow to open a small window that shows the project (offline) start value and the PLC (online) start value of each parameter:



PID Parameters

Enable manual entry

Proportional gain:	67.68273
Integral action time:	4.748969 s
Derivative action time:	1.068431 s
Derivative delay coefficient:	0.1
Proportional action weighting:	0.5762103
Derivative action weighting:	0.0
Sampling time of PID algorithm:	0.1000005 s

Start value PLC: 11.0
Start value project: 70.0

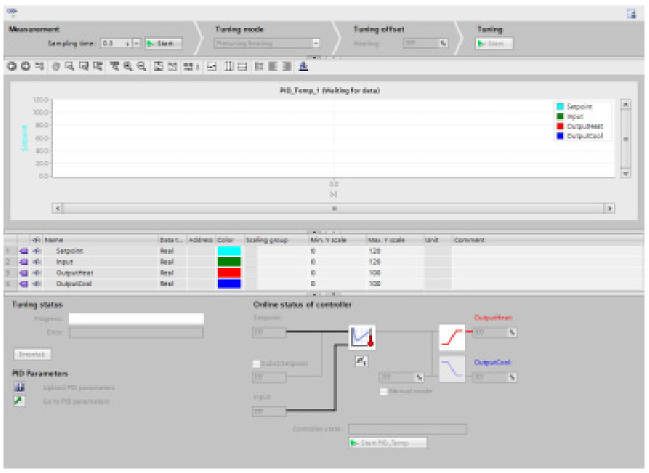
Tuning rule

Controller structure: PID

10.2.14 Commissioning the PID_Temp controller

Use the commissioning editor to configure the PID controller for autotuning at startup and for autotuning during operation. To open the commissioning editor, click the icon on either the instruction or the project navigator.

Table 10- 34 Sample commissioning screen (PID_Temp)

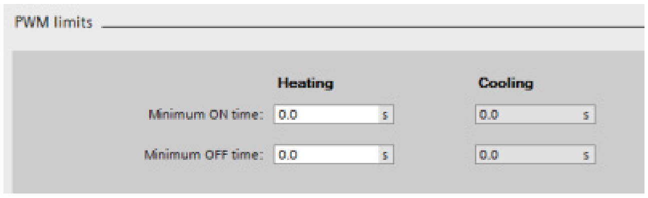
	Measurement: To display the setpoint, the process value (input value) and the output value in a real-time trend, enter the sample time and click the "Start" button. Tuning mode: To tune the PID_Temp loop, select either "Pretuning" or "Finetuning" (manual) and click the "Start" button. The PID controller runs through multiple phases to calculate system response and update times. The appropriate tuning parameters are calculated from these values. After the completion of the tuning process, you can store the new parameters by clicking the "Upload PID parameters" button in the "PID Parameters" section of the commissioning editor. If an error occurs during tuning, the output value of the PID goes to "0". The PID mode then is set to "inactive" mode. The status indicates the error.

PWM limits

Actuators that are controlled with the software PWM function of the PID_Temp may need to be protected from too short pulse durations (for example, a thyristor relay needs to be turned on for more than 20 ms before it can react at all); you assign a minimum on time. The actuator can also neglect short impulses and therefore corrupt the control quality. A minimum off time can be necessary (for example, to prevent overheating).

To show up the PWM limits view, you must open the functional view in the Technology objects (TO) configuration and select “PWM limits” from the “Advanced settings” node in the navigation tree.

If you open the “PWM limits” view in the functional view and activate monitoring (“glasses”button), all controls show the online monitor value from TO-DB with orange background color and multi-value control, and you can edit the values (if configuration conditions are fulfilled; refer to the table below).



PWM limits

	Heating	Cooling
Minimum ON time:	0.0 s	0.0 s
Minimum OFF time:	0.0 s	0.0 s

Setting	TO-DB parameter	Data type	Value range	Description
Minimum on time (heating) [1,2]	"Config.Output.Heat. MinimumOnTime"	Real	100000.0 >= "Config.Output. Heat. MinimumOnTime >= 0.0	A pulse at OutputHeat_PWM" is never shorter than this value.
Minimum off time (heating) [1,2]	"Config.Output.Heat. MinimumOffTime"	Real	100000.0 >= "Config.Output. Heat. MinimumOffTime >= 0.0	A break at OutputHeat_PWM is never shorter than this value.
Minimum on time (cooling) [1,3,4]	"Config.Output.Cool. MinimumOnTime"	Real	100000.0 >= Config.Output. Cool. MinimumOnTime >= 0.0	A pulse at OutputCool_PWM is never shorter than this value.
Minimum off time (cooling) [1,3,4]	"Config.Output.Cool. MinimumOffTime"	Real	100000.0 >= Config.Output. Cool. MinimumOffTime >= 0.0	A break at OutputCool_PWM is never shorter than this value.

1 The field displays "s" (seconds) as the time units.

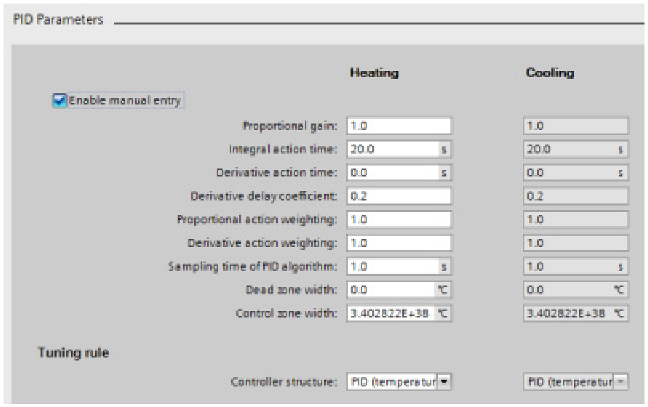
2 If the·selection Output (heating) in “Basic settings” view is not “OutputHeat_PWM” (Config.Output.Heat.Select = TRUE), you should set this value to "0.0".

3 If selection Output (cooling) in “Basic settings” view is not “OutputCool_PWM” (Config.Output.Cool.Select = TRUE), you should set this value to "0.0".

4 Only available if you check “Activate output (cooling)” in “Basic settings” view (Config.ActivateCooling = TRUE).

PID parameters

The “Advanced settings” view, "PID Parameters” section is shown below with the cooling and/or “PID parameterswitchover” feature deactivated.



PID Parameters

	Heating	Cooling
✓ Enable manual entry		
Proportional gain:	1.0	1.0
Integral action time:	20.0 s	20.0 s
Derivative action time:	0.0 s	0.0 s
Derivative delay coefficient:	0.2	0.2
Proportional action weighting:	1.0	1.0
Derivative action weighting:	1.0	1.0
Sampling time of PID algorithm:	1.0 s	1.0 s
Dead zone width:	0.0 ℃	0.0 ℃
Control zone width:	3.402822E+38 ℃	3.402822E+38 ℃

Tuning rule

Controller structure:	PID (temperatur	PID (temperatur

Setting	TO-DB parameter	Data type	Value range	Description
Enable manual entry	"Retain.CtrlParams. SetByUser"	Bool	Bool	You must check this checkbox to enter PID parameters manually.
Proportional gain (heat-ing) [2]	"Retain.CtrlParams. Heat.Gain"	Real	Gain >= 0.0	PID proportional gain for heating
Integral action time (heating) [1,2]	"Retain.CtrlParams. Heat.Ti"	Real	100000.0 >= Ti >= 0.0	PID integral action for heating.
Derivative action time (heating) [1,2]	"Retain.CtrlParams. Heat.Td"	Real	100000.0 >= Td >= 0.0	PID derivative action time for heating.
Derivative delay coeffi-cient(heatin g) [2]	"Retain.CtrlParams. Heat.TdFiltRatio"	Real	TdFiltRatio >= 0.0	PID derivative delay coefficient for heating that defines the deriva-tive lag time as coeffi-cient from the PID derivative time.
Proportional action weighting(h eating) [2]	"Retain.CtrlParams. Heat.PWeighting"	Real	1.0 >=PWeighting >= 0.0	Weighting of the PID proportional gain for heating in either direct- or loopback- control path.
Derivative action weighting (heating) [2]	"Retain.CtrlParams. Heat.DWeighting"	Real	1.0 >=DWeighting >= 0.0	Weighting of the PID derivative part for heat-ing in either direct- or loopback- control path.

Setting	TO-DB parameter	Data type	Value range	Description
Sampling time of PID algorithm (heating) [1,2]	"Retain.CtrlParams. Heat.Cycle"	Real	100000.0 >=Cycle > 0.0	Internal call cycle of the PID controller for heat-ing. Rounded to an integer multiple of the FB call cycle time.
Deadband width(heatin g) [2,3]	"Retain.CtrlParams. Heat.DeadZone"	Real	DeadZone>= 0.0	Width of the deadband for heating control deviation.
Control Zone (heat-ing)[2,3]	"Retain.CtrlParams. Heat.ControlZone"	Real	ControlZone> 0.0	Width of the control deviation zone for heat-ing where PID control is active. If control devia-tion leaves this range, output is switched to maximum output val-ues. Default value is "MaxReal" so control zone is deactivated as long as autotuning is not executed. Value "0.0" is prohibit-ed for Control Zone; with the value "0.0", PID_Temp behaves like a two-position controller that is always heating or cooling at full power.

Setting	TO-DB parameter	Data type	Value range	Description
Controller structure (heating)	"PIDSelfTune.SUT. TuneRuleHeat", "PIDSelfTune.TIR. TuneRuleHeat"	Int	"PIDSelf-Tune.SUT. TuneRuleHeat" = 0..2, "PIDSelf-Tune.TIR. TuneRuleHeat" = 0..5	You can select the tuning algorithm for heating. Possible selections: • PID (Temperature) (=default) ("PIDSelfTune.SUT. TuneRuleHeat" = 2) ("PIDSelfTune.TIR. TuneRuleHeat" = 0) • PID ("PIDSelfTune.SUT. TuneRuleHeat" = 0) ("PIDSelfTune.TIR. TuneRuleHeat" = 0) • PI ("PIDSelfTune.SUT. TuneRuleHeat" = 1) ("PIDSelfTune.TIR. TuneRuleHeat" = 4) Any other combination shows "User defined", but "User defined" is not provided by default. "PID (Temperature)" is new for PID_Temp, with a specific pretuning (SUT) method for temperature processes.
Proportional gain (cooling) [4]	"Retain.CtrlParams. Cool.Gain"	Real	Gain >= 0.0	PID proportional gain for cooling
Integral action time (cooling) [1,4]	"Retain.CtrlParams. Cool.Ti"	Real	100000.0 >=Ti >= 0.0	PID integral action for cooling
Derivative action time (cooling) [1,4]	"Retain.CtrlParams. Cool.Td"	Real	100000.0 >=Td >= 0.0	PID derivative action time for cooling
Derivative delay coefficient (cooling) [4]	Retain.CtrlParams. Cool.TdFiltRatio"	Real	TdFiltRatio>= 0.0	PID derivative delay coefficient for cooling that defines the derivative lag time as a coefficient from the PID derivative time.

Setting	TO-DB parameter	Data type	Value range	Description
Proportional action weighting (cooling) [4]	"Retain.CtrlParams. Cool.PWeighting"	Real	1.0 >=PWeighting >= 0.0	Weighting of the PID proportional gain for cooling in either the direct- or loopback- control path.
Derivative action weighting (cooling) [4]	Retain.CtrlParams. Cool.DWeighting"	Real	1.0 >=DWeighting >= 0.0	Weighting of the PID derivative part for cooling in either the direct- or loopback- control path.
Sampling time of PID algorithm (cooling) [1,4]	"Retain.CtrlParams. Cool.Cycle"	Real	100000.0 >=Cycle > 0.0	Internal call cycle of the PID controller for cooling. Rounded to an integer multiple of the FB call cycle time.
Deadband width (cooling) [3,4]	"Retain.CtrlParams. Cool.DeadZone"	Real	DeadZone>= 0.0	Width of the deadband for cooling control deviation
Control Zone (cooling) [3,4]	"Retain.CtrlParams. Cool.ControlZone"	Real	ControlZone> 0.0	Width of the control deviation zone for cooling where PID control is active. If control deviation leaves this range, output is switched to maximum output values. Default value is "MaxReal" so control zone is deactivated as long as autotuning is not executed. Value "0.0" is prohibited for Control Zone; with the value "0.0", PID_Temp behaves like a two-position controller that is always heating or cooling at full power.

Setting	TO-DB parameter	Data type	Value range	Description
Controller structure (cooling)	"PIDSelfTune.SUT. TuneRuleCool", "PIDSelfTune.TIR. TuneRuleCool"	Int	"PIDSelf-Tune.SUT. TuneRuleHeat" = 0..2, "PIDSelf-Tune.TIR. TuneRuleHeat" = 0..5	You can select the tuning algorithm for cooling. Possible selections: • PID (Temperature) (=default) ("PIDSelfTune.SUT. TuneRuleCool" = 2) ("PIDSelfTune.TIR. TuneRuleCool = 0) • PID ("PIDSelfTune.SUT. TuneRuleCool" = 0) ("PIDSelfTune.TIR. TuneRuleCool" = 0) • PI ("PIDSelfTune.SUT. TuneRuleCool" = 1) ("PIDSelfTune.TIR. TuneRuleCool" = 4) Any other combination shows “User defined”, but “User defined” is not provided by default. “PID (Temperature)” is new for PID_Temp, with a specific pretun-ing (SUT) method for temperature processes. Only available if you check/select the follow-ing items: “Activate output (cooling)” in “Basic settings” view ("Con-fig.ActivateCooling" = TRUE), and “PID pa-rameter switchover” in “Output settings” view (Con-fig.AdvancedCooling = TRUE).

Setting	TO-DB parameter	Data type	Value range	Description

[1] The field displays "s" (seconds) as the time units.

[2] Only available if you check "Enable manual entry" in PID parameters ("Retain.CtrlParams.SetByUser" = TRUE).

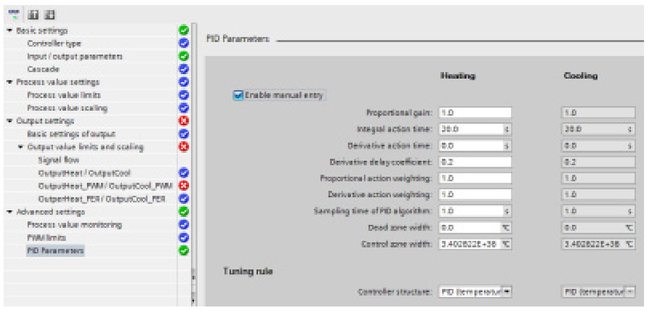
[3] Unit of measurement is displayed at the end of the field as selected in "Basic settings" view.

[4] Only available if you check/select the following items: "Enable manual entry" in PID parameters ("Retain.CtrlParams.SetByUser" = TRUE), "Activate output (cooling)" in "Basic settings" view ("Config.ActivateCooling" = TRUE), and "PID parameter switchover" in "Output settings" view (Config.AdvancedCooling = TRUE).

PID start value control

You can edit the actual values of the PID configuration parameters so that the behavior of the PID controller can be optimized in online mode.

Open the "Technology objects" for your PID controller and its "Configuration" object. To access the start value control, click the "eyeglasses icon" in the upper left corner of the dialog:



PID Parameters

☑ Enable manual entry

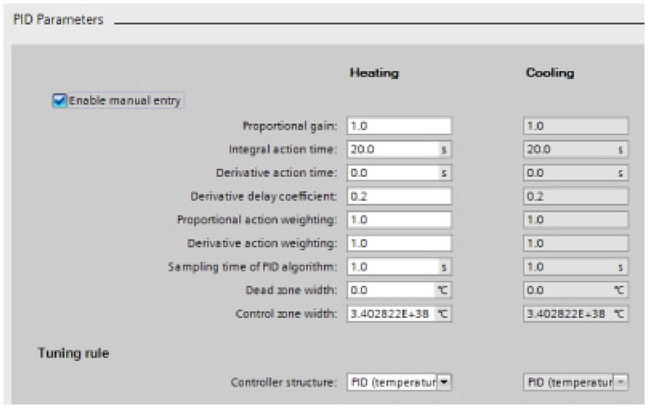
	Heating	Cooling
Proportional gain:	1.0	1.0
Integral action time:	20.0 s	20.0 s
Derivative action time:	0.0 s	0.0 s
Derivative delay coefficient:	0.2	0.2
Proportional action weighting:	1.0	1.0
Derivative action weighting:	1.0	1.0
Sampling time of PID algorithm:	1.0 s	1.0 s
Dead zone width:	0.0 °C	0.0 °C
Control zone width:	3.402822E+38 °C	3.402822E+38 °C

Tuning rule

	Heating	Cooling
Controller structure:	PID (temperatu	PID (temperatu

You can now change the value of any of your PID controller configuration parameters as shown in the figure below.

You can compare the actual value to the project (offline) start value and the PLC (online) start value of each parameter. This is necessary to compare online/offline differences of the Technology object data block (TO-DB) and to be informed about the values that will be used as current values on the next Stop-to-Start transition of the PLC. In addition, a compare icon gives a visual indication to help easily identify online/offline differences:


PID Parameters

✓ Enable manual entry

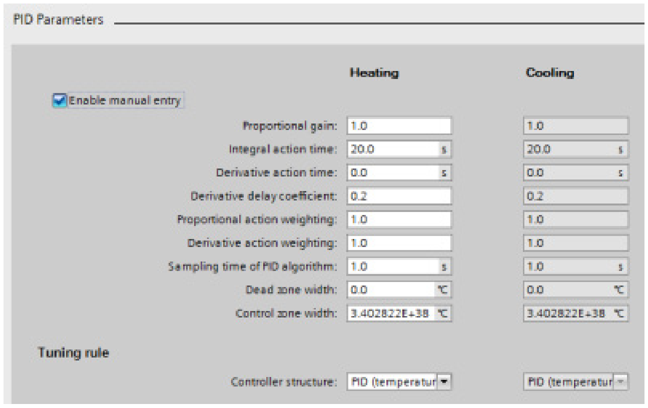
	Heating	Cooling
Proportional gain:	1.0	1.0
Integral action time:	20.0 s	20.0 s
Derivative action time:	0.0 s	0.0 s
Derivative delay coefficient:	0.2	0.2
Proportional action weighting:	1.0	1.0
Derivative action weighting:	1.0	1.0
Sampling time of PID algorithm:	1.0 s	1.0 s
Dead zone width:	0.0 ℃	0.0 ℃
Control zone width:	3.402822E+38 ℃	3.402822E+38 ℃

Tuning rule

Controller structure:	PID (temperatur	PID (temperatur

The figure above shows the PID parameter screen with compare icons showing which values are different between online and offline projects. A green icon indicates that the values are the same; a blue/orange icon indicates that the values are different.

Additionally, click the parameter button with the downward arrow to open a small window that shows the project (offline) start value and the PLC (online) start value of each parameter:


PID Parameters

✓ Enable manual entry

	Heating	Cooling
Proportional gain:	1.0	1.0
Integral action time:	20.0 s	20.0 s
Derivative action time:	0.0 s	0.0 s
Derivative delay coefficient:	0.2	0.2
Proportional action weighting:	1.0	1.0
Derivative action weighting:	1.0	1.0
Sampling time of PID algorithm:	1.0 s	1.0 s
Dead zone width:	0.0 ℃	0.0 ℃
Control zone width:	3.402822E+38 ℃	3.402822E+38 ℃

Tuning rule

Controller structure:	PID (temperatur	PID (temperatur

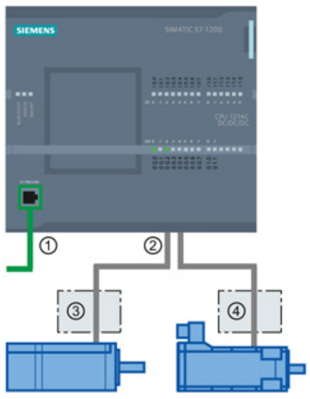
10.3 Motion control

The CPU provides motion control functionality for the operation of stepper motors and servo motors with pulse interface. The motion control functionality takes over the control and monitoring of the drives.

- The "Axis" technology object configures the mechanical drive data, drive interface, dynamic parameters, and other drive properties.
- You configure the pulse and direction outputs of the CPU for controlling the drive.
- Your user program uses the motion control instructions to control the axis and to initiate motion tasks.
- Use the PROFINET interface to establish the online connection between the CPU and the programming device. In addition to the online functions of the CPU, additional commissioning and diagnostic functions are available for motion control.

Note

Changes that you make to the motion control configuration and download in RUN mode do not take effect until the CPU transitions from STOP to RUN mode.



①	PROFINET
②	Pulse and direction outputs
③	Power section for stepper motor
④	Power section for servo motor

The DC/DC/DC variants of the CPU S7-1200 have onboard outputs for direct control of drives. The relay variants of the CPU require the signal board with DC outputs for drive control.

A signal board (SB) expands the onboard I/O to include a few additional I/O points. An SB with two digital outputs can be used as pulse and direction outputs to control one motor. An SB with four digital outputs can be used as pulse and direction outputs to control two motors. Built-in relay outputs cannot be used as pulse outputs to control motors. Whether you use onboard I/O or SB I/O or a combination of both, you can have a maximum number of four pulse generators.

The four pulse generators have default I/O assignments; however, they can be configured to any digital output on the CPU or SB. Pulse generators on the CPU cannot be assigned to SMs or to distributed I/O.

Note

Pulse-train outputs cannot be used by other instructions in the user program

When you configure the outputs of the CPU or signal board as pulse generators (for use with the PWM or motion control instructions), the corresponding output addresses no longer control the outputs. If your user program writes a value to an output used as a pulse generator, the CPU does not write that value to the physical output.

Table 10- 35 Maximum number of controllable drives

Type of CPU		Onboard I/O; No SB installed		With an SB (2 x DC outputs)		With an SB (4 x DC outputs)	
		With direction	Without direction	With direction	Without direction	With direction	Without direction
CPU 1211C	DC/DC/DC	2	4	3	4	4	4
	AC/DC/RLY	0	0	1	2	2	4
	DC/DC/RLY	0	0	1	2	2	4
CPU 1212C	DC/DC/DC	3	4	3	4	4	4
	AC/DC/RLY	0	0	1	2	2	4
	DC/DC/RLY	0	0	1	2	2	4
CPU 1214C	DC/DC/DC	4	4	4	4	4	4
	AC/DC/RLY	0	0	1	2	2	4
	DC/DC/RLY	0	0	1	2	2	4
CPU 1215C	DC/DC/DC	4	4	4	4	4	4
	AC/DC/RLY	0	0	1	2	2	4
	DC/DC/RLY	0	0	1	2	2	4
CPU 1217C	DC/DC/DC	4	4	4	4	4	4

Note

The maximum number of pulse generators is four.

Whether you use onboard I/O, SB I/O, or a combination of both, you can have a maximum number of four pulse generators.

Table 10- 36 CPU output: maximum frequency

CPU	CPU output channel	Pulse and direction output	A/B, quadrature, up/down, and pulse/direction
1211C	Qa.0 to Qa.3	100 kHz	100 kHz
1212C	Qa.0 to Qa.3	100 kHz	100 kHz
	Qa.4, Qa.5	20 kHz	20 kHz
1214C and 1215C	Qa.0 to Qa.3	100kHz	100kHz
	Qa.4 to Qb.1	20 kHz	20 kHz
1217C	DQa.0 to DQa.3 (.0+, .0- to .3+, .3-)	1 MHz	1 MHz
	DQa.4 to DQb.1	100 kHz	100 kHz

Table 10- 37 SB signal board output: maximum frequency (optional board)

SB signal board	SB output channel	Pulse and direction output	A/B, quadrature, up/down, and pulse/direction
SB 1222, 200 kHz	DQe.0 to DQe.3	200kHz	200 kHz
SB 1223, 200 kHz	DQe.0, DQe.1	200kHz	200 kHz
SB 1223	DQe.0, DQe.1	20 kHz	20 kHz

Table 10- 38 Limit frequencies of pulse outputs

Pulse output	Frequency
Onboard	4 PTO: 2 Hz ≤ f ≤ 1 MHz, 4 PTO: 2 Hz ≤ f ≤ 100 kHz, or any combination of these values for 4 PTOs.[1] [2]
Standard SB	2 Hz ≤ f ≤ 20 kHz
High-speed SBs	2 Hz ≤ f ≤ 200 kHz

[1] See the table below for four possible CPU 1217C output speed combinations.

[2] See the table below for four possible CPU 1211C, CPU 1212C, CPU 1214C, or CPU 1215C output speed combinations.

Example: CPU 1217C pulse output speed configurations

Note

The CPU 1217C can generate pulse outputs up to 1 MHz, using the onboard differential outputs.

The examples below show four possible output speed combinations:

- Example 1: 4 - 1 MHz PTOs, no direction output
- Example 2: 1 - 1 MHz, 2 - 100 kHz, and 1 - 20 kHz PTOs, all with direction output
- Example 3: 4 - 200 kHz PTOs, no direction output
- Example 4: 2 - 100 kHz PTOs and 2 - 200 kHz PTOs, all with direction output

P = Pulse **D = Direction**		**CPU on-board outputs**										**High-speed SB outputs**				**Standard SB out-puts**	
		1 MHz Outputs (Q)				100 kHz Outputs (Q)						200 kHz Outputs (Q)				20 kHz Outputs (Q)	
		0.0+	0.1+	0.2+	0.3+	0.4	0.5	0.6	0.7	1.0	1.1	4.0	4.1	4.2	4.3	4.0	4.1
		0.0-	0.1-	0.2-	0.3-												
Ex. 1: 4 - 1 MHz (no direction output)	PTO1	P															
	PTO2		P														
	PTO3			P													
	PTO4				P												
Ex. 2: 1 - 1 MHz; 2 - 100 and 1 - 20 kHz (all with direction output)	PTO1	P	D														
	PTO2					P	D										
	PTO3							P	D								
	PTO4															P	D
Ex. 3: 4 - 200 kHz (no direction output)	PTO1											P					
	PTO2												P				
	PTO3													P			
	PTO4														P		
Ex. 4: 2 - 100 kHz; 2 - 200 kHz (all with direction output)	PTO1					P	D										
	PTO2							P	D								
	PTO3											P	D				
	PTO4													P	D		

Example: CPU 1211C, CPU 1212C, CPU 1214C, and CPU 1215C pulse output speed configurations

The examples below show four possible output speed combinations:

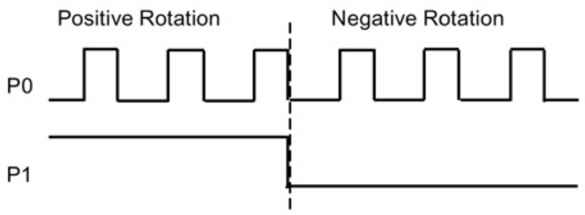
- Example 1: 4 - 100 kHz PTOs, no direction output
- Example 2: 2 - 100 kHz PTOs and 2 - 20 kHz PTOs, all with direction output
- Example 3: 4 - 200 kHz PTOs, no direction output
- Example 4: 2 - 100 kHz PTOs and 2 - 200 kHz PTOs, all with direction output

P = Pulse **D = Direction**		**CPU on-board outputs**										**High-speed SB outputs**				**Low-speed SB outputs**	
		100 kHz Outputs (Q)				20 kHz Outputs (Q)						200 kHz Outputs (Q)				20 kHz Outputs (Q)	
		0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	1.0	1.1	4.0	4.1	4.2	4.3	4.0	4.1
		CPU 1211C															
		CPU 1212C				CPU 1212C											
		CPU 1214C				CPU 1214C		CPU 1214C									
		CPU 1215C				CPU 1215C		CPU 1215C									
Ex. 1: 4 - 100 kHz (no direction output)	PTO1	P															
	PTO2		P														
	PTO3			P													
	PTO4				P												
Ex. 2: 2 - 100 kHz; 2 - 20 kHz (all with direction output)	PTO1	P	D														
	PTO2			P	D												
	PTO3					P	D										
	PTO4							P	D								
Ex. 3: 4 - 200 kHz (no direction output)	PTO1											P					
	PTO2												P				
	PTO3													P			
	PTO4														P		
Ex. 4: 2 - 100 kHz; 2 - 200 kHz (all with direction output)	PTO1	P	D														
	PTO2			P	D												
	PTO3											P	D				
	PTO4													P	D		

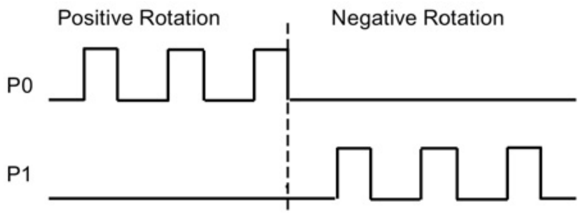
10.3.1 Phasing

You have four options for the "Phasing" interface to the stepper/servo drive. These options are as follows:

- PTO (pulse A and direction B): If you select a PTO (pulse A and direction B) option, then one output (P0) controls the pulsing and one output (P1) controls the direction. P1 is high (active) if pulsing is in the positive direction. P1 is low (inactive) if pulsing is in the negative direction:

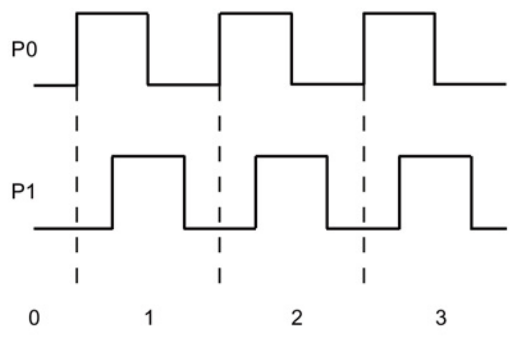
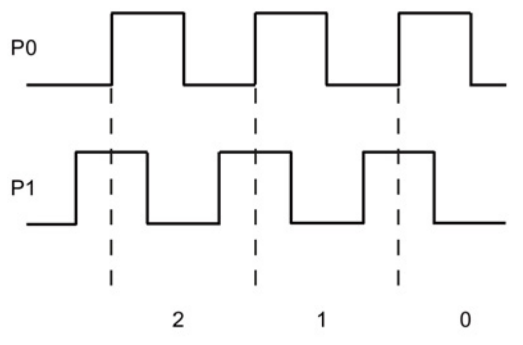


- PTO (count up A and count down B): If you select a PTO (count up A and count down B) option, then one output (P0) pulses for positive directions and a different output (P1) pulses for negative directions:



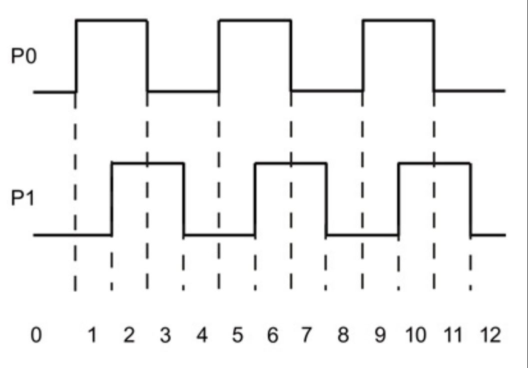
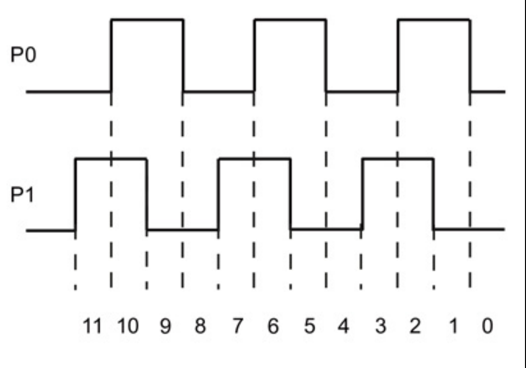
- PTO (A/B phase-shifted): If you select a PTO (A/B phase-shifted) option, then both outputs pulse at the speed specified, but 90 degrees out-of-phase. It is a 1X configuration, meaning one pulse is the amount of time between positive transitions of P0. In this case, the direction is determined by which output transitions high first. P0 leads P1 for the positive direction. P1 leads P0 for the negative direction.

  The number of pulses generated is based upon the number of 0 to 1 transitions of Phase A. The phase relationship determines the direction of movement:

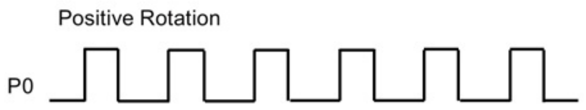
PTO (A/B phase-shifted)	
Phase A leads phase B (positive movement)	Phase A lags phase B (negative movement)
	
Number of pulses	Number of pulses

- PTO (A/B phase-shifted - fourfold): If you select a PTO (A/B phase-shifted - fourfold) option, then both outputs pulse at the speed specified, but 90 degrees out-of-phase. The fourfold is a 4X configuration, meaning one pulse is the transition of each output (both positive and negative). 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.

  Fourfold is based upon positive and negative transitions of both Phase A and Phase B. You configure the number of transitions. The phase relationship (A leading B or B leading A) determines the direction of movement.

PTO (A/B phase-shifted - fourfold)	
Phase A leads phase B (positive movement)	Phase A lags phase B (negative movement)
	
Number of pulses	Number of pulses

- PTO (pulse and direction (direction de-selected)): If you de-select the direction output in a PTO (pulse and direction (direction de-selected)), then output (P0) controls the pulsing. Output P1 is not used and is available for other program uses. Only positive motion commands are accepted by the CPU in this mode. Motion control 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 (one output) is shown in the figure below (assuming positive polarity):



10.3.2 Configuring a pulse generator

1. Add a Technology object:
   – In the Project tree, expand the node "Technology Objects" and select "Add new object".
   – Select the "Axis" icon (rename if required) and click "OK" to open the configuration editor for the axis object.
   – Display the "Select PTO for Axis Control" properties under the "Basic parameters" and select the desired pulse.

   Note

   If the PTO has not been previously configured in the CPU Properties, the PTO is configured to use one of the onboard outputs.

   If you use an output signal board, then select the "Device configuration" button to go to the CPU Properties. Under "Parameter assignment", in the "Pulse options", configure the output source to a signal board output.

   – Configure the remaining Basic and Extended parameters.
2. Program your application: Insert the MC_Power instruction in a code block.
   – For the Axis input, select the axis technology object that you created and configured.
   – Setting the Enable input to TRUE allows the other motion instructions to function.
   – Setting the Enable input FALSE cancels the other motion instructions.

   Note

   Include only one MC_Power instruction per axis.

   ---

3. Insert the other motion instructions to produce the required motion.

Note

Configuring a pulse generator to signal board outputs: Select the "Pulse generators (PTO/PWM)" properties for a CPU (in Device configuration) and enable a pulse generator. Two pulse generators are available for each S7-1200 CPU V1.0, V2.0, V2.1, and V2.2. S7-1200 CPU V3.0 and V4.0 CPUs have four pulse generators available. In this same configuration area under "Pulse options", select Pulse generator used as: "PTO".

Note

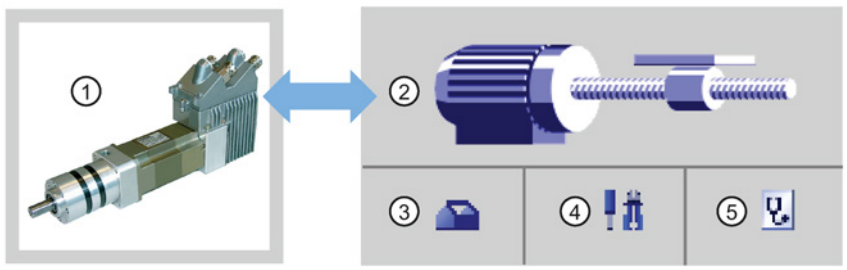
The CPU calculates motion tasks in "slices" or segments of 10 ms. As one slice is being executed, the next slice is waiting in the queue to be executed. If you interrupt the motion task on an axis (by executing another new motion task for that axis), the new motion task may not be executed for a maximum of 20 ms (the remainder of the current slice plus the queued slice).

10.3.3 Open loop motion control

10.3.3.1 Configuring the axis

You connect the open loop axis on the PLC and the drive through a PTO (Pulse Train Output).

STEP 7 provides the configuration tools, the commissioning tools, and the diagnostic tools for the "Axis" technology object.



①	Drive	④	Commissioning
②	Technology object	⑤	Diagnostics
③	Configuration		

Note

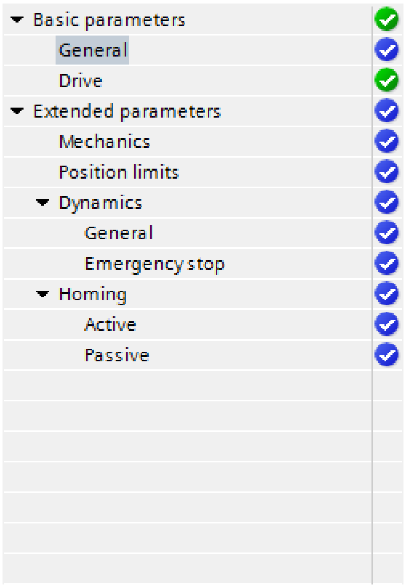
For CPU firmware releases V2.2 and earlier, the PTO requires the internal functionality of a high-speed counter (HSC). This means the corresponding HSC cannot be used elsewhere.

The assignment between PTO and HSC is fixed. If PTO1 is activated, it will be connected to HSC1. If PTO2 is activated, it will be connected to HSC2. You cannot monitor the current value (for example, in ID1000) when pulses are occurring.

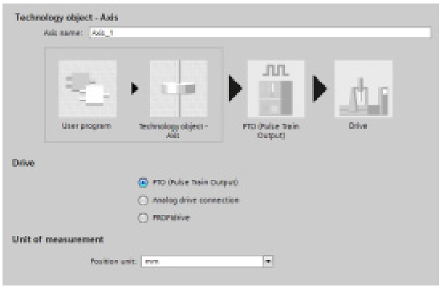
S7-1200 V3.0 and later CPUs do not have this restriction; all HSCs remain available for program use when pulse outputs are configured in these CPUs.

Table 10- 39 STEP 7 tools for motion control

Tool	Description
Configuration	Configures the following properties of the "Axis" technology object: • Selection of the PTO to be used and configuration of the drive interface • Properties of the mechanics and the transmission ratio of the drive (or machine or system) • Properties for position limits, dynamics, and homing Save the configuration in the data block of the technology object.
Commissioning	Tests the function of your axis without having to create a user program. When the tool is started, the control panel will be displayed. The following commands are available on the control panel: • Enable and disable axis • Move axis in jog mode • Position axis in absolute and relative terms • Home axis • Acknowledge errors The velocity and the acceleration / deceleration can be specified for the motion commands. The control panel also shows the current axis status.
Diagnostics	Monitors of the current status and error information for the axis and drive.



The tree selector for the PTO axis does not include the Encoder, Modulo, Position monitoring, and Control loop configuration menus.



Technology object - Axis

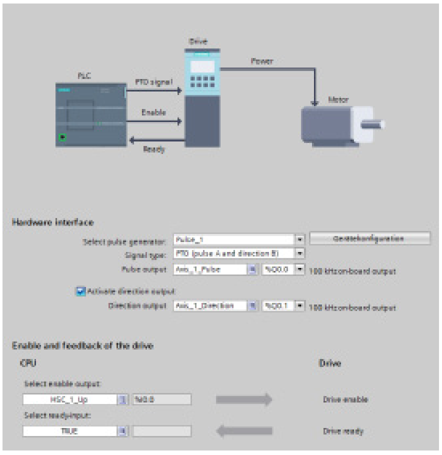
Axis name: Axis_1

Drive

[✓ PTO (Pulse Train Output)] Analog drive connection PROFIdrive

Unit of measurement

Position unit: mm



Hardware interface

Select pulse generator: Pulse_1 Gerätekonfiguration

Signal type: PTO (pulse A and direction B)

Pulse output: Axis_1_Pulse %Q0.0 100 kHz onboard output

[✓] Activate direction output

Direction output: Axis_1_Direction %Q0.1 100 kHz onboard output

Enable and feedback of the drive

CPU

Select enable output: HSC_1_Up %Q0.0

Select ready input: TRUE

Drive

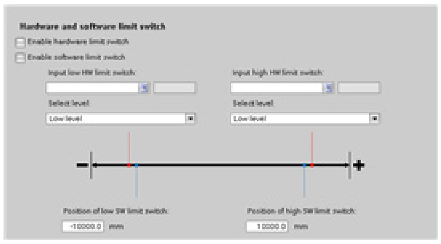
Drive enable

Drive ready

After you create the technology object for the axis, you configure the axis by defining the basic parameters, such as the PTO and the configuration of the drive interface. You also configure the other properties of the axis, such as position limits, dynamics, and homing.

Note

You may have to adapt the values of the input parameters of motion control instructions to the new dimension unit in the user program.

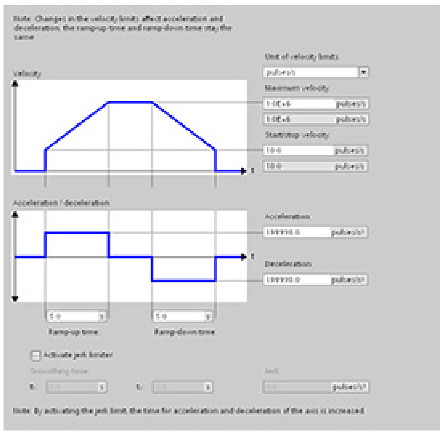

Hardware and software limit switch

☐ Enable hardware limit switch

☐ Enable software limit switch

Input low HW limit switch:	Input high HW limit switch:
Select level: Low level	Select level: Low level
Position of low SW limit switch: -10000.0 mm	Position of high SW limit switch: 10000.0 mm

Configure the properties for the drive signals, drive mechanics, and position monitoring (hardware and software limit switches).


Note: Changes in the velocity limits affect acceleration and deceleration, the ramp-up time and ramp-down time stay the same

Unit of velocity limits: pulses/s

Maximum velocity: 1.0E+6 pulses/s; 1.0E+6 pulses/s

Start/stop velocity: 10.0 pulses/s; 10.0 pulses/s

Acceleration: 199998.0 pulses/s²

Deceleration: 199998.0 pulses/s²

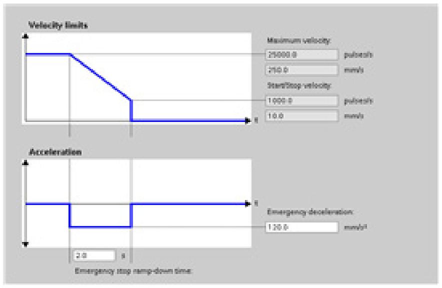
Ramp-up time: 5.0 s

Ramp-down time: 5.0 s

☐ Activate jerk limiter

Note: By activating the jerk limit, the time for acceleration and deceleration of the axis is increased.

You configure the motion dynamics and the behavior of the emergency stop command.


Velocity limits

Maximum velocity: 25000.0 pulses/s; 250.0 mm/s

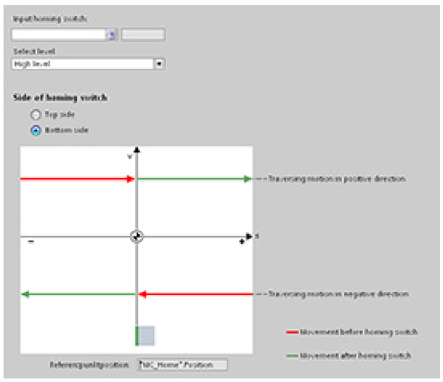
Start/Stop velocity: 1000.0 pulses/s; 10.0 mm/s

Acceleration

Emergency deceleration: 120.0 mm/s²

Emergency stop ramp-down time: 2.0 s

You also configure the homing behavior (passive and active).


Input homing switch:

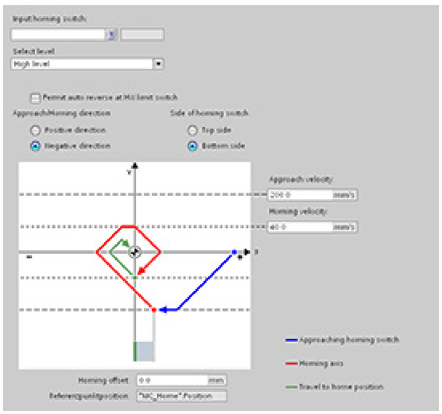
Select level: High level

Side of homing switch

○ Top side [✓ ● Bottom side]

Reference point position: "MC_Home".Position

v
s
Traversing motion in positive direction
Traversing motion in negative direction
Movement before homing switch
Movement after homing switch


Input homing switch:

Select level: High level

☐ Permit auto reverse at HW limit switch

Approach/Homing direction: ○ Positive direction [✓ ● Negative direction]

Side of homing switch: ○ Top side [✓ ● Bottom side]

Approach velocity: 200.0 mm/s

Homing velocity: 40.0 mm/s

Homing offset: 0.0 mm

Reference point position: "MC_Home".Position

Use the "Commissioning" control panel to test the functionality independently from your user program.

Click the "Startup" icon to commission the axis.

The control panel shows the current status of the axis. Not only can you enable and disable the axis, but you can also test the positioning of the axis (both in absolute and relative terms) and can specify the velocity, acceleration and deceleration. You can also test the homing and jogging tasks. The control panel also allows you to acknowledge errors.

10.3.3.2 Commissioning

"Status and error bits" diagnostic function

Use the "Status and error bits" diagnostic function to monitor the most important status and error messages for the axis. The diagnostic function display is available in online mode in "Manual control" mode and in "Automatic control" when the axis is active.

Table 10- 40 Status of the axis

Status	Description
Enabled	The axis is enabled and ready to be controlled via motion control tasks. (Tag of technology object: <Axis name>.StatusBits.Enable)
Homed	The axis is homed and is capable of executing absolute positioning tasks of motion control instruction "MC_MoveAbsolute". The axis does not have to be homed for relative homing. Special situations: • During active homing, the status is FALSE. • If a homed axis undergoes passive homing, the status is set to TRUE during passive homing. (Tag of technology object: <Axis name>.StatusBits.HomingDone)
Error	An error has occurred in the "Axis" technology object. More information about the error is available in automatic control at the ErrorID and ErrorInfo parameters of the motion control instructions. In manual mode, the "Last error" field of the control panel displays detailed information about the cause of error. (Tag of technology object: <Axis name>.StatusBits.Error)
Control panel active	The "Manual control" mode was enabled in the control panel. The control panel has control priority over the "Axis" technology object. The axis cannot be controlled from the user program. (Tag of technology object: <Axis name>.StatusBits.ControlPanelActive)

Table 10- 41 Drive status

Status	Description
Drive ready	The drive is ready for operation. (Tag of technology object: <Axis name>.StatusBits.DriveReady)
Error	The drive has reported an error after failure of its ready signal. (Tag of technology object: <Axis name>.ErrorBits.DriveFault)

Table 10- 42 Status of the axis motion

Status	Description
Standstill	The axis is at a standstill. (Tag of technology object: <Axis name>.StatusBits.StandStill)
Accelerating	The axis accelerates. (Tag of technology object: <Axis name>.StatusBits.Acceleration)
Constant velocity	The axis travels at constant velocity. (Tag of technology object: <Axis name>.StatusBits.ConstantVelocity)
Decelerating	The axis decelerates (slows down). (Tag of technology object: <Axis name>.StatusBits.Deceleration)

Table 10- 43 Status of the motion mode

Status	Description
Positioning	The axis executes a positioning task of motion control instruction "MC_MoveAbsolute" or "MC_MoveRelative" or of the control panel. (Tag of technology object: <Axis name>.StatusBits.PositioningCommand)
Speed Command	The axis executes a task at set speed of motion control instruction "MC_MoveVelocity" or "MC_MoveJog" or of the control panel. (Tag of technology object: <Axis name>.StatusBits.SpeedCommand)
Homing	The axis executes a homing task of motion control instruction "MC_Home" or the control panel. (Tag of technology object: <Axis name>.StatusBits.Homing)

Table 10- 44 Error bits

Error	Description
Min software limit reached	The lower software limit switch has been reached. (Tag of technology object: <Axis name>.ErrorBits.SwLimitMinReached)
Min software limit exceeded	The lower software limit switch has been exceeded. (Tag of technology object: <Axis name>.ErrorBits.SwLimitMinExceeded)
Max software limit reached	The upper software limit switch has been reached. (Tag of technology object: <Axis name>.ErrorBits.SwLimitMaxReached)
Max software limit exceeded	The upper software limit switch has been exceeded. (Tag of technology object: <Axis name>.ErrorBits.SwLimitMaxExceeded)
Negative hardware limit	The lower hardware limit switch has been approached. (Tag of technology object: <Axis name>.ErrorBits.HwLimitMin)
Positive hardware limit	The upper hardware limit switch has been approached. (Tag of technology object: <Axis name>.ErrorBits.HwLimitMax)
PTO already used	A second axis is using the same PTO and is enabled with "MC_Power". (Tag of technology object: <Axis name>.ErrorBits.HwUsed)

Error	Description
Configuration error	The "Axis" technology object was incorrectly configured or editable configuration data were modified incorrectly during runtime of the user program. (Tag of technology object: <Axis name>.ErrorBits.ConfigFault)
General Error	An internal error has occurred. (Tag of technology object: <Axis name>.ErrorBits.SystemFault)

"Motion status" diagnostic function

Use the "Motion status" diagnostic function to monitor the motion status of the axis. The diagnostic function display is available in online mode in "Manual control" mode and in "Automatic control" when the axis is active.

Table 10- 45 Motion status

Status	Description
Target position	The "Target position" field indicates the current target position of an active positioning task of motion control instruction "MC_MoveAbsolute" or "MC_MoveRelative" or of the control panel. The value of the "Target position" is only valid during execution of a positioning task. (Tag of technology object: <Axis name>.MotionStatus.TargetPosition)
Current position	The "Current position" field indicates the current axis position. If the axis is not homed, the value indicates the position value relative to the enable position of the axis. (Tag of technology object: <Axis name>.MotionStatus.Position)
Current velocity	The "Current velocity" field indicates the actual axis velocity. (Tag of technology object: <Axis name>.MotionStatus.Velocity)

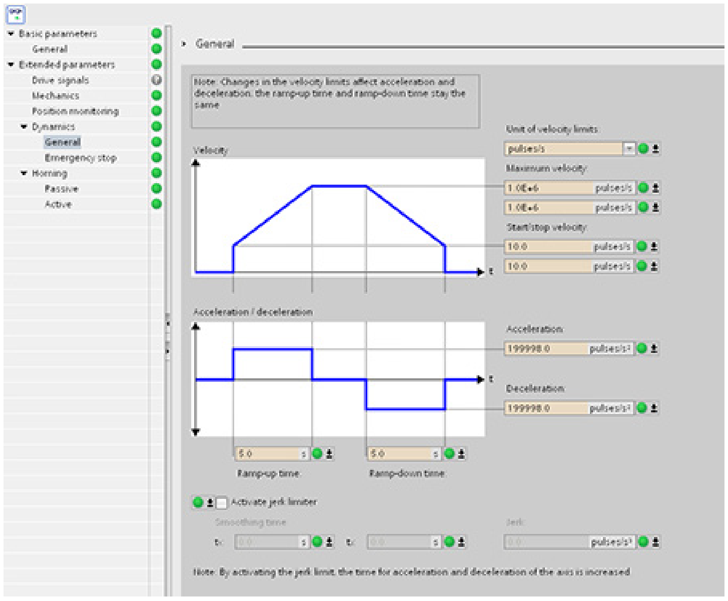
Table 10- 46 Dynamic limits

Dynamic limit	Description
Velocity	The "Velocity" field indicates the configured maximum velocity of the axis. (Tag of technology object: <Axis name>.Config.DynamicLimits.MaxVelocity)
Acceleration	The "Acceleration" field indicates the currently configured acceleration of the axis. (Tag of technology object: <Axis name>.Config.DynamicDefaults.Acceleration)
Deceleration	The "Deceleration" field indicates the currently configured deceleration of the axis. (Tag of technology object: <Axis name>.Config.DynamicDefaults.Deceleration)

Motion start value control

You can edit the actual values of the Motion configuration parameters so that the behavior of the process can be optimized in online mode.

Open the "Technology objects" for your motion control and its "Configuration" object. To access the start value control, click the "eyeglasses icon" in the upper left corner of the dialog:



Note: Changes in the velocity limits affect acceleration and deceleration. the ramp-up time and ramp-down time stay the same

Unit of velocity limits: pulses/s

Maximum velocity: 1.0E+6 pulses/s; 1.0E+6 pulses/s

Start/stop velocity: 10.0 pulses/s; 10.0 pulses/s

Acceleration: 199998.0 pulses/s²

Deceleration: 199998.0 pulses/s²

Ramp-up time: 5.0 s

Ramp-down time: 5.0 s

☐ Activate jerk limiter

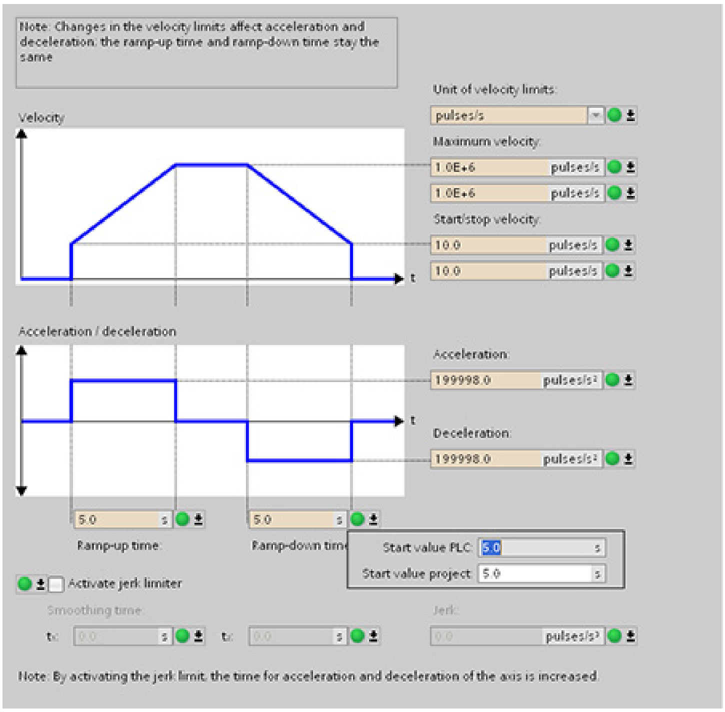
Smoothing time: t₁: 0.0 s; t₂: 0.0 s

Jerk: 0.0 pulses/s³

Note: By activating the jerk limit, the time for acceleration and deceleration of the axis is increased

You can now change the value of any of your motion control configuration parameters as shown in the figure below.

You can compare the actual value to the project (offline) start value and the PLC (online) start value of each parameter. This is necessary to compare online/offline differences of the Technology object data block (TO-DB) and to be informed about the values that will be used as current values on the next Stop-to-Start transition of the PLC. In addition, a compare icon gives a visual indication to help easily identify online/offline differences.



Note: Changes in the velocity limits affect acceleration and deceleration: the ramp-up time and ramp-down time stay the same

Unit of velocity limits: pulses/s

Maximum velocity: 1.0E+6 pulses/s

1.0E+6 pulses/s

Start/stop velocity: 10.0 pulses/s

10.0 pulses/s

Acceleration: 199998.0 pulses/s²

Deceleration: 199998.0 pulses/s²

Ramp-up time: 5.0 s

Ramp-down time: 5.0 s

Start value PLC: 5.0 s

Start value project: 5.0 s

Activate jerk limiter

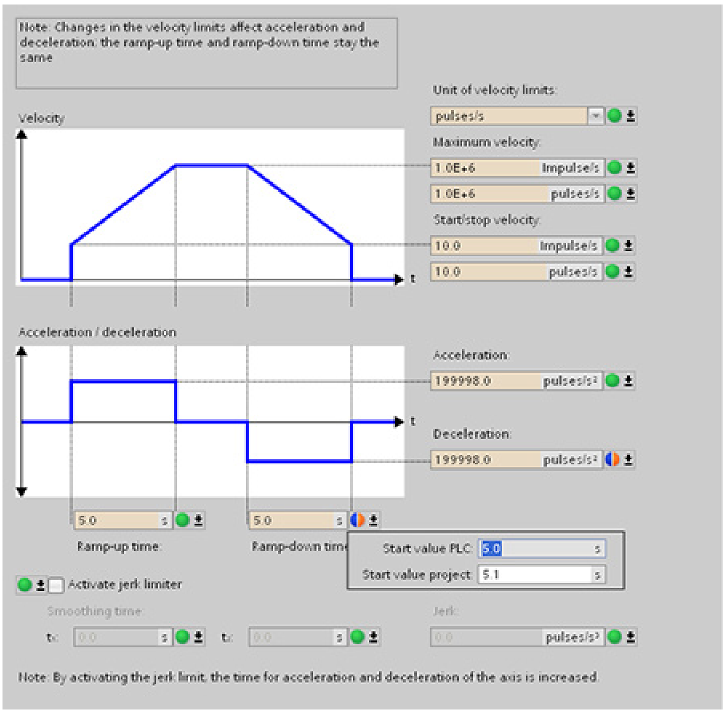
Smoothing time: t1: 0.0 s t2: 0.0 s

Jerk: 0.0 pulses/s³

Note: By activating the jerk limit, the time for acceleration and deceleration of the axis is increased.

The figure above shows the Motion parameter screen with compare icons showing which values are different between online and offline projects. A green icon indicates that the values are the same; a blue/orange icon indicates that the values are different.

Additionally, click the parameter button with the downward arrow to open a small window that shows the project (offline) start value and the PLC (online) start value of each parameter.



Note: Changes in the velocity limits affect acceleration and deceleration: the ramp-up time and ramp-down time stay the same

Unit of velocity limits: pulses/s

Maximum velocity: 1.0E+6 Impulse/s

1.0E+6 pulses/s

Start/stop velocity: 10.0 Impulse/s

10.0 pulses/s

Acceleration: 199998.0 pulses/s²

Deceleration: 199998.0 pulses/s²

Ramp-up time: 5.0 s

Ramp-down time: 5.0 s

Start value PLC: 5.0 s

Start value project: 5.1 s

Activate jerk limiter

Smoothing time: t1: 0.0 s t2: 0.0 s

Jerk: 0.0 pulses/s³

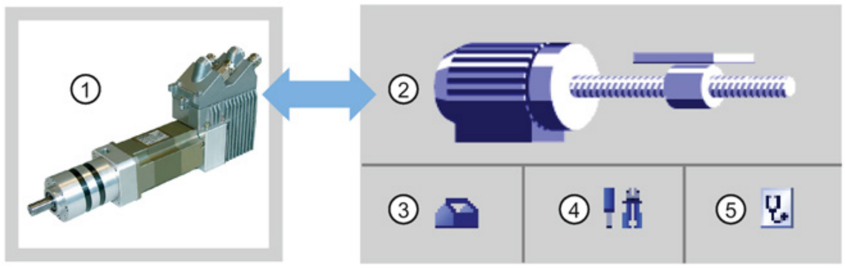
Note: By activating the jerk limit, the time for acceleration and deceleration of the axis is increased.

10.3.4 Closed loop motion control

10.3.4.1 Configuring the axis

You connect the closed loop axis on the PLC and the drive through the analog drive or PROFIdrive. The closed loop axis requires an encoder as well.

STEP 7 provides the configuration tools, the commissioning tools, and the diagnostic tools for the "Axis" technology object.



①	Drive	④	Commissioning
②	Technology object	⑤	Diagnostics
③	Configuration		

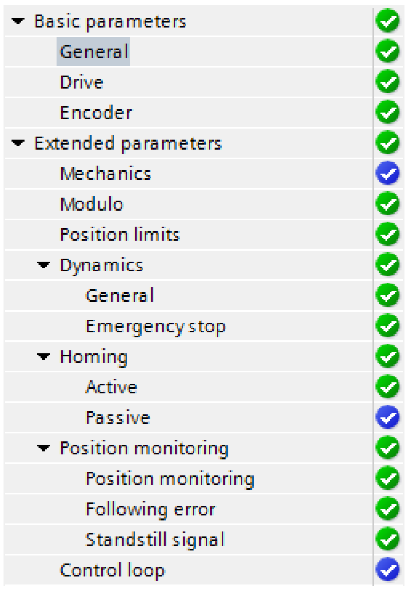
Table 10- 47 STEP 7 tools for closed loop motion control

Tool	Description
Configuration	Configures the following properties of the "Axis" technology object: • Selection of the analog drive connection or PROFIdrive to be used and configuration of the drive and encoder interface • Properties of the mechanics and the transmission ratio of the drive and encoder (or machine or system) • Properties for position limits, dynamics, and homing Save the configuration in the data block of the technology object.
Commissioning	Tests the function of your axis without having to create a user program. When the tool is started, the control panel will be displayed. The following commands are available on the control panel: • Enable and disable axis • Move axis in jog mode • Position axis in absolute and relative terms • Home axis • Acknowledge errors The velocity and the acceleration / deceleration can be specified for the motion commands. The control panel also shows the current axis status.
Diagnostics	Monitors of the current status and error information for the axis and drive.

Note

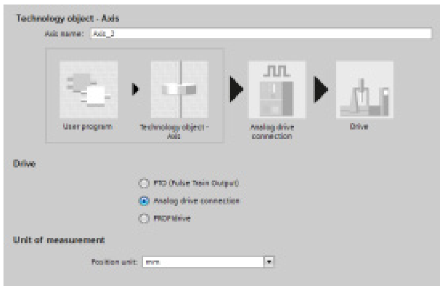
You may have to adapt the values of the input parameters of motion control instructions to the new dimension unit in the user program.

After you create the technology object for the axis, you configure the axis by defining the basic parameters, either the Analog drive or the PROFIdrive connection and the configuration of the drive and encoder.



The tree selector for the analog drive or PROFIdrive connection includes the Encoder, Modulo, Position monitoring, and Control loop configuration menus.

Analog drive connection configuration



Technology object - Axis

Axis name: Axis_3

Drive

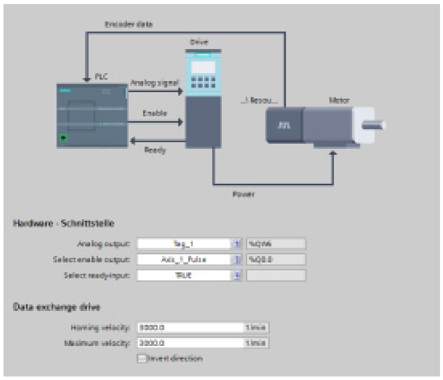
- PTO (Pulse Train Output)
- [✓ Analog drive connection]
- PROFIdrive

Unit of measurement

Position unit: mm

In the General configuration dialog, you select the following parameters:

- "Analog drive connection" radio button
- Unit of measurement



Hardware - Schnittstelle

Analog output:	Tag_1	%QW6
Select enable output:	Axis_1_Pulse	%Q0.0
Select ready-input:	TRUE	

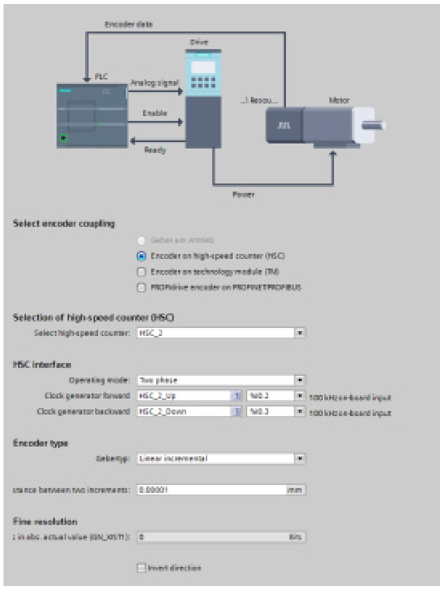
Data exchange drive

Homing velocity:	3000.0	1/min
Maximum velocity:	3000.0	1/min

☐ Invert direction

In the Drive configuration dialog, you select the following parameters:

- Analog drive hardware outputs
- Data exchange drive velocities



Select encoder coupling

- Geber am Antrieb
- [✓ Encoder on high-speed counter (HSC)]
- Encoder on technology module (TM)
- PROFIdrive encoder on PROFINET/PROFIBUS

Selection of high-speed counter (HSC)

Select high-speed counter: HSC_2

HSC interface

Operating mode:	Two phase		
Clock generator forward	HSC_2_Up	%I0.2	100 kHz on-board input
Clock generator backward	HSC_2_Down	%I0.3	100 kHz on-board input

Encoder type

Gebertyp: Linear incremental

stance between two increments: 0.00001 mm

Fine resolution

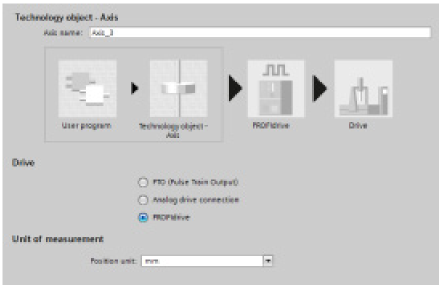
: in abs. actual value (GN_XIST1): 0 Bits

☐ Invert direction

In the Encoder configuration dialog, you select the following parameters:

- Analog drive encoder coupling (for example, a high-speed counter (HSC))
- HSC interface
- Encoder type
- Fine resolution

PROFIdrive configuration



Technology object - Axis

Axis name: Axis_3

Drive

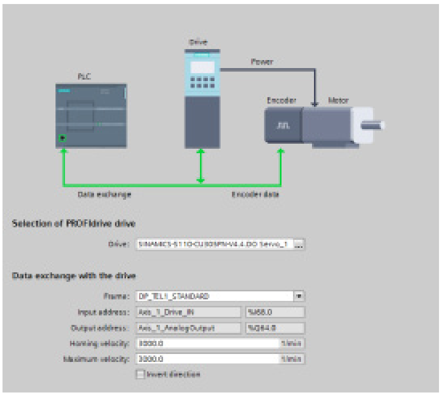
( ) PTO (Pulse Train Output)
( ) Analog drive connection
(●) PROFIdrive

Unit of measurement

Position unit: mm

In the General configuration dialog, you select the following parameters:

- "PROFIdrive" radio button
- Unit of measurement



Selection of PROFIdrive drive

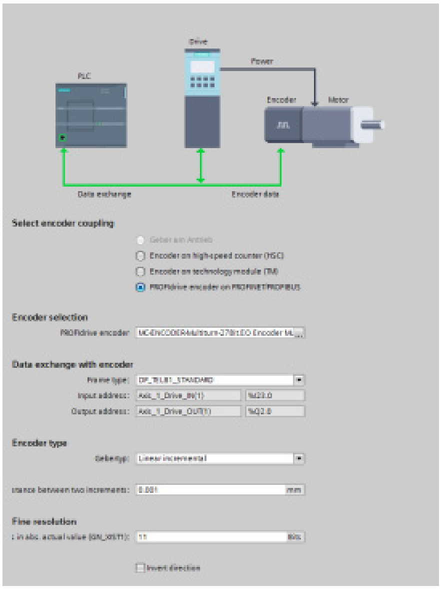
Drive: SINAMICS-S110-CU305PN-V4.4.DO Servo_1

Data exchange with the drive

Frame: DP_TEL1_STANDARD
Input address: Axis_1_Drive_IN %I68.0
Output address: Axis_1_AnalogOutput %Q64.0
Homing velocity: 3000.0 1/min
Maximum velocity: 3000.0 1/min
[ ] Invert direction

In the Drive configuration dialog, you select the following parameters:

- PROFIdrive drive
- Data exchange with the drive



Select encoder coupling

( ) Geber am Antrieb
( ) Encoder on high-speed counter (HSC)
( ) Encoder on technology module (TM)
(●) PROFIdrive encoder on PROFINET/PROFIBUS

Encoder selection

PROFIdrive encoder: MC-ENCODERMultiturn-27Bit.DO Encoder M...

Data exchange with encoder

Frame type: DP_TEL81_STANDARD
Input address: Axis_1_Drive_IN(1) %I23.0
Output address: Axis_1_Drive_OUT(1) %Q2.0

Encoder type

Gebertyp: Linear incremental

stance between two increments: 0.001 mm

Fine resolution

: in abs. actual value (G1_XIST1): 11 Bits

[ ] Invert direction

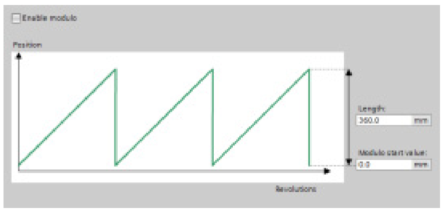
In the Encoder configuration dialog, you select the following parameters:

- PROFIdrive encoder coupling (for example, a PROFIdrive encoder on PROFINET)
- PROFIdrive encoder
- Data exchange with the encoder
- Encoder type
- Fine resolution

Extended parameters

You can also configure the following properties of the closed loop axis:

- Modulo
- Position limits
- Dynamics
- Homing
- Position monitoring
- Following error
- Standstill signal
- Control loop

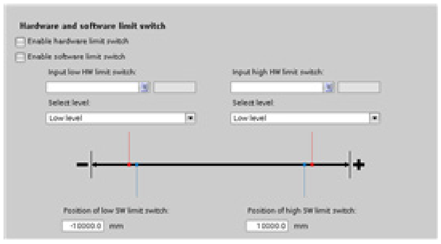


☐ Enable modulo

Length: 360.0 mm

Modulo start value: 0.0 mm

Modulo: You can configure a "Modulo" axis to move the load in a cyclic area which has a start value/start position and a given length. If the position of the load reaches the end of this area, it is automatically set to the start value again. You enable the "Length" and "Modulo start value" fields when you check the "Enable Modulo" check box.



Hardware and software limit switch

☐ Enable hardware limit switch

☐ Enable software limit switch

Input low HW limit switch:

Select level: Low level

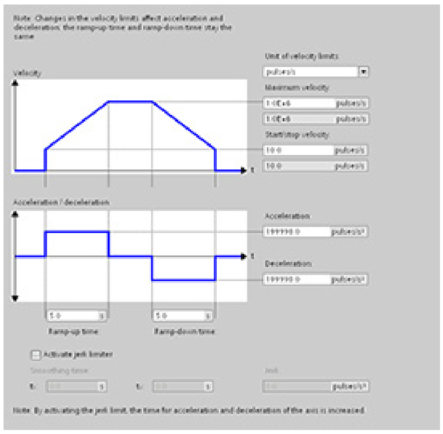
Input high HW limit switch:

Select level: Low level

Position of low SW limit switch: -10000.0 mm

Position of high SW limit switch: 10000.0 mm

Position limits: You can configure the properties for the drive signals, drive mechanics, and position monitoring (hardware and software limit switches).



Note: Changes in the velocity limits affect acceleration and deceleration; the ramp-up time and ramp-down time stay the same

Unit of velocity limits: pulses/s

Maximum velocity: 1.0E+6 pulses/s; 1.0E+6 pulses/s

Start/stop velocity: 10.0 pulses/s; 10.0 pulses/s

Acceleration: 199998.0 pulses/s²

Deceleration: 199998.0 pulses/s²

Ramp-up time: 5.0 s

Ramp-down time: 5.0 s

☐ Activate jerk limiter

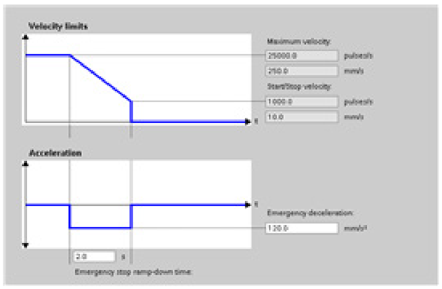
Smoothing time

t₁: 0.0 s t₂: 0.0 s

Jerk: 0.0 pulses/s³

Note: By activating the jerk limit, the time for acceleration and deceleration of the axis is increased.

Dynamics: You can configure the motion dynamics and the behavior of the emergency stop command.



Velocity limits

Maximum velocity: 25000.0 pulses/s; 250.0 mm/s

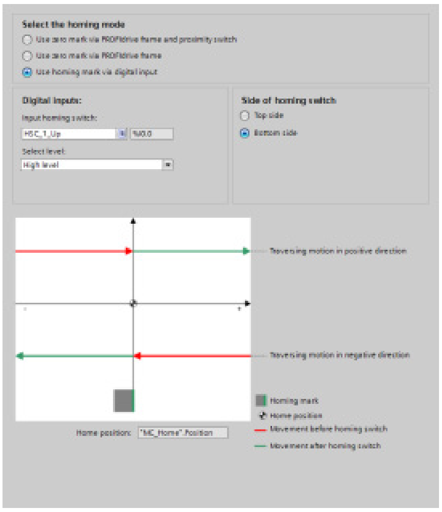
Start/Stop velocity: 1000.0 pulses/s; 10.0 mm/s

Acceleration

Emergency deceleration: 120.0 mm/s²

Emergency stop ramp-down time: 2.0 s

Homing: You can configure the homing behavior (passive and active).



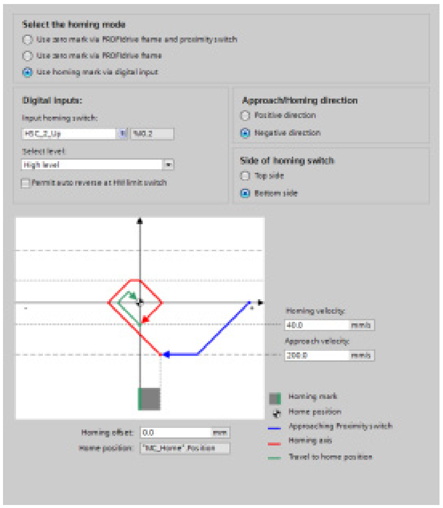
Select the homing mode
( ) Use zero mark via PROFIdrive frame and proximity switch
( ) Use zero mark via PROFIdrive frame
(✓) Use homing mark via digital input

Digital inputs:
Input homing switch: HSC_1_Up %I0.0
Select level: High level

Side of homing switch
( ) Top side
(✓) Bottom side

Home position: "MC_Home".Position

Traversing motion in positive direction
Traversing motion in negative direction
Homing mark
Home position
Movement before homing switch
Movement after homing switch



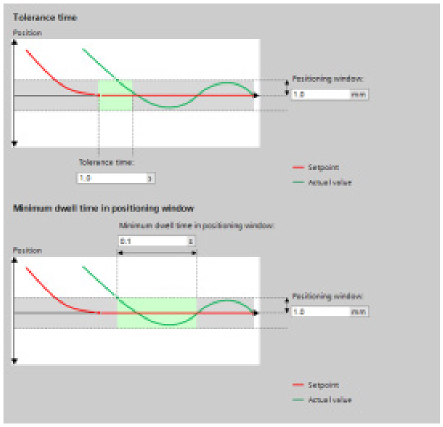
Select the homing mode
( ) Use zero mark via PROFIdrive frame and proximity switch
( ) Use zero mark via PROFIdrive frame
(✓) Use homing mark via digital input

Digital inputs:
Input homing switch: HSC_2_Up %I0.2
Select level: High level
[ ] Permit auto reverse at HW limit switch

Approach/Homing direction
( ) Positive direction
(✓) Negative direction

Side of homing switch
( ) Top side
(✓) Bottom side

Homing velocity: 40.0 mm/s
Approach velocity: 200.0 mm/s
Homing offset: 0.0 mm
Home position: "MC_Home".Position



Tolerance time

Positioning window: 1.0 mm
Tolerance time: 1.0 s

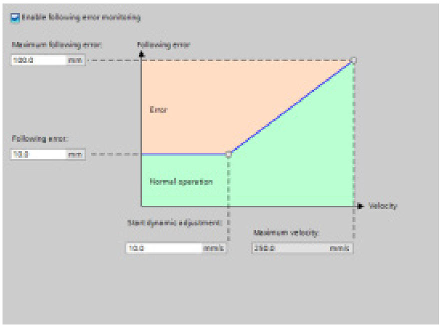
Minimum dwell time in positioning window

Minimum dwell time in positioning window: 0.1 s
Positioning window: 1.0 mm

"Positioning monitoring": You can configure tolerance time as well as minimum dwell time for the positioning window.

The system connects the following three parameters directly with the axis TO-DB:

- Positioning window
- Tolerance time
- Minimum dwell time in positioning window

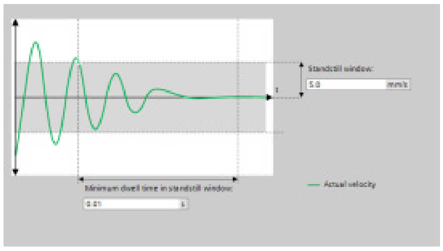


[✓] Enable following error monitoring

Maximum following error: 100.0 mm
Following error: 10.0 mm
Start dynamic adjustment: 10.0 mm/s
Maximum velocity: 250.0 mm/s

"Following error": You can configure the difference of the allowed error distance over a velocity range. You check the "Enable following error monitoring" check box to activate following error. You can configure the following the parameters:

- Maximum following error
- Following error
- Start dynamic adjustment
- Maximum velocity

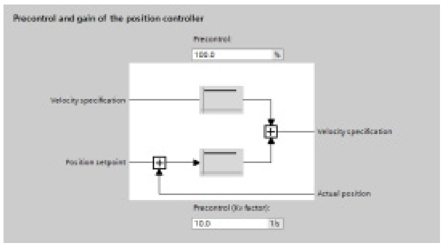


Standstill window: 5.0 mm/s

Minimum dwell time in standstill window: 0.01 s

"Standstill signal": You can configure the following the parameters:

- Minimum dwell time in standstill window
- Standstill window.



Precontrol and gain of the position controller

Precontrol: 100.0 %

Precontrol (Kv factor): 10.0 1/s

"Control loop": You can configure the veloci-ty gain known as "Precontrol (Kv factor)".

Use the "Commissioning" control panel to test the functionality independently from your user program.

Click the "Startup" icon to commission the axis.

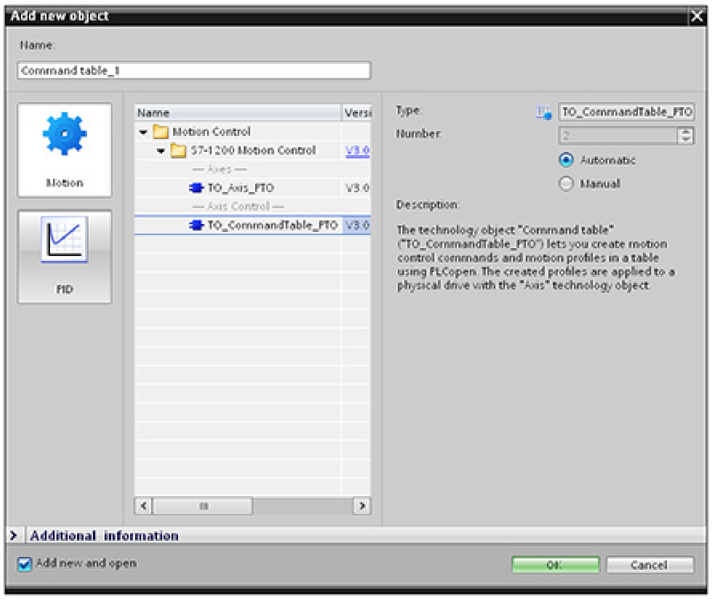
The control panel shows the current status of the axis. Not only can you enable and disable the axis, but you can also test the positioning of the axis (both in absolute and relative terms) and can specify the velocity, acceleration and deceleration. You can also test the homing and jogging tasks. The control panel also allows you to acknowledge errors.

10.3.5 Configuring the TO_CommandTable_PTO

You can configure a MC_CommandTable instruction using the Technology objects. The following example demonstrates how this is done.

Adding a Technology object

1. In the Project tree, expand the node "Technology Objects" and select "Add new object".
2. Select the "CommandTable" icon (rename if required), and click "OK" to open the configuration editor for the CommandTable object.



Name: Command table_1

Name	Versi
Motion Control	
S7-1200 Motion Control	V3.0
— Axes —	
TO_Axis_PTO	V3.0
— Axis Control —	
TO_CommandTable_PTO	V3.0

Type: TO_CommandTable_PTO

Number: 2

[✓ Automatic] Manual

Description:

The technology object "Command table" ("TO_CommandTable_PTO") lets you create motion control commands and motion profiles in a table using PLCopen. The created profiles are applied to a physical drive with the "Axis" technology object.

[✓ Add new and open]

Planning the steps for your application

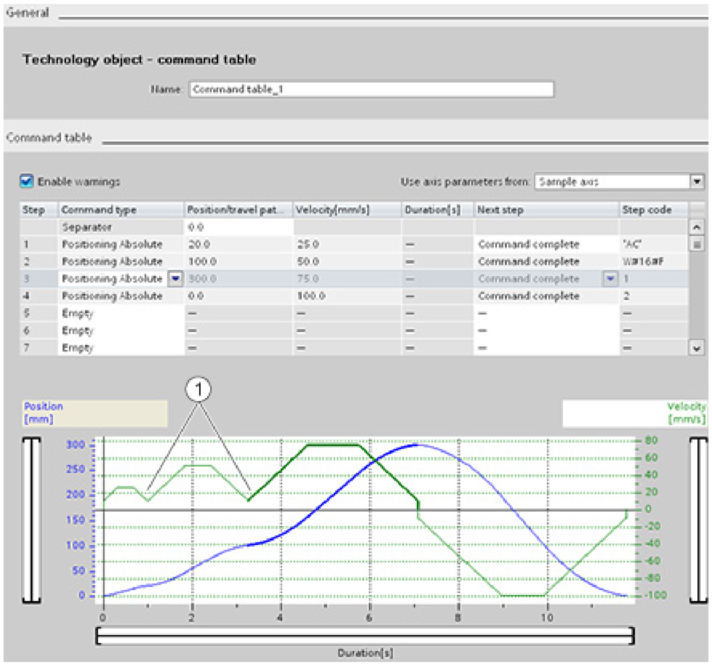
You can create the desired movement sequence in the "Command Table" configuration window, and check the result against the graphic view in the trend diagram.

You can select the command types that are to be used for processing the command table. Up to 32 steps can be entered. The commands are processed in sequence, easily producing a complex motion profile.

Table 10- 48 MC_CommandTable command types

Command type	Description
Empty	The empty serves as a placeholder for any commands to be added. The empty entry is ignored when the command table is processed
Halt	Pause axis. Note: The command only takes place after a "Velocity setpoint" command.
Positioning Relative	Positions the axis based upon distance. The command moves the axis by the given distance and velocity.
Positioning Absolute	Positions the axis based upon location. The command moves the axis to the given location, using the velocity specified.
Velocity setpoint	Moves the axis at the given velocity.
Wait	Waits until the given period is over. "Wait" does not stop an active traversing motion.
Separator	Adds a "Separator" line above the selected line. The separator line allows more than one profile to be defined in a single command table.

In the figure below, "Command complete" is used as the transition to the next step. This type of transition allows your device to decelerate to the start/stop speed and then accelerate once again at the start of the next step.



General

Technology object - command table

Name: Command table_1

Command table

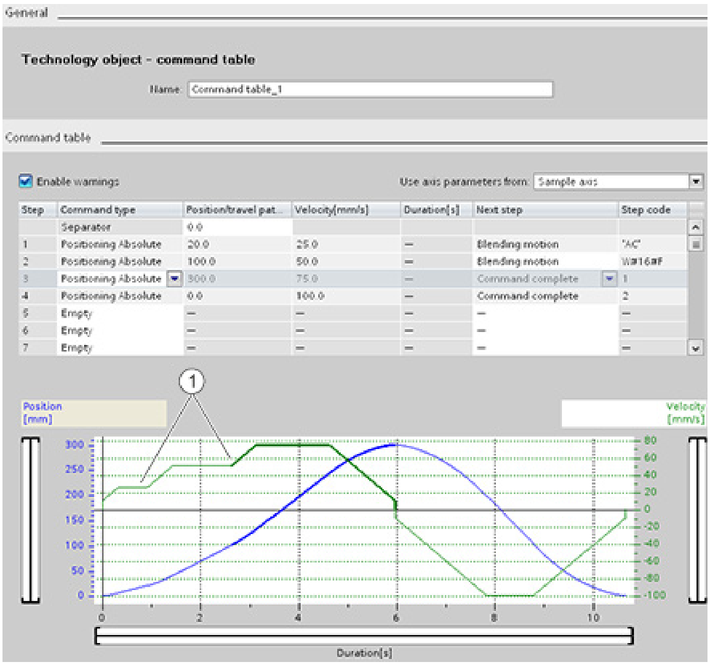
☑ Enable warnings

Use axis parameters from: Sample axis

Step	Command type	Position/travel pat...	Velocity[mm/s]	Duration[s]	Next step	Step code
	Separator	0.0				
1	Positioning Absolute	20.0	25.0	—	Command complete	'AC'
2	Positioning Absolute	100.0	50.0	—	Command complete	W#16#F
3	Positioning Absolute	300.0	75.0	—	Command complete	1
4	Positioning Absolute	0.0	100.0	—	Command complete	2
5	Empty	—	—	—	—	—
6	Empty	—	—	—	—	—
7	Empty	—	—	—	—	—

① Axis decelerates to the start/stop speed between steps.

In the figure below, "Blending motion" is used as the transition to the next step. This type of transition allows your device to maintain its velocity into the start of the next step, resulting in a smooth transition for the device from one step to the next. Using blending can shorten the total time required for a profile to execute completely. Without blending, this example takes seven seconds to run. With blending, the execution time is reduced by one second to a total of six seconds.


General

Technology object - command table

Name: Command table_1

Command table

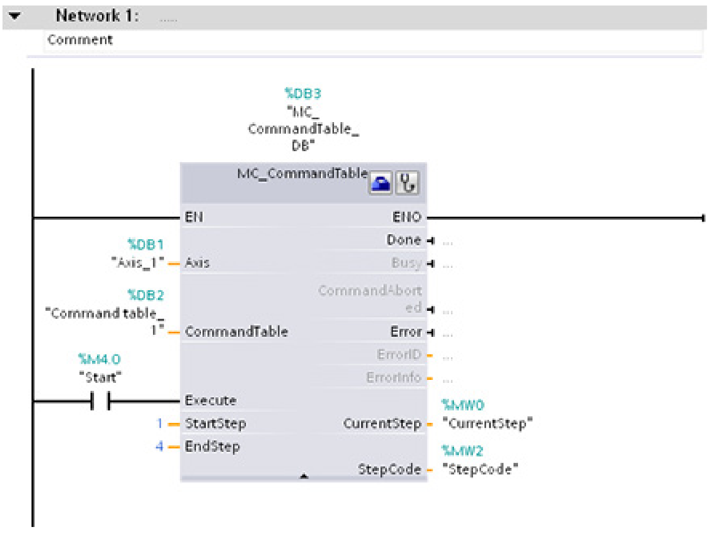
☑ Enable warnings

Use axis parameters from: Sample axis

Step	Command type	Position/travel pat...	Velocity[mm/s]	Duration[s]	Next step	Step code
	Separator	0.0				
1	Positioning Absolute	20.0	25.0	—	Blending motion	'AC'
2	Positioning Absolute	100.0	50.0	—	Blending motion	W#16#F
3	Positioning Absolute	300.0	75.0	—	Command complete	1
4	Positioning Absolute	0.0	100.0	—	Command complete	2
5	Empty	—	—	—	—	—
6	Empty	—	—	—	—	—
7	Empty	—	—	—	—	—

① Axis continues to move and accelerates or decelerates to the next step velocity, saving time and mechanical wear.

The operation of your CommandTable is controlled by an MC_CommandTable instruction, as shown below:



10.3.6 Operation of motion control for S7-1200

10.3.6.1 CPU outputs used for motion control

The CPU provides four pulse output generators. Each pulse output generator provides one pulse output and one direction output for controlling a stepper motor drive or a servo motor drive with pulse interface. The pulse output provides the drive with the pulses required for motor motion. The direction output controls the travel direction of the drive.

The PTO output generates a square wave output of variable frequency. Pulse generation is controlled by configuration and execution information supplied through H/W configuration and/or SFCs/SFBs.

Based upon the user's selection while the CPU is in RUN mode, either the values stored in the image register or the pulse generator outputs drive the digital outputs. In STOP mode, the PTO generator does not control the outputs.

Onboard CPU outputs and outputs of a signal board can be used as pulse and direction outputs. You select between onboard CPU outputs and outputs of the signal board during device configuration under Pulse generators (PTO/PWM) on the "Properties" tab. Only PTO (Pulse Train Output) applies to motion control.

The table below shows the default I/O assignments; however, the four pulse generators can be configured to any digital output.

Note

Pulse-train outputs cannot be used by other instructions in the user program.

When you configure the outputs of the CPU or signal board as pulse generators (for use with the PWM or motion control instructions), the corresponding output addresses no longer control the outputs. If your user program writes a value to an output used as a pulse generator, the CPU does not write that value to the physical output.

Note

PTO direction outputs can be freed for use elsewhere in your program.

Each PTO requires the assignment of two outputs: one as a pulse output and one as a direction output. You can use just the pulse output and not the direction output. You can then free the direction output for other purposes in your user program. The output cannot be used for both the PTO direction output and in the user program, simultaneously.

Table 10- 49 Default address assignments of the pulse and direction outputs

Usage of outputs for motion control		
	Pulse	Direction
PTO1		
Built-in I/O	Q0.0	Q0.1
SB I/O	Q4.0	Q4.1
PTO2		
Built-in I/O	Q0.2	Q0.3
SB I/O	Q4.2 [1]	Q4.3 [1]
PTO3		
Built-in I/O	Q0.4 [2]	Q0.5 [2]
SB I/O	Q4.0	Q4.1
PTO4		
Built-in I/O	Q0.6 [3]	Q0.7 [3]
SB I/O	Q4.2	Q4.3

1 Outputs Q4.2 and Q4.3 are only available on the SB1222 DQ4.

2 The CPU 1211C does not have outputs Q0.4, Q0.5, Q0.6, or Q0.7. Therefore, these outputs cannot be used in the CPU 1211C.

3 The CPU 1212C does not have outputs Q0.6 or Q0.7. Therefore, these outputs cannot be used in the CPU 1212C.

4 This table applies to the CPU 1211C, CPU 1212C, CPU 1214C, CPU 1215C, and CPU 1217C PTO functions.

Drive interface

For motion control, you can optionally configure a drive interface for "Drive enabled" and "Drive ready". When using the drive interface, the digital output for the drive enable and the digital input for "drive ready" can be freely selected.

Note

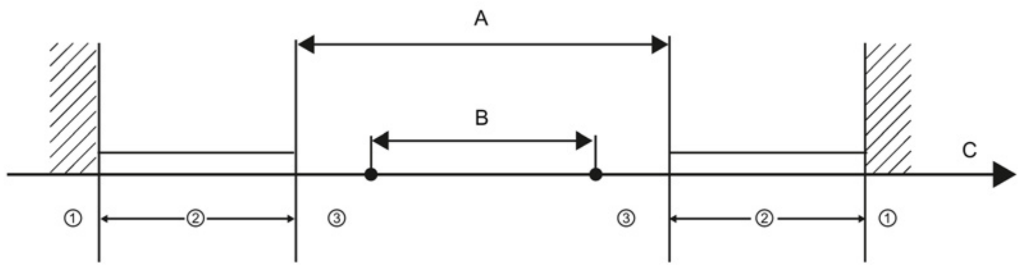
The firmware will take control through the corresponding pulse and direction outputs if the PTO (Pulse Train Output) has been selected and assigned to an axis.

With this takeover of the control function, the connection between the process image and I/O output is also disconnected. While the user has the possibility of writing the process image of pulse and direction outputs via the user program or watch table, this is never transferred to the I/O output. Accordingly, it is also not possible to monitor the I/O output via the user program or watch table. The information read merely reflects the value of the process image and does not match the actual status of the I/O output in any respect.

For all other CPU outputs that are not used permanently by the CPU firmware, the status of the I/O output can be controlled or monitored via the process image, as usual.

10.3.6.2 Hardware and software limit switches for motion control

Use the hardware and software limit switches to limit the "allowed travel range" and the "working range" of your axis.



①	Mechanical stop	A	Allowed travel range for the axis
②	Lower and upper hardware limits	B	Working range of the axis
③	Lower and upper software limits	C	Distance

Hardware and software limit switches must be activated prior to use in the configuration or in the user program. Software limit switches are only active after homing the axis.

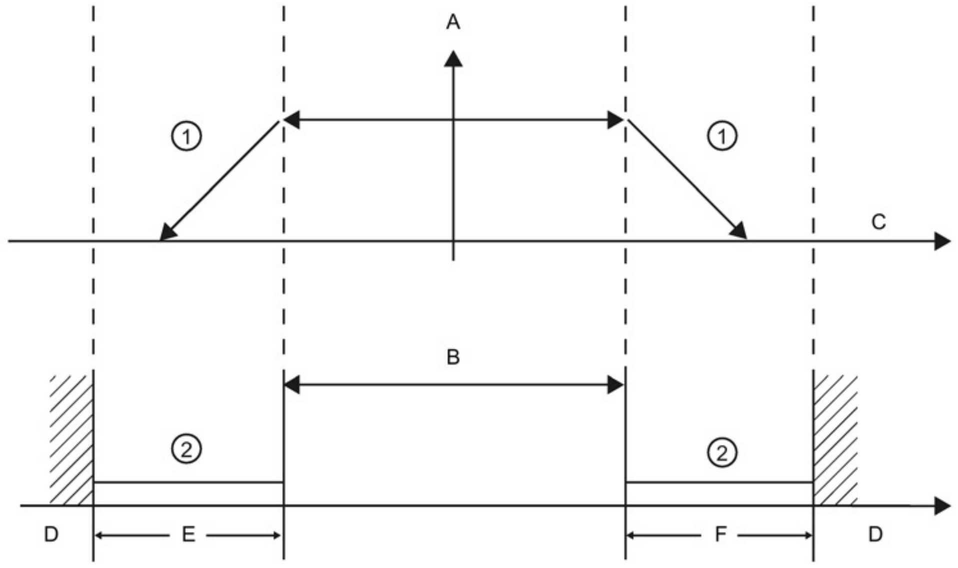
Hardware limit switches

Hardware limit switches determine the maximum travel range of the axis. Hardware limit switches are physical switching elements that must be connected to interrupt-capable inputs of the CPU. Use only hardware limit switches that remain permanently switched after being approached. This switching status may only be revoked after a return to the allowed travel range.

Table 10- 50 Available inputs for hardware limits

Description	RPS	LIM-	LIM+
Built-in I/O	I0.0 - I1.5		
SB I/O	I4.0 - I4.3		

When the hardware limit switches are approached, the axis brakes to a standstill at the configured emergency deceleration. The specified emergency deceleration must be sufficient to reliably stop the axis before the mechanical stop. The following diagram presents the behavior of the axis after it approaches the hardware limit switches.



①	The axis brakes to a standstill at the configured emergency deceleration.
②	Range in which the hardware limit switches signal the stats "approached".
A	[Velocity]
B	Allowed travel range
C	Distance
D	Mechanical stop
E	Lower hardware limit switch
F	Upper hardware limit switch

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 20.0 ms accumulated duration before the filter becomes fully responsive to new inputs. During this time, short "0" pulse events of duration less than 20.0 ms may not be detected or counted.

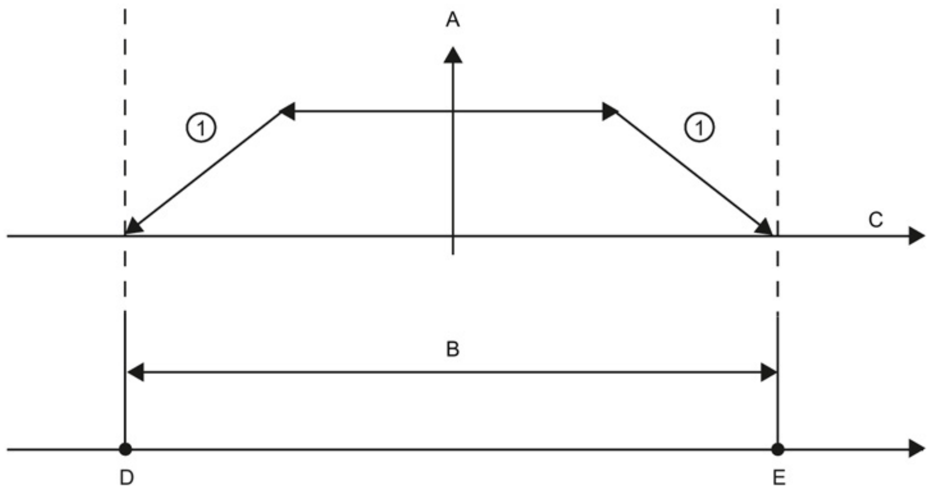
This changing of filter times can result in unexpected machine or process operation, which may cause death or serious injury to personnel, and/or damage to equipment.

To ensure that a new filter time goes immediately into effect, a power cycle of the CPU must be applied.

Software limit switches

Software limit switches limit the "working range" of the axis. They should fall inside the hardware limit switches relative to the travel range. Because the positions of the software limit switches can be set flexibly, the working range of the axis can be restricted on an individual basis depending on the current traversing profile. In contrast to hardware limit switches, software limit switches are implemented exclusively by means of the software and do not require their own switching elements.

If software limit switches are activated, an active motion is stopped at the position of the software limit switch. The axis is braked at the configured deceleration. The following diagram presents the behavior of the axis until it reaches the software limit switches.



①	The axis brakes to a standstill at the configured deceleration.
A	[Velocity]
B	Working range
C	Distance
D	Lower software limit switch
E	Upper software limit switch

Use additional hardware limit switches if a mechanical endstop is located after the software limit switches and there is a risk of mechanical damage.

Additional information

Your user program can override the hardware or software position limits by enabling or disabling both hardware and software limits functionality. The selection is made from the Axis DB.

- To enable or disable the hardware limit functionality, access the "Active" tag (Bool) in the DB path "<axis name>/Config/**PositonLimits_HW**". The state of the "Active" tag enables or disables the use of hardware position limits.
- To enable or disable software position limit functionality, access "Active" tag (Bool) in the DB path "<axis name>/Config/**Position Limits_SW**". The state of this "Active" tag enables or disables the software position limits.

You can also modify the software position limits with your user program (for example, to add flexibility for machine setup or to shorten machine change-over time). Your user program can write new values to the " MinPosition " and " MaxPosition " tags (engineering units in Real format) in the DB "<axis name>/Config/**PositionLimits_SW**".

10.3.6.3 Homing

Homing refers to the matching of the axis coordinates to the real, physical drive position. (If the drive is currently at position x, the axis will be adjusted to be in position x.) For position-controlled axes, the entries and displays for the position refer exactly to these axis coordinates.

Note

The agreement between the axis coordinates and the real situation is extremely important. This step is necessary to ensure that the absolute target position of the axis is also achieved exactly with the drive.

The MC_Home instruction initiates the homing of the axis.

There are 4 different homing functions. The first two functions allow the user to set the current position of the axis and the second two position the axis with respect to a Home reference Sensor.

- Mode 0 - Direct Referencing Absolute: When executed this mode tells the axis exactly where it is. It sets the internal position variable to the value of the Position input of the Homing instruction. This is used for machine calibration and setup.

  The axis position is set regardless of the reference point switch. Active traversing motions are not aborted. The value of the Position input parameter of the MC_Home instruction is set immediately as the reference point of the axis. To assign the reference point to an exact mechanical position, the axis must be at a standstill at this position at the time of the homing operation.
- Mode 1 - Direct Referencing Relative: When executed this mode uses the internal position variable and adds the value of the Position input on the Homing instruction to it. This is typically used to account for machine offset.

  The axis position is set regardless of the reference point switch. Active traversing motions are not aborted. The following statement applies to the axis position after homing: New axis position = current axis position + value of the Position parameter of the MC_Home instruction.

- Mode 2 - Passive Referencing: When the axis is moving and passes the Reference Point Switch the current position is set as the home position. This feature will help account for normal machine wear and gear backlash and prevent the need for manual compensation for wear. The Position input on the Homing instruction, as before, adds to the location indicated by the Reference Point Switch allowing easy offset of the Home position.

  During passive homing, the MC_Home instruction does not carry out any homing motion. The traversing motion required for this step must be implemented by the user via other motion control instructions. When the reference point switch is detected, the axis is homed according to the configuration. Active traversing motions are not aborted upon start of passive homing.
- Mode 3 - Active Referencing: This mode is the most precise method of Homing the Axis. The initial direction and velocity of movement is configured in the Technology Object Configuration Extended Parameters-Homing. This is dependent upon machine configuration. There is also the ability to determine if the leading edge or falling edge of the Reference Point Switch signal is the Home position. Virtually all sensors have an active range and if the Steady State On position was used as the Home signal then there would be a possibility for error in the Homing position since the On signal active range would cover a range of distance. By using either the leading or falling edge of that signal a much more precise Home position results. As with all other modes the value of the Position input on the Homing instruction is added to the Hardware referenced position.

  In active homing mode, the MC_Home instruction performs the required reference point approach. When the reference point switch is detected, the axis is homed according to the configuration. Active traversing motions are aborted.

Modes 0 and 1 do not require that the axis be moved at all. They are typically used in setup and calibration. Modes 2 and 3 require that the axis move and pass a sensor that is configured in the "Axis" technology object as the Reference Point Switch. The reference point can be placed in the work area of the axis or outside of the normal work area but within movement range.

Configuration of homing parameters

Configure the parameters for active and passive homing in the "Homing" configuration window. The homing method is set using the "Mode" input parameter of the motion control instruction. Here, Mode = 2 means passive homing and Mode = 3 means active homing.

Note

Use one of the following measures to ensure that the machine does not travel to a mechanical endstop in the event of a direction reversal:

- Keep the approach velocity low
- Increase the configured acceleration/deceleration
- Increase the distance between hardware limit switch and mechanical stop

Table 10- 51 Configuration parameters for homing the axis

Parameter	Description
Input reference point switch (Active and passive homing)	Select the digital input for the reference point switch from the drop-down list box. The input must be interrupt-capable. The onboard CPU inputs and inputs of an inserted signal board can be selected as inputs for the reference point switch. The default filter time for the digital inputs is 6.4 ms. When the digital inputs are used as a reference point switch, this can result in undesired decelerations and thus inaccuracies. Depending on the reduced velocity and extent of the reference point switch, the reference point may not be detected. The filter time can be set under "Input filter" in the device configuration of the digital inputs. The specified filter time must be less than the duration of the input signal at the reference point switch.
Auto reverse after reaching the hardware limit switches (Active homing only)	Activate the check box to use the hardware limit switch as a reversing cam for the reference point approach. The hardware limit switches must be configured and activated for direction reversal. If the hardware limit switch is reached during active homing, the axis brakes at the configured deceleration (not with the emergency deceleration) and reverses direction. The reference point switch is then sensed in reverse direction. If the direction reversal is not active and the axis reaches the hardware limit switch during active homing, the reference point approach is aborted with an error and the axis is braked at the emergency deceleration.
Approach direction (Active and passive homing)	With the direction selection, you determine the "approach direction" used during active homing to search for the reference point switch, as well as the homing direction. The homing direction specifies the travel direction the axis uses to approach the configured side of the reference point switch to carry out the homing operation.
Reference point switch (Active and passive homing)	• Active homing: Select whether the axis is to be referenced on the left or right side of the reference point switch. Depending on the start position of the axis and the configuration of the homing parameters, the reference point approach sequence can differ from the diagram in the configuration window. • Passive homing: With passive homing, the traversing motions for purposes of homing must be implemented by the user via motion commands. The side of the reference point switch on which homing occurs depends on the following factors: – "Approach direction" configuration – "Reference point switch" configuration – Current travel direction during passive homing
Approach velocity (Active homing only)	Specify the velocity at which the reference point switch is to be searched for during the reference point approach. Limit values (independent of the selected user unit): Start/stop velocity ≤ approach velocity ≤ maximum velocity

Parameter	Description
Reduced velocity (Active homing only)	Specify the velocity at which the axis approaches the reference point switch for homing. Limit values (independent of the selected user unit): Start/stop velocity ≤ reduced velocity ≤ maximum velocity
Home position offset (Active homing only)	If the desired reference position deviates from the position of the reference point 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 reference point switch: 1. Move the axis at reduced velocity by the value of the home position offset. 2. When the position of the home position offset is reached, the axis position is set to the absolute reference position. The absolute reference position is specified via parameter "Position" of motion control instruction "MC_Home". Limit values (independent of the selected user unit): -1.0e12 ≤ home position offset ≤ 1.0e12

Table 10- 52 Factors that affect homing

Influencing factors:			Result:
Configuration Approach direction	Configuration Reference point switch	Current travel direction	Homing on Reference point switch
Positive	"Left (negative) side"	Positive direction	Left
		Negative direction	Right
Positive	"Right (positive) side"	Positive direction	Right
		Negative direction	Left
Negative	"Left (negative) side"	Positive direction	Right
		Negative direction	Left
Negative	"Right (positive) side"	Positive direction	Left
		Negative direction	Right

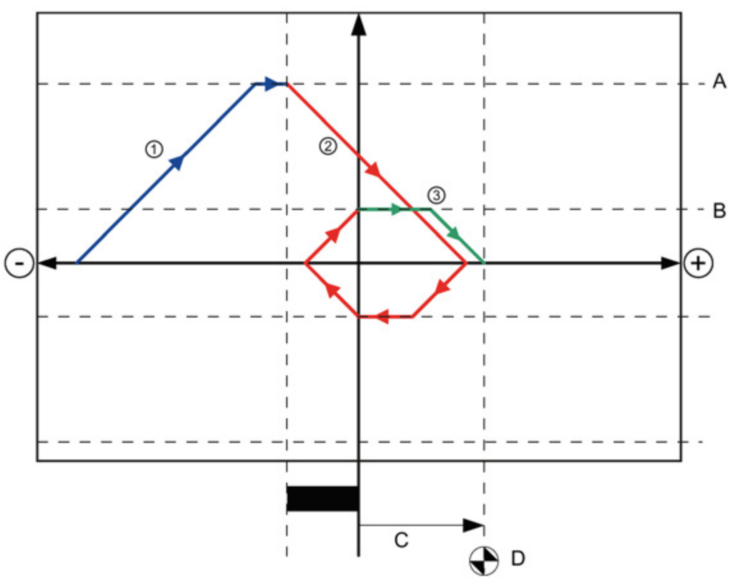
Sequence for active homing

You start active homing with motion control instruction "MC_Home" (input parameter Mode = 3). Input parameter "Position" specifies the absolute reference point coordinates in this case. Alternatively, you can start active homing on the control panel for test purposes.

The following diagram shows an example of a characteristic curve for an active reference point approach with the following configuration parameters:

- "Approach direction" = "Positive approach direction"
- "Reference point switch" = "Right (positive) side"
- Value of "home position offset" > 0

Table 10- 53 Velocity characteristics of MC homing

Operation	Notes	
	A	Approach velocity
	B	Reduced velocity
	C	Home position coordinate
	D	Home position offset

①	Search phase (blue curve segment): When active homing starts, the axis accelerates to the configured "approach velocity" and searches at this velocity for the reference point switch.
②	Reference point approach (red curve section): When the reference point switch is detected, the axis in this example brakes and reverses, to be homed on the configured side of the reference point switch at the configured "reduced velocity".
③	Travel to reference point position (green curve segment): After homing at the reference point switch, the axis travels to the "Reference point coordinates" at the "reduced velocity". On reaching the "Reference point coordinates", the axis is stopped at the position value that was specified in the Position input parameter of the MC_Home instruction".

Note

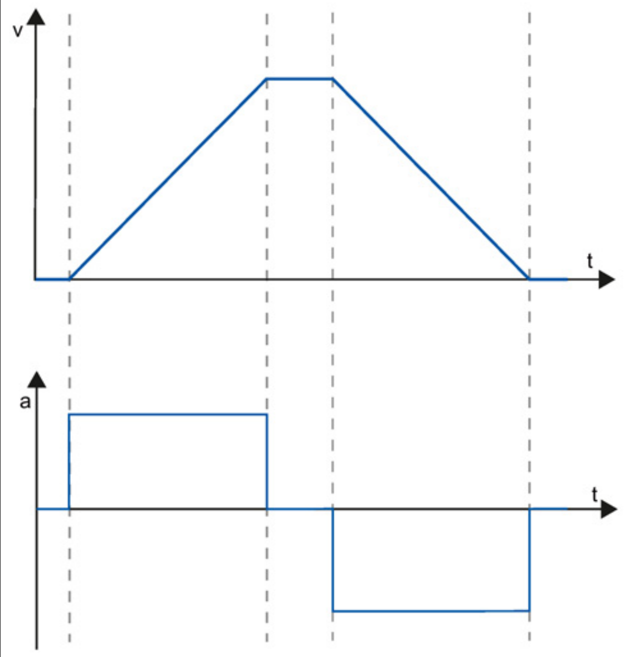
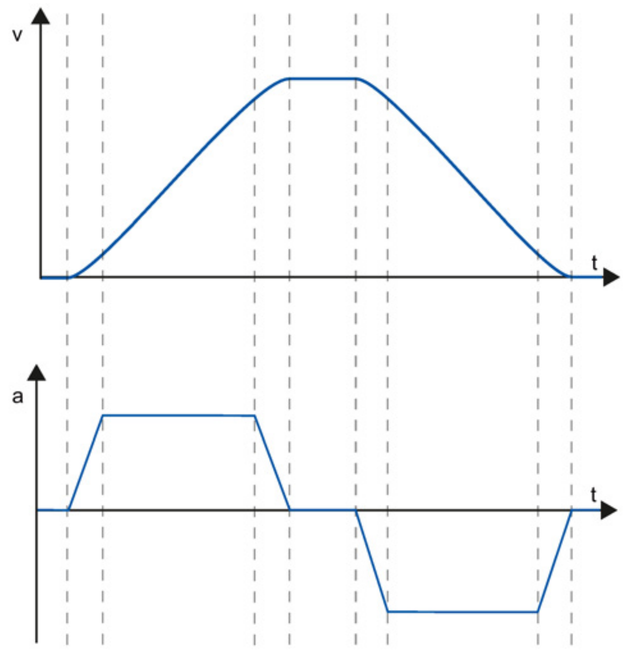
If the homing search does not function as you expected, check the inputs assigned to the hardware limits or to the reference point. These inputs may have had their edge interrupts disabled in device configuration.

Examine the configuration data for the axis technology object of concern to see which inputs (if any) are assigned for "HW Low Limit Switch Input", "HW High Limit Switch Input", and "Input reference point switch". Then open the Device configuration for the CPU and examine each of the assigned inputs. Verify the "Enable rising edge detection" and "Enable falling edge detection" are both selected. If these properties are not selected, delete the specified inputs in the axis configuration and select them again.

10.3.6.4 Jerk limit

With the jerk limit you can reduce the stresses on your mechanics during an acceleration and deceleration ramp. The value for the acceleration and deceleration is not changed abruptly when the step limiter is active; it is adapted in a transition phase. The figure below shows the velocity and acceleration curve without and with jerk limit.

Table 10- 54 Jerk limit

Travel without step limiter	Travel with step limiter
	

The jerk limit gives a "smoothed" velocity profile of the axis motion. This ensures soft starting and braking of a conveyor belt for example.

10.3.7 Motion control instructions

10.3.7.1 MC instruction overview

The motion control instructions use an associated technology data block and the dedicated PTO (pulse train outputs) of the CPU to control the motion on an axis.

- MC_Power (Page 572) enables and disables a motion control axis.
- MC_Reset (Page 575) resets all motion control errors. All motion control errors that can be acknowledged are acknowledged.
- MC_Home (Page 576) establishes the relationship between the axis control program and the axis mechanical positioning system.
- MC_Halt (Page 578) cancels all motion processes and causes the axis motion to stop. The stop position is not defined.
- MC_MoveAbsolute (Page 580) starts motion to an absolute position. The job ends when the target position is reached.
- MC_MoveRelative (Page 582) starts a positioning motion relative to the start position.
- MC_MoveVelocity (Page 584) causes the axis to travel with the specified speed.
- MC_MoveJog (Page 587) executes jog mode for testing and startup purposes.
- MC_CommandTable (Page 589) runs axis commands as a movement sequence.
- MC_ChangeDynamic (Page 592) changes Dynamics settings for the axis.
- MC_WriteParam (Page 594) writes a select number of parameters to change the functionality of the axis from the user program.
- MC_ReadParam (Page 596) reads a select number of parameters that indicate the current position, velocity, and so forth of the axis defined in the Axis input.

CPU firmware levels

If you have an S7-1200 CPU with V4.1 firmware, select the V5.0 version of each motion instruction.

If you have an S7-1200 CPU with V4.0 or earlier firmware, select the applicable V4.0, V3.0, V2.0, or V1.0 version of each motion instruction.

Note

The CPU calculates motion tasks in "slices" or segments of 10 ms. As one slice is being executed, the next slice is waiting in the queue to be executed. If you interrupt the motion task on an axis (by executing another new motion task for that axis), the new motion task may not be executed for a maximum of 20 ms (the remainder of the current slice plus the queued slice).

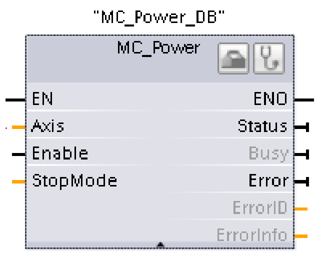
10.3.7.2 MC_Power (Release/block axis) instruction

Note

If the axis is switched off due to an error, it will be enabled again automatically after the error has been eliminated and acknowledged. This requires that the Enable input parameter has retained the value TRUE during this process.

Table 10- 55 MC_Power instruction

LAD / FBD	SCL	Description

 "MC_Power_DB"
MC_Power
EN
ENO
Axis
Status
Enable
Busy
StopMode
Error
ErrorID
ErrorInfo
	`"MC_Power_DB"(`
	`Axis:=_multi_fb_in_,`
	`Enable:=_bool_in_,`
	`StopMode:=_int_in_,`
	`Status=>_bool_out_,`
	`Busy=>_bool_out_,`
	`Error=>_bool_out_,`
	`ErrorID=>_word_out_,`
	`ErrorIn-`
	`fo=>_word_out_);`
		The MC_Power motion control instruction enables or disables an axis. Before you can enable or disable the axis, ensure the following conditions:
		• The technology object has been configured correctly.
		• There is no pending enable-inhibiting error.
		The execution of MC_Power cannot be aborted by a motion control task. Disabling the axis (input parameter Enable = FALSE) aborts all motion control tasks for the associated technology object.

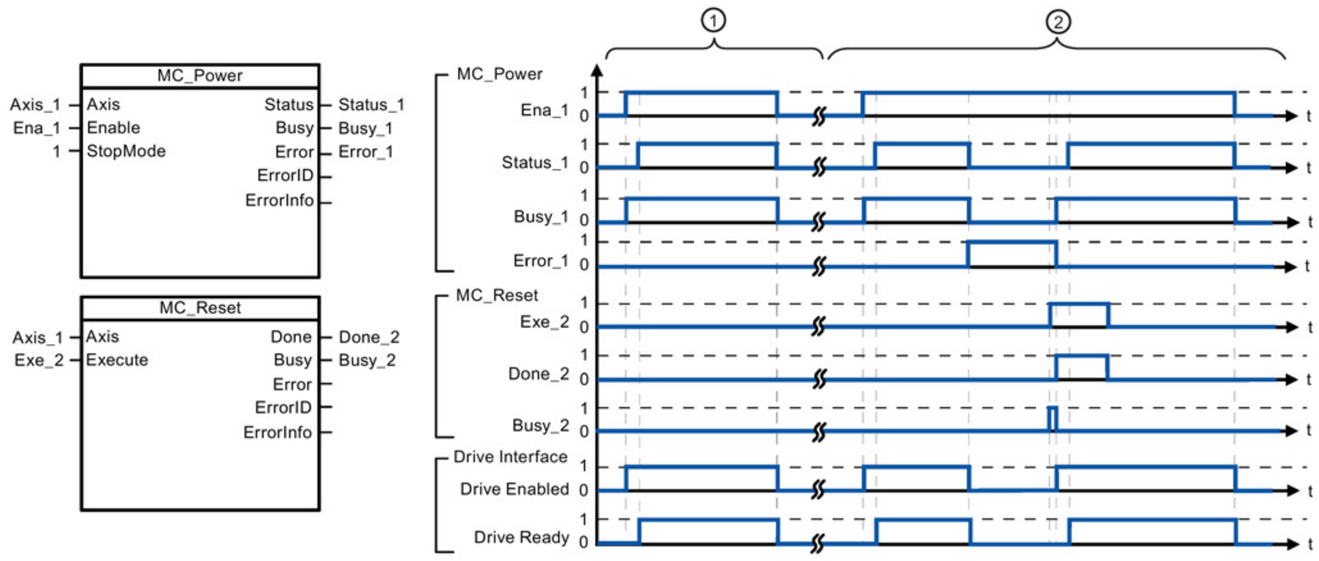
[1] STEP 7 automatically creates the DB when you insert the instruction.

[2] In the SCL example, "MC_Power_DB" is the name of the instance DB.

Table 10- 56 Parameters for the MC_Power instruction

Parameter and type		Data type	Description
Axis	IN	TO_Axis_1	Axis technology object
Enable	IN	Bool	• FALSE (default): All active tasks are aborted according to the parameterized "StopMode" and the axis is stopped. • TRUE: Motion Control attempts to enable the axis.
StopMode	IN	Int	• 0: Emergency stop: If a request to disable the axis is pending, the axis brakes at the configured emergency deceleration. The axis is disabled after reaching standstill. • 1: Immediate stop: If a request to disable the axis is pending, this axis is disabled without deceleration. Pulse output is stopped immediately. • 2: Emergency stop with jerk control: If a request to disable the axis is pending, the axis brakes at the configured emergency stop deceleration. If the jerk control is activated, the configured jerk is taken into account. The axis is disabled after reaching standstill.

Parameter and type		Data type	Description
Status	OUT	Bool	Status of axis enable: • FALSE: The axis is disabled: – The axis does not execute motion control tasks and does not accept any new tasks (exception: MC_Reset task). – The axis is not homed. – Upon disabling, the status does not change to FALSE until the axis reaches a standstill. • TRUE: The axis is enabled: – The axis is ready to execute motion control tasks. – Upon axis enabling, the status does not change to TRUE until the signal "Drive ready" is pending. If the "Drive ready" drive interface was not configured in the axis configuration, the status changes to TRUE immediately.
Busy	OUT	Bool	FALSE: MC_Power is not active. TRUE: MC_Power is active.
Error	OUT	Bool	FALSE: No error TRUE: An error has occurred in motion control instruction "MC_Power" or in the associated technology object. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUT	Word	Error ID for parameter "Error""
ErrorInfo	OUT	Word	Error info ID for parameter "ErrorID"



① An axis is enabled and then disabled again. After the drive has signaled "Drive ready" back to the CPU, the successful enable can be read out via "Status_1".

② Following an axis enable, an error has occurred that caused the axis to be disabled. The error is eliminated and acknowledged with "MC_Reset". The axis is then enabled again.

To enable an axis with configured drive interface, follow these steps:

1. Check the requirements indicated above.
2. Initialize input parameter "StopMode" with the desired value. Set input parameter "Enable" to TRUE.

   The enable output for "Drive enabled" changes to TRUE to enable the power to the drive. The CPU waits for the "Drive ready" signal of the drive.

   When the "Drive ready" signal is available at the configured ready input of the CPU, the axis becomes enabled. Output parameter "Status" and technology object tag <Axis name>.StatusBits.Enable indicates the value TRUE.

To enable an axis without configured drive interface, follow these steps:

1. Check the requirements indicated above.
2. Initialize input parameter "StopMode" with the desired value. Set input parameter "Enable" to TRUE. The axis is enabled. Output parameter "Status" and technology object tag <Axis name>.StatusBits.Enable indicate the value TRUE.

To disable an axis, follow these steps:

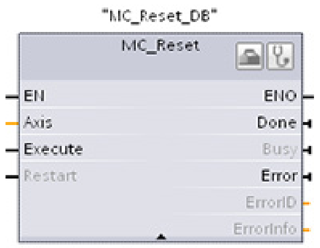
1. Bring the axis to a standstill.

   You can identify when the axis is at a standstill in technology object tag <Axis name>.StatusBits.StandStill.
2. Set input parameter "Enable" to FALSE after standstill is reached.
3. If output parameters "Busy" and "Status" and technology object tag <Axis name>.StatusBits.Enable indicate the value FALSE, disabling of the axis is complete.

10.3.7.3 MC_Reset (Confirm error) instruction

Table 10- 57 MC_Reset instruction

LAD / FBD	SCL	Description

	`"MC_Reset_DB"(`
	`    Axis:=_multi_fb_in_,`
	`    Execute:=_bool_in_,`
	`    Restart:=_bool_in_,`
	`    Done=>_bool_out_,`
	`    Busy=>_bool_out_,`
	`    Error=>_bool_out_,`
	`    ErrorID=>_word_out_,`
	`    ErrorInfo=>_word_out_);`
		Use the MC_Reset instruction to acknowledge "Operating error with axis stop" and "Configuration error". The errors that require acknowledgement can be found in the "List of ErrorIDs and ErrorInfos" under "Remedy".
		Before using the MC_Reset instruction, you must have eliminated the cause of a pending configuration error requiring acknowledgement (for example, by changing an invalid acceleration value in "Axis" technology object to a valid value).
		As of V3.0 and later, the Restart command allows the axis configuration to be downloaded to the work memory in the RUN operating mode.

1 STEP 7 automatically creates the DB when you insert the instruction.

2 In the SCL example, "MC_Reset_DB" is the name of the instance DB.

The MC_Reset task cannot be aborted by any other motion control task. The new MC_Reset task does not abort any other active motion control tasks.

Table 10- 58 Parameters of the MC_Reset instruction

Parameter and type		Data type	Description
Axis	IN	TO_Axis_1	Axis technology object
Execute	IN	Bool	Start of the task with a positive edge
Restart	IN	Bool	TRUE = Download the axis configuration from the load memory to the work memory. The command can only be executed when the axis is disabled.
			FALSE = Acknowledges pending errors
Done	OUT	Bool	TRUE = Error has been acknowledged.
Busy	OUT	Bool	TRUE = The task is being executed.
Error	OUT	Bool	TRUE = An error has occurred during execution of the task. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUTP	Word	Error ID for parameter "Error""
ErrorInfo	OUT	Word	Error info ID for parameter "ErrorID"

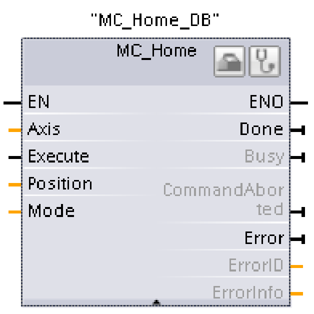
To acknowledge an error with MC_Reset, follow these steps:

1. Check the requirements indicated above.
2. Start the acknowledgement of the error with a rising edge at the Execute input parameter.
3. The error has been acknowledged when Done equals TRUE and the technology object tag <Axis name>.StatusBits.Error equals FALSE.

10.3.7.4 MC_Home (Home axis) instruction

Table 10- 59 MC_Home instruction

LAD / FBD	SCL	Description

 "MC_Home_DB"
MC_Home
EN
ENO
Axis
Done
Execute
Busy
Position
CommandAborted
Mode
Error
ErrorID
ErrorInfo
	`"MC_Home_DB"(`
	`Axis:=_multi_fb_in_,`
	`Execute:=_bool_in_,`
	`Position:=_real_in_,`
	`Mode:=_int_in_,`
	`Done=>_bool_out_,`
	`Busy=>_bool_out_,`
	`CommandAborted=>_bool_out_,`
	`Error=>_bool_out_,`
	`ErrorID=>_word_out_,`
	`ErrorInfo=>_word_out_);`
		Use the MC_Home instruction to match the axis coordinates to the real, physical drive position. Homing is required for absolute positioning of the axis:
		In order to use the MC_Home instruction, the axis must first be enabled.

1 STEP 7 automatically creates the DB when you insert the instruction.

2 In the SCL example, "MC_Home_DB" is the name of the instance DB.

The following types of homing are available:

- Direct homing absolute (Mode = 0): The current axis position is set to the value of parameter "Position".
- Direct homing relative (Mode = 1): The current axis position is offset by the value of parameter "Position".
- Passive homing (Mode = 2): During passive homing, the MC_Home instruction does not carry out any homing motion. The traversing motion required for this step must be implemented by the user via other motion control instructions. When the reference point switch is detected, the axis is homed.
- Active homing (Mode = 3): The homing procedure is executed automatically.

Table 10- 60 Parameters for the MC_Home instruction

Parameter and type		Data type	Description
Axis	IN	TO_Axis_PTO	Axis technology object
Execute	IN	Bool	Start of the task with a positive edge
Position	IN	Real	• Mode = 0, 2, and 3 (Absolute position of axis after completion of the homing operation) • Mode = 1 (Correction value for the current axis position) Limit values: $-1.0e^{12} \le$ Position $\le 1.0e^{12}$

Parameter and type		Data type	Description
Mode	IN	Int	Homing mode • 0: Direct homing absolute New axis position is the position value of parameter "Posi-tion". • 1: Direct homing relative New axis position is the current axis position + position value of parameter "Position". • 2: Passive homing Homing according to the axis configuration. Following homing, the value of parameter "Position" is set as the new axis position. • 3: Active homing Reference point approach in accordance with the axis configuration. Following homing, the value of parameter "Position" is set as the new axis position.
Done	OUT	Bool	TRUE = Task completed
Busy	OUT	Bool	TRUE = The task is being executed.
CommandAborted	OUT	Bool	TRUE = During execution the task was aborted by another task.
Error	OUT	Bool	TRUE = An error has occurred during execution of the task. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUT	Word	Error ID for parameter "Error""
ErrorInfo	OUT	Word	Error info ID for parameter "ErrorID"

Note

Axis homing is lost under the following conditions

- Disabling of axis by the MC_Power instruction
- Switchover between automatic control and manual control
- Upon start of active homing (After successful completion of the homing operation, axis homing is available again.)
- After power-cycling the CPU
- After CPU restart (RUN-to-STOP or STOP-to-RUN)

To home the axis, follow these steps:

1. Check the requirements indicated above.
2. Initialize the necessary input parameters with values, and start the homing operation with a rising edge at input parameter "Execute".
3. If output parameter "Done" and technology object tag <Axis name>.StatusBits.HomingDone indicate the value TRUE, homing is complete.

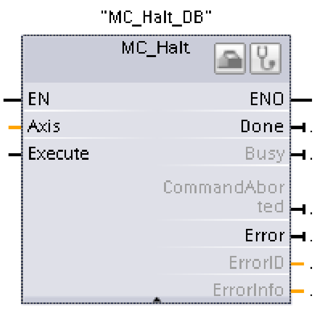
Table 10- 61 Override response

Mode	Description	
0 or 1	The MC_Home task cannot be aborted by any other motion control task. The new MC_Home task does not abort any active motion control tasks. Position-related motion tasks are resumed after homing according to the new homing position (value at the Position input parameter).	
2	The MC_Home task can be aborted by the following motion control tasks: MC_Home task Mode = 2, 3: The new MC_Home task aborts the following active motion control task. MC_Home task Mode = 2: Position-related motion tasks are resumed after homing according to the new homing position (value at the Position input parameter).	
3	The MC_Home task can be aborted by the following motion control tasks: • MC_Home Mode = 3 • MC_Halt • MC_MoveAbsolute • MC_MoveRelative • MC_MoveVelocity • MC_MoveJog	The new MC_Home task aborts the following active motion control tasks: • MC_Home Mode = 2, 3 • MC_Halt • MC_MoveAbsolute • MC_MoveRelative • MC_MoveVelocity • MC_MoveJog

10.3.7.5 MC_Halt (Pause axis) instruction

Table 10- 62 MC_Halt instruction

LAD / FBD	SCL	Description

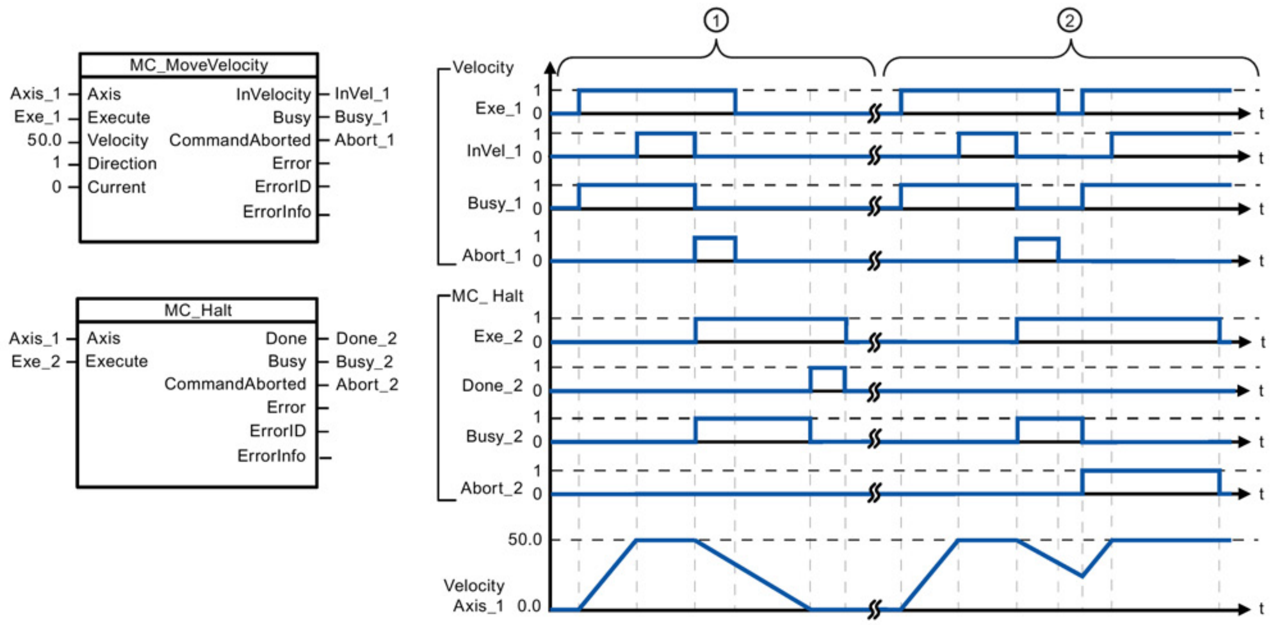
 "MC_Halt_DB"
MC_Halt
EN
ENO
Axis
Done
Execute
Busy
CommandAborted
Error
ErrorID
ErrorInfo
	```
	"MC_Halt_DB"(
	    Axis:=_multi_fb_in_,
	    Execute:=_bool_in_,
	    Done=>_bool_out_,
	    Busy=>_bool_out_,
	    CommandAborted=>_bool_out_,
	    Error=>_bool_out_,
	    ErrorID=>_word_out_,
	    ErrorInfo=>_word_out_);
	```
		Use the MC_Halt instruction to stop all motion and to bring the axis to a stand-still. The stand-still position is not defined.
		In order to use the MC_Halt instruction, the axis must first be enabled.

1 STEP 7 automatically creates the DB when you insert the instruction.

2 In the SCL example, "MC_Halt_DB" is the name of the instance DB.

Table 10- 63 Parameters for the MC_Halt instruction

Parameter and type		Data type	Description
Axis	IN	TO_Axis_1	Axis technology object
Execute	IN	Bool	Start of the task with a positive edge
Done	OUT	Bool	TRUE = Zero velocity reached
Busy	OUT	Bool	TRUE = The task is being executed.
CommandAborted	OUT	Bool	TRUE = During execution the task was aborted by another task.
Error	OUT	Bool	TRUE = An error has occurred during execution of the task. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUT	Word	Error ID for parameter "Error"
ErrorInfo	OUT	Word	Error info ID for parameter "ErrorID"



The following values were configured in the "Dynamics > General" configuration window: Acceleration = 10.0 and Deceleration = 5.0

① The axis is braked by an MC_Halt task until it comes to a standstill. The axis standstill is signaled via "Done_2".

② While an MC_Halt task is braking the axis, this task is aborted by another motion task. The abort is signaled via "Abort_2".

Override response

The MC_Halt task can be aborted by the following motion control tasks:

- MC_Home Mode = 3
- MC_Halt
- MC_MoveAbsolute
- MC_MoveRelative
- MC_MoveVelocity
- MC_MoveJog

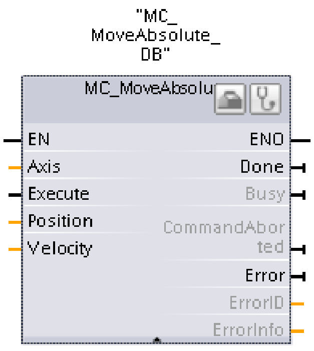
The new MC_Halt task aborts the following active motion control tasks:

- MC_Home Mode = 3
- MC_Halt
- MC_MoveAbsolute
- MC_MoveRelative
- MC_MoveVelocity
- MC_MoveJog

10.3.7.6 MC_MoveAbsolute (Position axis absolutely) instruction

Table 10- 64 MC_MoveAbsolute instruction

LAD / FBD	SCL	Description

 "MC_MoveAbsolute_DB"
MC_MoveAbsolu
EN
ENO
Axis
Done
Execute
Busy
Position
CommandAborted
Velocity
Error
ErrorID
ErrorInfo
	`"MC_MoveAbsolute_DB"(`
	`Axis:=_multi_fb_in_,`
	`Execute:=_bool_in_,`
	`Position:=_real_in_,`
	`Velocity:=_real_in_,`
	`Done=>_bool_out_,`
	`Busy=>_bool_out_,`
	`CommandAborted=>_bool_out_,`
	`Error=>_bool_out_,`
	`ErrorID=>_word_out_,`
	`ErrorInfo=>_word_out_);`
		Use the MC_MoveAbsolute instruction to start a positioning motion of the axis to an absolute position.
		In order to use the MC_MoveAbsolute instruction, the axis must first be enabled and also must be homed.

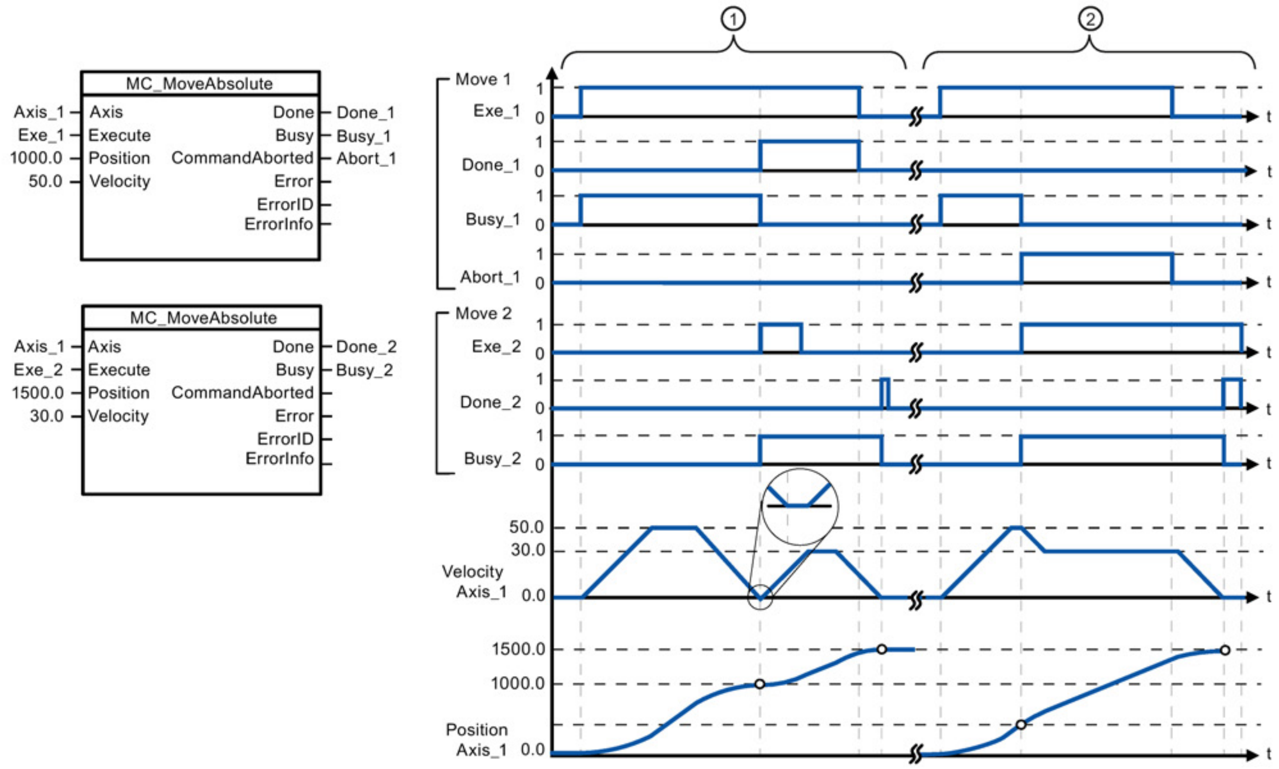
1 STEP 7 automatically creates the DB when you insert the instruction.

2 In the SCL example, "MC_MoveAbsolute_DB" is the name of the instance DB.

Table 10- 65 Parameters for the MC_MoveAbsolute instruction

Parameter and type		Data type	Description
Axis	IN	TO_Axis_1	Axis technology object
Execute	IN	Bool	Start of the task with a positive edge (Default value: False)
Position	IN	Real	Absolute target position (Default value: 0.0) Limit values: $-1.0e^{12} \leq$ Position $\leq 1.0e^{12}$
Velocity	IN	Real	Velocity of axis (Default value: 10.0) This velocity is not always reached because of the configured acceleration and deceleration and the target position to be approached. Limit values: Start/stop velocity ≤ Velocity ≤ maximum velocity
Done	OUT	Bool	TRUE = Absolute target position reached
Busy	OUT	Bool	TRUE = The task is being executed.
CommandAborted	OUT	Bool	TRUE = During execution the task was aborted by another task.

Parameter and type		Data type	Description
Error	OUT	Bool	TRUE = An error has occurred during execution of the task. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUT	Word	Error ID for parameter "Error" (Default value: 0000)
ErrorInfo	OUT	Word	Error info ID for parameter "ErrorID" (Default value: 0000)



The following values were configured in the "Dynamics > General" configuration window: Acceleration = 10.0 and Deceleration = 10.0

① An axis is moved to absolute position 1000.0 with a MC_MoveAbsolute task. When the axis reaches the target position, this is signaled via "Done_1". When "Done_1" = TRUE, another MC_MoveAbsolute task, with target position 1500.0, is started. Because of the response times (e.g., cycle time of user program, etc.), the axis comes to a standstill briefly (see zoomed-in detail). When the axis reaches the new target position, this is signaled via "Done_2".

② An active MC_MoveAbsolute task is aborted by another MC_MoveAbsolute task. The abort is signaled via "Abort_1". The axis is then moved at the new velocity to the new target position 1500.0. When the new target position is reached, this is signaled via "Done_2".

Override response

The MC_MoveAbsolute task can be aborted by the following motion control tasks:

- MC_Home Mode = 3
- MC_Halt
- MC_MoveAbsolute
- MC_MoveRelative
- MC_MoveVelocity
- MC_MoveJog

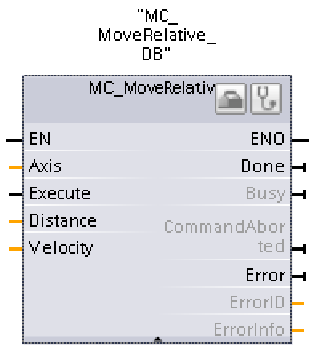
The new MC_MoveAbsolute task aborts the following active motion control tasks:

- MC_Home Mode = 3
- MC_Halt
- MC_MoveAbsolute
- MC_MoveRelative
- MC_MoveVelocity
- MC_MoveJog

10.3.7.7 MC_MoveRelative (Position axis relatively) instruction

Table 10- 66 MC_MoveRelative instruction

LAD / FBD	SCL	Description

 "MC_MoveRelative_DB"
MC_MoveRelativ
EN
ENO
Axis
Done
Execute
Busy
Distance
CommandAborted
Velocity
Error
ErrorID
ErrorInfo
	`"MC_MoveRelative_DB"(`
	`    Axis:=_multi_fb_in_,`
	`    Execute:=_bool_in_,`
	`    Distance:=_real_in_,`
	`    Velocity:=_real_in_,`
	`    Done=>_bool_out_,`
	`    Busy=>_bool_out_,`
	`    CommandAborted=>_bool_out_,`
	`    Error=>_bool_out_,`
	`    ErrorID=>_word_out_,`
	`    ErrorInfo=>_word_out_);`
		Use the MC_MoveRelative instruction to start a positioning motion relative to the start position.
		In order to use the MC_MoveRelative instruction, the axis must first be enabled.

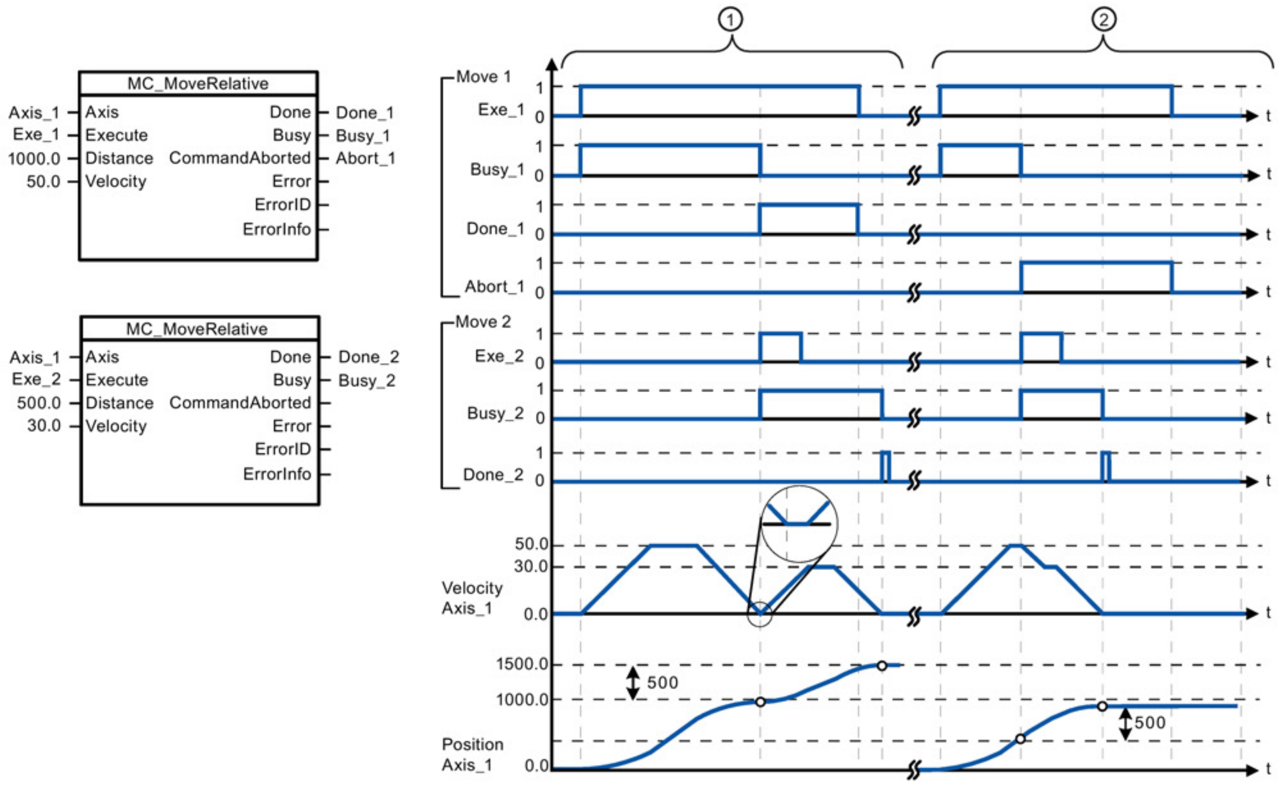
[1] STEP 7 automatically creates the DB when you insert the instruction.

[2] In the SCL example, "MC_MoveRelative_DB " is the name of the instance DB.

Table 10- 67 Parameters for the MC_MoveRelative instruction

Parameter and type		Data type	Description
Axis	IN	TO_Axis_1	Axis technology object
Execute	IN	Bool	Start of the task with a positive edge (Default value: False)
Distance	IN	Real	Travel distance for the positioning operation (Default value: 0.0) Limit values: $-1.0e^{12} \leq$ Distance $\leq 1.0e^{12}$
Velocity	IN	Real	Velocity of axis (Default value: 10.0) This velocity is not always reached on account of the configured acceleration and deceleration and the distance to be traveled. Limit values: Start/stop velocity ≤ Velocity ≤ maximum velocity
Done	OUT	Bool	TRUE = Target position reached
Busy	OUT	Bool	TRUE = The task is being executed.

Parameter and type		Data type	Description
CommandAborted	OUT	Bool	TRUE = During execution the task was aborted by another task.
Error	OUT	Bool	TRUE = An error has occurred during execution of the task. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUT	Word	Error ID for parameter "Error" (Default value: 0000)
ErrorInfo	OUT	Word	Error info ID for parameter "ErrorID" (Default value: 0000)



The following values were configured in the "Dynamics > General" configuration window: Acceleration = 10.0 and Deceleration = 10.0

① The axis is moved by an MC_MoveRelative task by the distance ("Distance") 1000.0. When the axis reaches the target position, this is signaled via "Done_1". When "Done_1" = TRUE, another MC_MoveRelative task, with travel distance 500.0, is started. Because of the response times (for example, cycle time of user program), the axis comes to a standstill briefly (see zoomed-in detail). When the axis reaches the new target position, this is signaled via "Done_2".

② An active MC_MoveRelative task is aborted by another MC_MoveRelative task. The abort is signaled via "Abort_1". The axis is then moved at the new velocity by the new distance ("Distance") 500.0. When the new target position is reached, this is signaled via "Done_2".

Override response

The MC_MoveRelative task can be aborted by the following motion control tasks:

- MC_Home Mode = 3
- MC_Halt
- MC_MoveAbsolute
- MC_MoveRelative
- MC_MoveVelocity
- MC_MoveJog

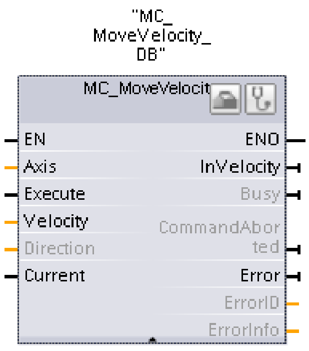
The new MC_MoveRelative task aborts the following active motion control tasks:

- MC_Home Mode = 3
- MC_Halt
- MC_MoveAbsolute
- MC_MoveRelative
- MC_MoveVelocity
- MC_MoveJog

10.3.7.8 MC_MoveVelocity (Move axis at predefined velocity) instruction

Table 10- 68 MC_MoveVelocity instruction

LAD / FBD	SCL	Description

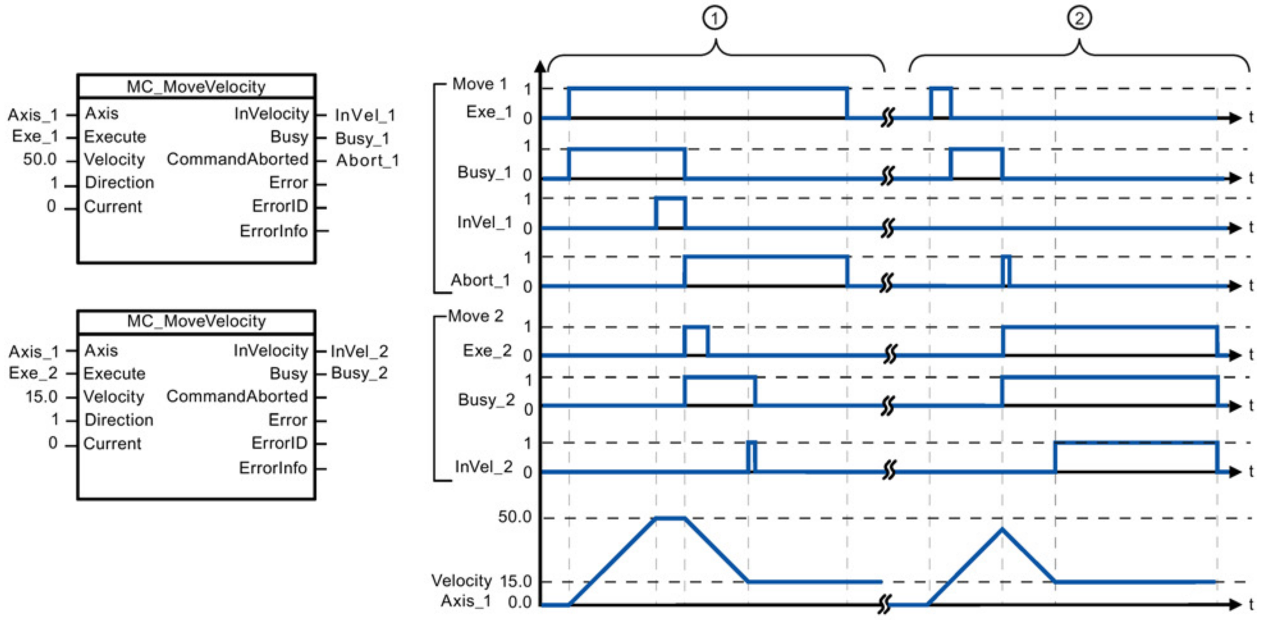
 "MC_ MoveVelocity_ DB"
MC_MoveVelocit
EN
ENO
Axis
InVelocity
Execute
Busy
Velocity
CommandAborted
Direction
Current
Error
ErrorID
ErrorInfo
	`"MC_MoveVelocity_DB"(`
	`    Axis:=_multi_fb_in_,`
	`    Execute:=_bool_in_,`
	`    Velocity:=_real_in_,`
	`    Direction:=_int_in_,`
	`    Current:=_bool_in_,`
	`    InVelocity=>_bool_out_,`
	`    Busy=>_bool_out_,`
	`    CommandAborted=>_bool_out_,`
	`    Error=>_bool_out_,`
	`    ErrorID=>_word_out_,`
	`    ErrorInfo=>_word_out_);`
		Use the MC_MoveVelocity instruction to move the axis constantly at the specified velocity.
		In order to use the MC_MoveVelocity instruction, the axis must first be enabled.

1 STEP 7 automatically creates the DB when you insert the instruction.

2 In the SCL example, "MC_MoveVelocity_DB " is the name of the instance DB.

Table 10- 69 Parameters for the MC_MoveVelocity instruction

Parameter and type		Data type	Description
Axis	IN	TO_Axis_1	Axis technology object
Execute	IN	Bool	Start of the task with a positive edge (Default value: False)
Velocity	IN	Real	Velocity specification for axis motion (Default value: 10.0) Limit values: Start/stop velocity ≤ |Velocity| ≤ maximum velocity (Velocity = 0.0 is allowed)
Direction	IN	Int	Direction specification: • 0: Direction of rotation corresponds to the sign of the value in parameter "Velocity" (Default value) • 1: Positive direction of rotation (The sign of the value in parameter "Velocity" is ignored.) • 2: Negative direction of rotation (The sign of the value in parameter "Velocity" is ignored.)
Current	IN	Bool	Maintain current velocity: • FALSE: "Maintain current velocity" is deactivated. The values of parameters "Velocity" and "Direction" are used. (Default value) • TRUE: "Maintain current velocity" is activated. The values in parameters "Velocity" and "Direction" are not taken into account. When the axis resumes motion at the current velocity, the "InVelocity" parameter returns the value TRUE.
InVelocity	OUT	Bool	TRUE: • If "Current" = FALSE: The velocity specified in parameter "Velocity" was reached. • If "Current" = TRUE: The axis travels at the current velocity at the start time.
Busy	OUT	Bool	TRUE = The task is being executed.
CommandAborted	OUT	Bool	TRUE = During execution the task was aborted by another task.
Error	OUT	Bool	TRUE = An error has occurred during execution of the task. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUT	Word	Error ID for parameter "Error" (Default value: 0000)
ErrorInfo	OUT	Word	Error info ID for parameter "ErrorID" (Default value: 0000)



The following values were configured in the "Dynamics > General" configuration window: Acceleration = 10.0 and Deceleration = 10.0

① An active MC_MoveVelocity task signals via "InVel_1" that its target velocity has been reached. It is then aborted by another MC_MoveVelocity task. The abort is signaled via "Abort_1". When the new target velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the new constant velocity.

② An active MC_MoveVelocity task is aborted by another MC_MoveVelocity task prior to reaching its target velocity. The abort is signaled via "Abort_1". When the new target velocity 15.0 is reached, this is signaled via "InVel_2". The axis then continues moving at the new constant velocity.

Override response

The MC_MoveVelocity task can be aborted by the following motion control tasks:

- MC_Home Mode = 3
- MC_Halt
- MC_MoveAbsolute
- MC_MoveRelative
- MC_MoveVelocity
- MC_MoveJog

The new MC_MoveVelocity task aborts the following active motion control tasks:

- MC_Home Mode = 3
- MC_Halt
- MC_MoveAbsolute
- MC_MoveRelative
- MC_MoveVelocity
- MC_MoveJog

Note

Behavior with zero set velocity (Velocity = 0.0)

An MC_MoveVelocity task with "Velocity" = 0.0 (such as an MC_Halt task) aborts active motion tasks and stops the axis with the configured deceleration. When the axis comes to a standstill, output parameter "InVelocity" indicates TRUE for at least one program cycle.

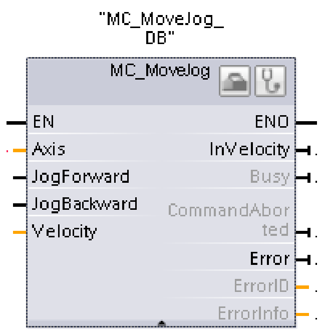
"Busy" indicates the value TRUE during the deceleration operation and changes to FALSE together with "InVelocity". If parameter "Execute" = TRUE is set, "InVelocity" and "Busy" are latched.

When the MC_MoveVelocity task is started, status bit "SpeedCommand" is set in the technology object. Status bit "ConstantVelocity" is set upon axis standstill. Both bits are adapted to the new situation when a new motion task is started.

10.3.7.9 MC_MoveJog (Move axis in jog mode) instruction

Table 10- 70 MC_MoveJog instruction

LAD / FBD	SCL	Description

 "MC_MoveJog_DB"
MC_MoveJog
EN
ENO
Axis
InVelocity
JogForward
Busy
JogBackward
CommandAborted
Velocity
Error
ErrorID
ErrorInfo
	`"MC_MoveJog_DB"(`
	`    Axis:=_multi_fb_in_,`
	`    JogForward:=_bool_in_,`
	`    JogBackward:=_bool_in_,`
	`    Velocity:=_real_in_,`
	`    InVelocity=>_bool_out_,`
	`    Busy=>_bool_out_,`
	`    CommandAborted=>_bool_out_,`
	`    Error=>_bool_out_,`
	`    ErrorID=>_word_out_,`
	`    ErrorInfo=>_word_out_);`
		Use the MC_MoveJog instruction to move the axis constantly at the specified velocity in jog mode. This instruction is typically used for testing and commissioning purposes.
		In order to use the MC_MoveJog instruction, the axis must first be enabled.

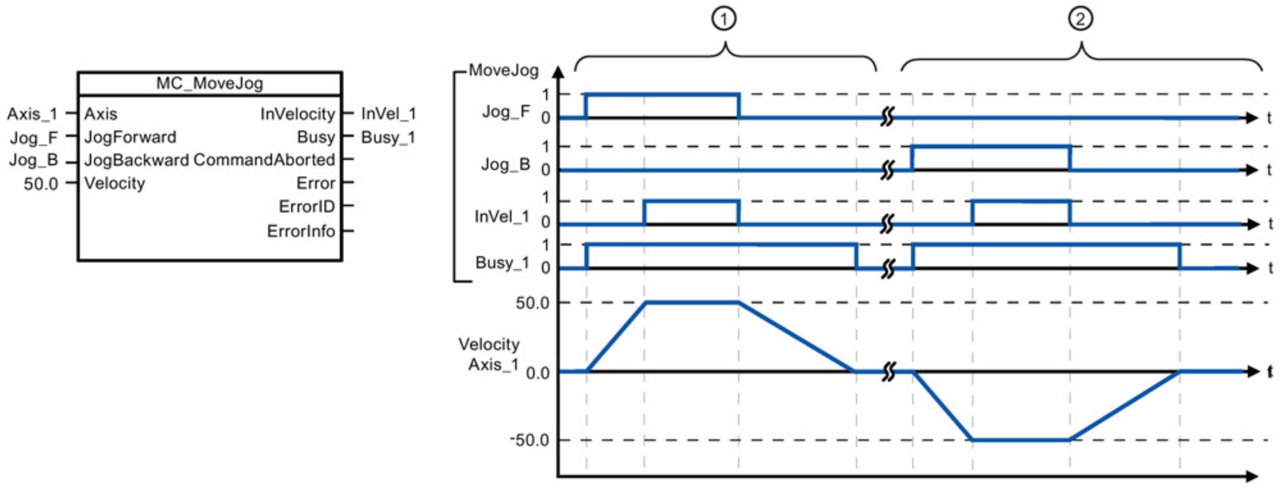
1 STEP 7 automatically creates the DB when you insert the instruction.

2 In the SCL example, "MC_MoveJog_DB " is the name of the instance DB.

Table 10- 71 Parameters for the MC_MoveJog instruction

Parameter and type		Data type	Description
Axis	IN	TO_Axis_1	Axis technology object
JogForward[1]	IN	Bool	As long as the parameter is TRUE, the axis moves in the positive direction at the velocity specified in parameter "Velocity". The sign of the value in parameter "Velocity" is ignored. (Default value: False)
JogBackward[1]	IN	Bool	As long as the parameter is TRUE, the axis moves in the negative direction at the velocity specified in parameter "Velocity". The sign of the value in parameter "Velocity" is ignored. (Default value: False)
Velocity	IN	Real	Preset velocity for jog mode (Default value: 10.0) Limit values: Start/stop velocity ≤ |Velocity| ≤ maximum velocity
InVelocity	OUT	Bool	TRUE = The velocity specified in parameter "Velocity" was reached.
Busy	OUT	Bool	TRUE = The task is being executed.
CommandAborted	OUT	Bool	TRUE = During execution the task was aborted by another task.
Error	OUT	Bool	TRUE = An error has occurred during execution of the task. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo".
ErrorID	OUT	Word	Error ID for parameter "Error" (Default value: 0000)
ErrorInfo	OUT	Word	Error info ID for parameter "ErrorID" (Default value: 0000)

[1] If both the JogForward and JogBackward parameters are simultaneously TRUE, the axis stops with the configured deceleration. An error is indicated in parameters "Error", "ErrorID", and "ErrorInfo".



The following values were configured in the "Dynamics > General" configuration window: Acceleration = 10.0 and Deceleration = 5.0

① The axis is moved in the positive direction in jog mode via "Jog_F". When the target velocity 50.0 is reached, this is signaled via "InVelo_1". The axis brakes to a standstill again after Jog_F is reset.

② The axis is moved in the negative direction in jog mode via "Jog_B". When the target velocity 50.0 is reached, this is signaled via "InVelo_1". The axis brakes to a standstill again after Jog_B is reset.

Override response

The MC_MoveJog task can be aborted by the following motion control tasks:

- MC_Home Mode = 3
- MC_Halt
- MC_MoveAbsolute
- MC_MoveRelative
- MC_MoveVelocity
- MC_MoveJog

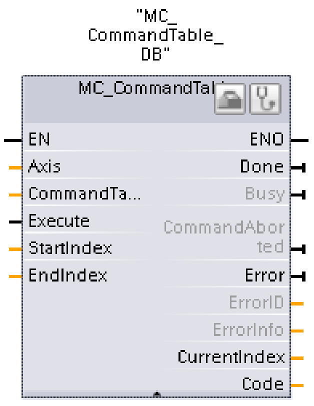
The new MC_MoveJog task aborts the following active motion control tasks:

- MC_Home Mode = 3
- MC_Halt
- MC_MoveAbsolute
- MC_MoveRelative
- MC_MoveVelocity
- MC_MoveJog

10.3.7.10 MC_CommandTable (Run axis commans as movement sequence) instruction

Table 10- 72 MC_CommandTable instruction

LAD / FBD	SCL	Description

 "MC_
CommandTable_
DB"
MC_CommandTa...
EN
ENO
Axis
Done
CommandTa...
Busy
Execute
CommandAborted
StartIndex
EndIndex
Error
ErrorID
ErrorInfo
CurrentIndex
Code
	`"MC_CommandTable_DB"(`
	`Axis:=_multi_fb_in_,`
	`CommandTable:=_multi_fb_in_,`
	`Execute:=_bool_in_,`
	`StartIndex:=_uint_in_,`
	`EndIndex:=_uint_in_,`
	`Done=>_bool_out_,`
	`Busy=>_bool_out_,`
	`CommandAborted=>_bool_out_,`
	`Error=>_bool_out_,`
	`ErrorID=>_word_out_,`
	`ErrorInfo=>_word_out_,`
	`CurrentIndex=>_uint_out_,`
	`Code=>_word_out_);`
		Executes a series of individual motions for a motor control axis that can combine into a movement sequence.
		Individual motions are configured in a technology object command table for pulse train output (TO_CommandTable_PTO).

1 STEP 7 automatically creates the DB when you insert the instruction.

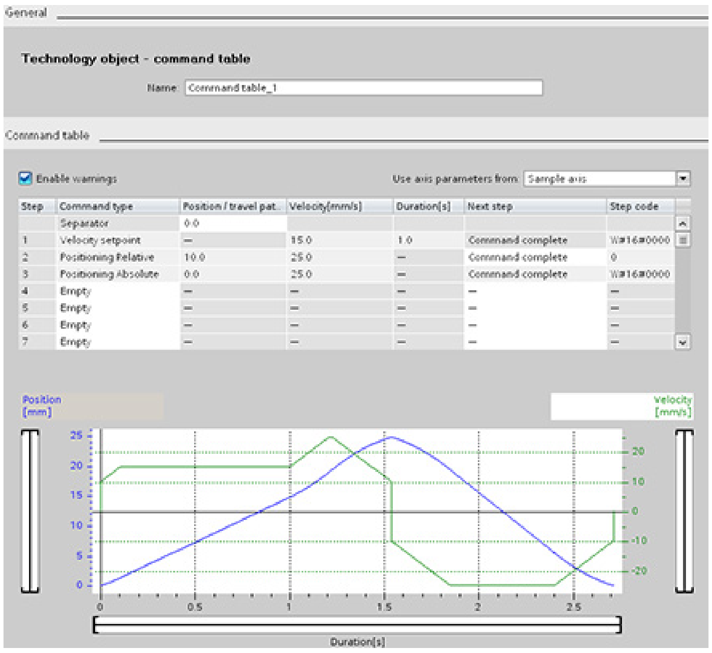
2 In the SCL example, "MC_CommandTable_DB " is the name of the instance DB.

Table 10- 73 Parameters for the MC_CommandTable instruction

Parameter and type		Data type	Initial value	Description
Axis	IN	TO_Axis_1	-	Axis technology object
Table	IN	TO_CommandTable_1	-	Command table technology object
Execute	IN	Bool	FALSE	Start job with rising edge
StartIndex	IN	Int	1	Start command table processing with this step Limits: 1 ≤ **StartIndex** ≤ EndIndex
EndIndex	IN	Int	32	End command table processing with this step Limits: StartIndex ≤ **EndIndex** ≤ 32
Done	OUT	Bool	FALSE	MC_CommandTable processing completed successfully

Parameter and type		Data type	Initial value	Description
Busy	OUT	Bool	FALSE	Operation in progress
CommandAborted	OUT	Bool	FALSE	The task was aborted during processing by another task.
Error	OUT	Bool	FALSE	An error occurred during processing. The cause is indicated by the parameters ErrorID and ErrorInfo.
ErrorID	OUT	Word	16#0000	Error identifier
ErrorInfo	OUT	Word	16#0000	Error information
Step	OUT	Int	0	Step currently in process
Code	OUT	Word	16#0000	User defined identifier of the step currently in process

You can create the desired movement sequence in the "Command Table" configuration window and check the result against the graphic view in the trend diagram.



Technology object - command table

Name: Command table_1

☑ Enable warnings

Use axis parameters from: Sample axis

Step	Command type	Position / travel pat..	Velocity[mm/s]	Duration[s]	Next step	Step code
	Separator	0.0				
1	Velocity setpoint	—	15.0	1.0	Command complete	W#16#0000
2	Positioning Relative	10.0	25.0	—	Command complete	0
3	Positioning Absolute	0.0	25.0	—	Command complete	W#16#0000
4	Empty	—	—	—	—	—
5	Empty	—	—	—	—	—
6	Empty	—	—	—	—	—
7	Empty	—	—	—	—	—

You can select the command types that are to be used for processing the command table. Up to 32 jobs can be entered. The commands are processed in sequence.

Table 10- 74 MC_CommandTable command types

Command type	Description
Empty	The empty serves as a placeholder for any commands to be added. The empty entry is ignored when the command table is processed
Halt	Pause axis. Note: The command only takes place after a "Velocity setpoint" command.
Positioning Relative	Positions the axis based upon distance. The command moves the axis by the given distance and velocity.
Positioning Absolute	Positions the axis based upon location. The command moves the axis to the given location, using the velocity specified.

Command type	Description
Velocity setpoint	Moves the axis at the given velocity.
Wait	Waits until the given period is over. "Wait" does not stop an active traversing motion.
Separator	Adds a "Separator" line above the selected line. The separator line allows more than one profile to be defined in a single command table.

Prerequisites for MC_CommandTable execution:

- The technology object TO_Axis_PTO V2.0 must be correctly configured.
- The technology object TO_CommandTable_PTO must be correctly configured.
- The axis must be released.

Override response

The MC_CommandTable task can be aborted by the following motion control tasks:

- MC_Home Mode = 3
- MC_Halt
- MC_MoveAbsolute
- MC_MoveRelative
- MC_MoveVelocity
- MC_MoveJog
- MC_CommandTable

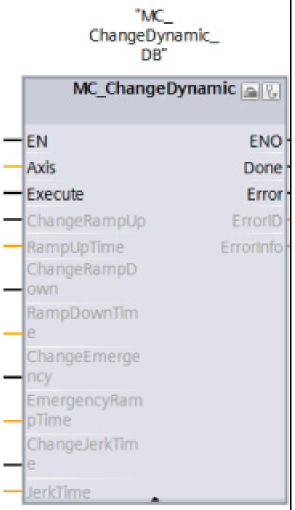
The new MC_CommandTable task aborts the following active motion control tasks:

- MC_Home Mode = 3
- MC_Halt
- MC_MoveAbsolute
- MC_MoveRelative
- MC_MoveVelocity
- MC_MoveJog
- MC_CommandTable
- The current motion control job with the launch of the first "Positioning Relative", "Positioning Absolute", "Velocity setpoint" or "Halt" command

10.3.7.11 MC_ChangeDynamic (Change dynamc settings for the axis) instruction

Table 10- 75 MC_ChangeDynamic instruction

LAD / FBD	SCL	Description

 "MC_ ChangeDynamic_ DB"
MC_ChangeDynamic
EN
Axis
Execute
ChangeRampUp
RampUpTime
ChangeRampDown
RampDownTime
ChangeEmergency
EmergencyRampTime
ChangeJerkTime
JerkTime
ENO
Done
Error
ErrorID
ErrorInfo
	`"MC_ChangeDynamic_DB"(`
	`Execute:=_bool_in_,`
	`ChangeRampUp:=_bool_in_,`
	`RampUpTime:=_real_in_,`
	`ChangeRampDown:=_bool_in_,`
	`RampDownTime:=_real_in_,`
	`ChangeEmergency:=_bool_in_,`
	`EmergencyRampTime:=_real_in_,`
	`ChangeJerkTime:=_bool_in_,`
	`JerkTime:=_real_in_,`
	`Done=>_bool_out_,`
	`Error=>_bool_out_,`
	`ErrorID=>_word_out_,`
	`ErrorInfo=>_word_out_);`
		Changes the dynamic settings of a motion control axis:
		• Change the ramp-up time (acceleration) value
		• Change the ramp-down time (deceleration) value
		• Change the emergency stop ramp-down time (emergency stop deceleration) value
		• Change the smoothing time (jerk) value

1 STEP 7 automatically creates the DB when you insert the instruction.

2 In the SCL example, "MC_ChangeDynamic_DB " is the name of the instance DB.

Table 10- 76 Parameters for the MC_ChangeDynamic instruction

Parameter and type		Data type	Description
Axis	IN	TO_Axis_1	Axis technology object
Execute	IN	Bool	Start of the command with a positive edge. Default value: FALSE
ChangeRampUp	IN	Bool	TRUE = Change ramp-up time in line with input parameter "RampUpTime". Default value: FALSE
RampUpTime	IN	Real	Time (in seconds) to accelerate from standstill to the configured maximum velocity without jerk limit. Default value: 5.00 The change will influence the tag <Axis name>. Config.DynamicDefaults.Acceleration. The effectiveness of the change is shown in the description of this tag.
ChangeRampDown	IN	Bool	TRUE = Change ramp-down time in line with input parameter "RampDownTime". Default value: FALSE
RampDownTime	IN	Real	Time (in seconds) to decelerate axis from the configured maximum velocity to standstill without jerk limiter. Default value: 5.00 The change will influence the tag <Axis name>. Config.DynamicDefaults.Deceleration. The effectiveness of the change is shown in the description of this tag.
ChangeEmergency	IN	Bool	TRUE = Change emergency stop ramp-down time in line with input parameter "EmergencyRampTime" Default value: FALSE

Parameter and type		Data type	Description
EmergencyRampTime	IN	Real	Time (in seconds) to decelerate the axis from configured maximum velocity to standstill without jerk limiter in emergency stop mode. Default value: 2.00 The change will influence the tag <Axis name>. Config.DynamicDefaults.EmergencyDeceleration. The effectiveness of the change is shown in the description of this tag.
ChangeJerkTime	IN	Bool	TRUE = Change smoothing time according to the input parameter "JerkTime". Default value: FALSE
JerkTime	IN	Real	Smoothing time (in seconds) used for the axis acceleration and deceleration ramps. Default value: 0.25 The change will influence the tag <Axis name>. Config.DynamicDefaults.Jerk. The effectiveness of the change is shown in the description of this tag.
Done	OUT	Bool	TRUE = The changed values have been written to the technology data block. The description of the tags will show when the change becomes effective. Default value: FALSE
Error	OUT	Bool	TRUE = An error occurred during execution of the command. The cause of the error can be found in parameters "ErrorID" and "ErrorInfo". Default value: FALSE
ErrorID	OUT	Word	Error identifier. Default value: 16#0000
ErrorInfo	IN	Word	Error information. Default value: 16#0000

Prerequisites for MC_ ChangeDynamic execution:

- The technology object TO_Axis_PTO V2.0 must be correctly configured.
- The axis must be released.

Override response

An MC_ChangeDynamic command cannot be aborted by any other Motion Control command.

A new MC_ChangeDynamic command does not abort any active Motion Control jobs.

Note

The input parameters "RampUpTime", "RampDownTime", "EmergencyRampTime" and "RoundingOffTime" can be specified with values that makes the resultant axis parameters "acceleration", "delay", "emergency stop-delay" and "jerk" outside the permissible limits.

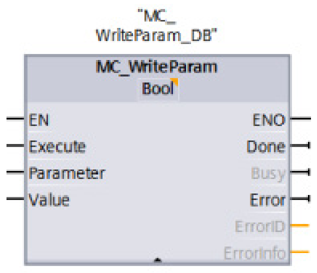
Make sure you keep the MC_ChangeDynamic parameters within the limits of the dynamic configuration settings for the axis technology object.

10.3.7.12 MC_WriteParam (write parameters of a technology object) instruction

You use the MC_WriteParam instruction to write a select number of parameters to change the functionality of the axis from the user program.

Table 10- 77 MC_WriteParam instruction

LAD / FBD	SCL	Description

 "MC_ WriteParam_DB"
MC_WriteParam
Bool
EN
Execute
Parameter
Value
ENO
Done
Busy
Error
ErrorID
ErrorInfo
	`"MC_WriteParam_DB"(`
	`    Parameter:=_variant_in_,`
	`    Value:=_variant_in_,`
	`    Execute:=_bool_in_,`
	`    Done:=_bool_out_,`
	`    Error:=_real_out_,`
	`    ErrorID:=_word_out_,`
	`    ErrorInfo:=_word_out_);`
		You use the MC_WriteParam instruction to write to public parameters (for example, acceleration and user DB values).

1 STEP 7 automatically creates the DB when you insert the instruction.

2 In the SCL example, "MC_WriteParam_DB" is the name of the instance DB.

You can write to the parameters that are public. You cannot write to "MotionStatus" and "StatusBits". The valid parameters are listed in the table below:

Writeable parameter name	Writeable parameter name
Actor.InverseDirection	DynamicDefaults.Acceleration
Actor.DirectionMode	DynamicDefaults.Deceleration
Actor.DriveParameter.PulsesPerDriveRevolution	DynamicDefaults.Jerk
Sensor[1].ActiveHoming.Mode	DynamicDefaults.EmergencyDeceleration
Sensor[1].ActiveHoming.SideInput	PositionLimitsHW.Active
Sensor[1].ActiveHoming.Offset	PositionLimitsHW.MaxSwitchedLevel
Sensor[1].ActiveHoming.SwitchedLevel	PositionLimitsHW.MinSwitchedLevel
Sensor[1].PassiveHoming.Mode	PositionLimitsSW.Active
Sensor[1].PassiveHoming.SideInput	PositionLimitsSW.MinPosition
Sensor[1].PassiveHoming.SwitchedLevel	PositionLimitsSW.MaxPosition
Units.LengthUnit	Homing.AutoReversal
Mechanics.LeadScrew	Homing.ApproachDirection
DynamicLimits.MinVelocity	Homing.ApproachVelocity
DynamicLimits.MaxVelocity	Homing.ReferencingVelocity

Table 10- 78 Parameters for the MC_WriteParam instruction

Parameter and type		Data type	Description
PARAMNAME	IN	Variant	Name of parameter where value is written
VALUE	IN	Variant	Value to write to assigned parameter
EXECUTE	IN	Bool	Start the instruction. Default value: FALSE
DONE	OUT	Bool	Value has been written. Default value: FALSE
BUSY	OUT	Bool	If TRUE, the instruction is operating. Default value: FALSE
ERROR	OUT	Real	If TRUE, an error occurred. Default value: FALSE
ERRORID	OUT	Word	ID of the error
ERRORINFO	OUT	Word	Related information to the ERRORID

Table 10- 79 Condition codes for ERRORID and ERRORINFO

ERRORID (W#16#...)	ERRORINFO (W#16#...)	Description
0	0	Successful change of an Axis TO-DB parameter
8410[1]	0028[1]	Set an invalid parameter (Axis TO-DB parameter with incorrect length)
8410[1]	0029[1]	Set an invalid parameter (no Axis TO-DB parameter)
8410[1]	002B[1]	Set an Invalid parameter (read-only Axis TO-DB parameter)
8410[1]	002C[1]	Set a valid parameter, but axis is not disabled
Config Error[2]	Config Error[2]	Set a valid parameter (public read-only Axis TO-DB parameter) out-of-range
Config Error[3]	Config Error[3]	Set a valid parameter (public Axis TO-DB parameter) out-of-range

[1] Error at MC_WriteParam

[2] Error at MC_Power

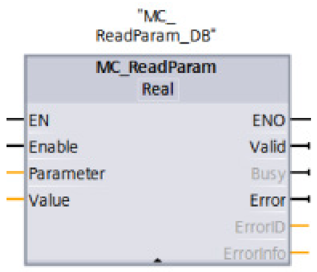
[3] Error at MC_Power and MC_MoveXXX or MC_CommandTable

10.3.7.13 MC_ReadParam instruction (read parameters of a technology object) instruction

You use the MC_ReadParam instruction to read a select number of parameters that indicate the current position, velocity, and so forth of the axis defined in the Axis input.

Table 10- 80 MC_ReadParam instruction

LAD / FBD	SCL	Description

 "MC_ReadParam_DB"
MC_ReadParam
Real
EN
Enable
Parameter
Value
ENO
Valid
Busy
Error
ErrorID
ErrorInfo
	`"MC_ReadParam_DB"(`
	`    Enable:=_bool_in_,`
	`    Parameter:=_variant_in_,`
	`    Value:=_variant_in_out_,`
	`    Valid:=_bool_out_,`
	`    Busy:=_bool_out_,`
	`    Error:=_real_out_,`
	`    ErrorID:=_word_out_,`
	`    ErrorInfo:=_word_out_);`
		You use the MC_ReadParam instruction to read single status values, independent of the cycle control point.

[1] STEP 7 automatically creates the DB when you insert the instruction.

[2] In the SCL example, "MC_ReadParam_DB " is the name of the instance DB.

The MC_ReadParam instruction works on an enable behavior. As long as the input "Enable" is true the instruction reads the specified "Parameter" to the "Value" storage location.

The "MotionStatus" "Position" value updates at each Cycle Control Point (CCP) based upon the current HSC value.

The "MotionStatus" "Velocity" value is the command velocity at the end of the current segment (updated ~10ms). The MC_ReadParam can also read this value.

If an error occurs, the instruction switches to an error state that can only be reset by a new rising edge at the input "Enable".

Table 10- 81 Parameters for the MC_ReadParam instruction

Parameter and type		Data type	Description
ENABLE	IN	Bool	Start the instruction. Default value: FALSE
PARAMETER	IN	Variant	Pointer to the TO-parameter that is to be read
VALID	OUT	Bool	If TRUE, the value has been read. Default value: FALSE
BUSY	OUT	Bool	If TRUE, the instruction is operating. Default value: FALSE
ERROR	OUT	Real	If TRUE, an error occurred. Default value: FALSE
ERRORID	OUT	Word	ID of the error. Default value: 0
ERRORINFO	OUT	Word	Related information to the ERRORID. Default value: 0
VALUE	INOUT	Variant	Pointer to the location where the read value is stored

Table 10- 82 Condition codes for ERRORID and ERRORINFO

ERRORID (W#16#...)	ERRORINFO (W#16#...)	Description
0	0	Successful read of a parameter
8410	0028	Invalid parameter (incorrect length)
8410	0029	Invalid parameter (no TO-DB)
8410	0030	Invalid parameter (not readable)
8411	0032	Invalid parameter (wrong value)

TO parameters

The axis "MotionStatus" consists of four values. You will want to monitor changes in these values, which can be read while the program is running:

Variable name	Data type	Readable through MC_ReadParam
MotionStatus:	Structure	No
• Position	REAL	Yes
• Velocity	REAL	Yes
• Distance	REAL	Yes
• TargetPosition	REAL	Yes

10.3.8 Monitoring active commands

10.3.8.1 Monitoring MC instructions with a "Done" output parameter

Motion control instructions with the output parameter "Done" are started by the input parameter "Execute" and have a defined conclusion (for example, with motion control instruction "MC_Home": Homing was successful). The task is complete and the axis is at a standstill.

- The output parameter "Done" indicates the value TRUE, if the task has been successfully completed.
- The output parameters "Busy", "CommandAborted", and "Error" signal that the task is still being processed, has been aborted or an error is pending. The motion control instruction "MC_Reset" cannot be aborted and thus has no "CommandAborted" output parameter.
  - During processing of the motion control task, the output parameter "Busy" indicates the value TRUE. If the task has been completed, aborted, or stopped by an error, the output parameter "Busy" changes its value to FALSE. This change occurs regardless of the signal at input parameter "Execute".
  - Output parameters "Done", "CommandAborted", and "Error" indicate the value TRUE for at least one cycle. These status messages are latched while input parameter "Execute" is set to TRUE.

The tasks of the following motion control instructions have a defined conclusion:

- MC_Reset
- MC_Home
- MC_Halt
- MC_MoveAbsolute
- MC_MoveRelative

The behavior of the status bits is presented below for various example situations.

- The first example shows the behavior of the axis for a completed task. If the motion control task has been completely executed by the time of its conclusion, this is indicated by the value TRUE in output parameter "Done". The signal status of input parameter "Execute" influences the display duration in the output parameter "Done".
- The second example shows the behavior of the axis for an aborted task. If the motion control task is aborted during execution, this is indicated by the value TRUE in output parameter "CommandAborted". The signal status of the input parameter "Execute" influences the display duration in the output parameter "CommandAborted".
- The third example shows the behavior of the axis if an error occurs. If an error occurs during execution of the motion control task, this is indicated by the value TRUE in the output parameter "Error". The signal status of the input parameter "Execute" influences the display duration in the output parameter "Error".

Table 10- 83 Example 1 - Complete execution of task

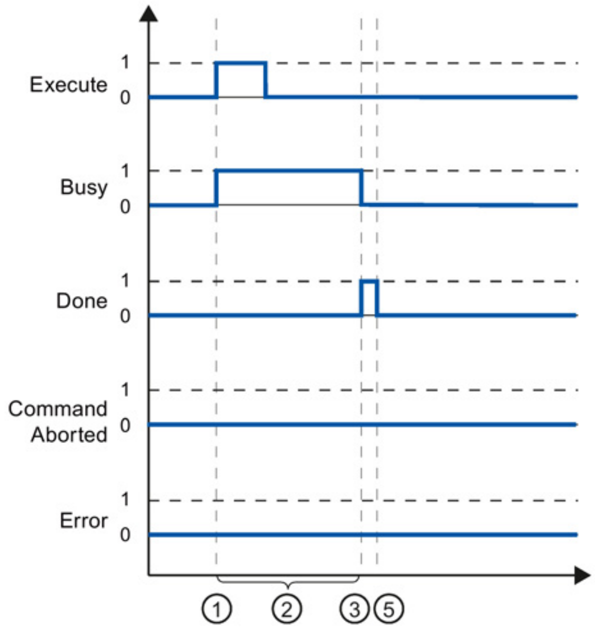
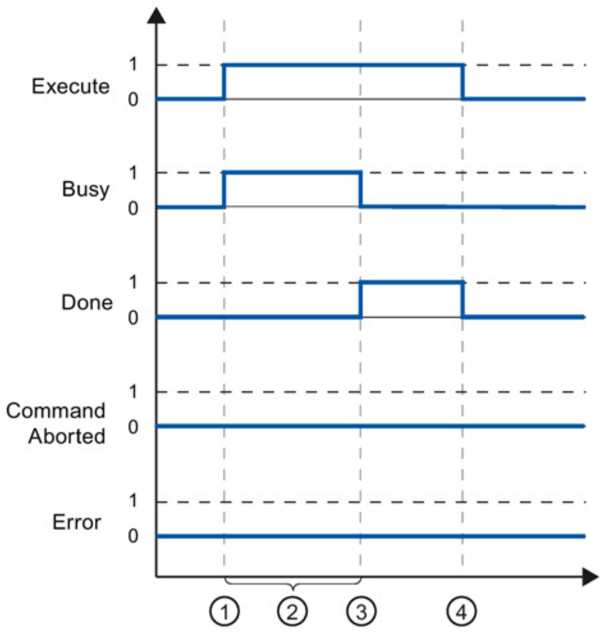
 Execute Busy Done Command Aborted Error 1 0 ① ② ③⑤ If "Execute" = FALSE during the processing of the task	 Execute Busy Done Command Aborted Error 1 0 ① ② ③ ④ If "Execute" = FALSE after completion of the task
① The task is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the task, or the value TRUE can be retained until after completion of the task. ② While the task is active, the output parameter "Busy" indicates the value TRUE. ③ With conclusion of the task (for example, for motion control instruction "MC_Home": Homing was successful), output parameter "Busy" changes to FALSE and "Done" to TRUE. ④ If "Execute" retains the value TRUE until after completion of the task, then "Done" also remains TRUE and changes its value to FALSE together with "Execute". ⑤ If "Execute" has been set to FALSE before the task is complete, "Done" indicates the value TRUE for only one execution cycle.	

Table 10- 84 Example 2 - Aborting the task

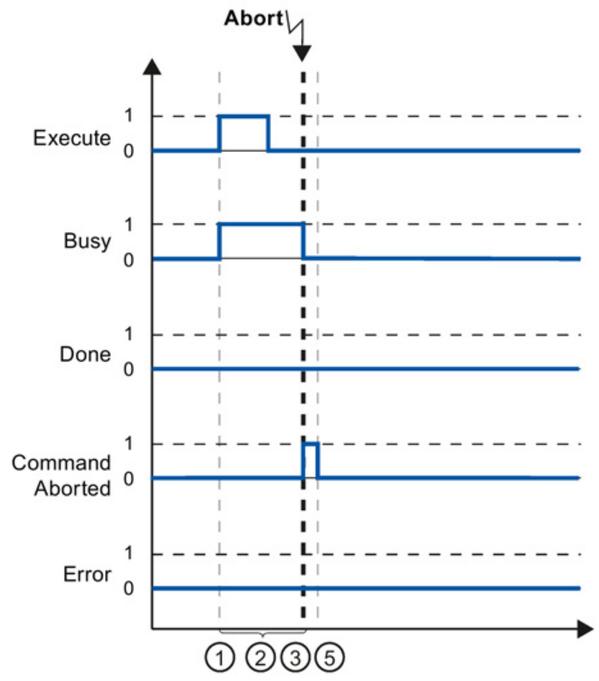
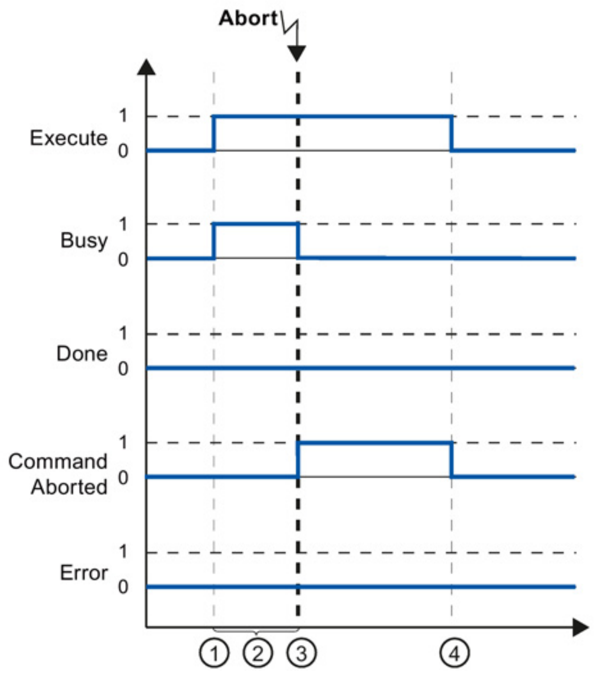
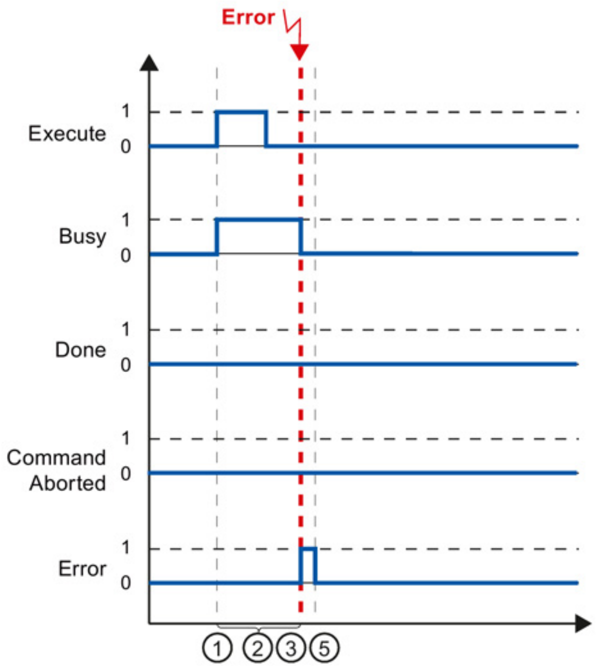
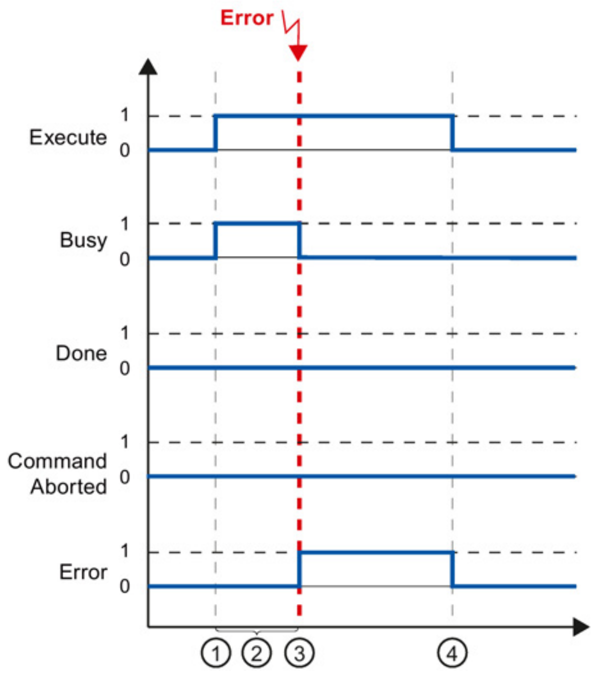
 Abort Execute Busy Done Command Aborted Error 1 0 ① ② ③ ⑤ If "Execute" = FALSE before the task is aborted	 Abort Execute Busy Done Command Aborted Error 1 0 ① ② ③ ④ If "Execute" = FALSE after the task is aborted
① The task is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the task, or the value TRUE can be retained until after completion of the task. ② While the task is active, the output parameter "Busy" indicates the value TRUE. ③ During task execution, the task is aborted by another motion control task. If the task is aborted, output parameter "Busy" changes to FALSE and "CommandAborted" to TRUE. ④ If "Execute" retains the value TRUE until after the task is aborted, then "CommandAborted" also remains TRUE and changes its value to FALSE together with "Execute". ⑤ If "Execute" has been set to FALSE before the task is aborted, "CommandAborted" indicates the value TRUE for only one execution cycle.	

Table 10- 85 Example 3 - Error during task execution

 Error Execute Busy Done Command Aborted Error 1 0 ①②③⑤ If "Execute" = FALSE before the error occurs	 Error Execute Busy Done Command Aborted Error 1 0 ①②③④ If "Execute" = FALSE after the error occurs
① The task is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the task, or the value TRUE can be retained until after completion of the task. ② While the task is active, the output parameter "Busy" indicates the value TRUE. ③ An error occurred during task execution. When the error occurs, the output parameter "Busy" changes to FALSE and "Error" to TRUE. ④ If "Execute" retains the value TRUE until after the error occurs, then "Error" also remains TRUE and only changes its value to FALSE together with "Execute". ⑤ If "Execute" has been set to FALSE before the error occurs, "Error" indicates the value TRUE for only one execution cycle.	

10.3.8.2 Monitoring the MC_Velocity instruction

The tasks of the motion control instruction "MC_MoveVelocity" implement a move at the specified velocity:

- The tasks of motion control instruction "MC_MoveVelocity" do not have a defined end. The task objective is fulfilled when the parameterized velocity is reached for the first time and the axis travels at constant velocity. When the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "InVelocity".
- The task is complete when the parameterized velocity has been reached and input parameter "Execute" has been set to the value FALSE. However, the axis motion is not yet complete upon completion of the task. For example, the axis motion can be stopped with motion control task "MC_Halt".
- The output parameters "Busy", "CommandAborted", and "Error" signal that the task is still being processed, has been aborted or an error is pending.
  - During execution of the motion control task, output parameter "Busy" indicates the value TRUE. If the task has been completed, aborted, or stopped by an error, the output parameter "Busy" changes its value to FALSE. This change occurs regardless of the signal at input parameter "Execute".
  - The output parameters "InVelocity", "CommandAborted", and "Error" indicate the value TRUE for at least one cycle, when their conditions are met. These status messages are latched while input parameter "Execute" is set to TRUE.

The behavior of the status bits is presented below for various example situations:

- The first example shows the behavior when the axis reaches the parameterized velocity. If the motion control task has been executed by the time the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "InVelocity". The signal status of the input parameter "Execute" influences the display duration in the output parameter "InVelocity".
- The second example shows the behavior if the task is aborted before achieving the parameterized velocity. If the motion control task is aborted before the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "CommandAborted". The signal status of input parameter "Execute" influences the display duration in output parameter "CommandAborted".
- The third example shows the behavior of the axis if an error occurs before achieving the parameterized velocity. If an error occurs during execution of the motion control task before the parameterized velocity has been reached, this is indicated by the value TRUE in the output parameter "Error". The signal status of the input parameter "Execute" influences the display duration in the output parameter "Error".

Table 10- 86 Example 1 - If the parameterized velocity is reached

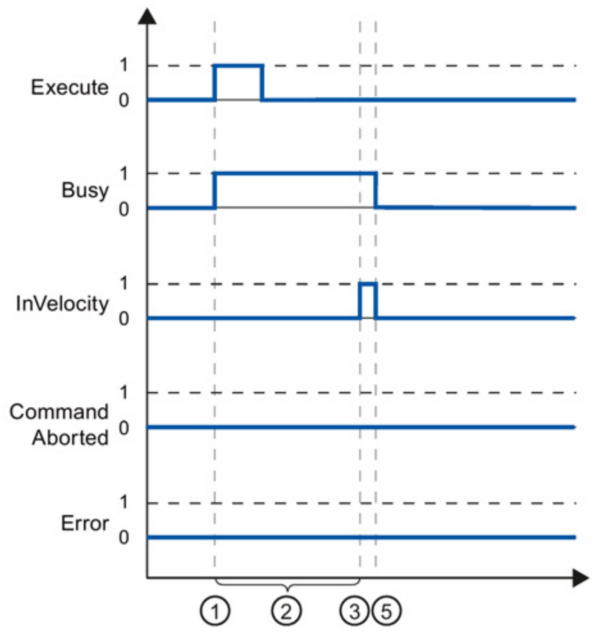
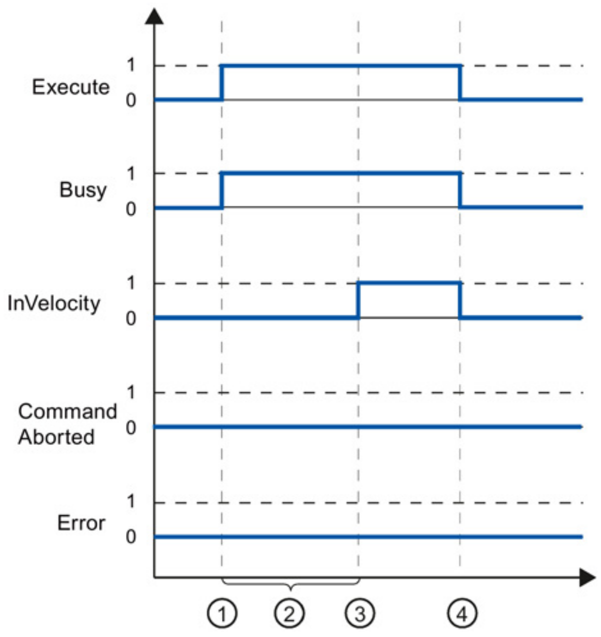
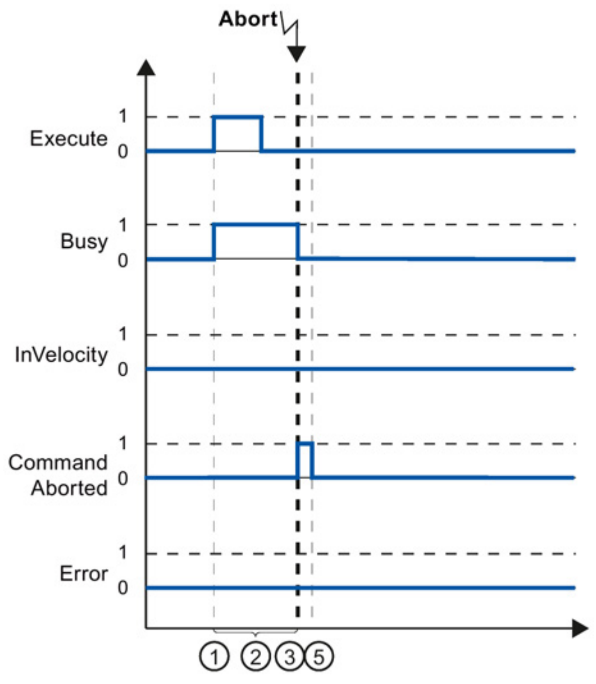
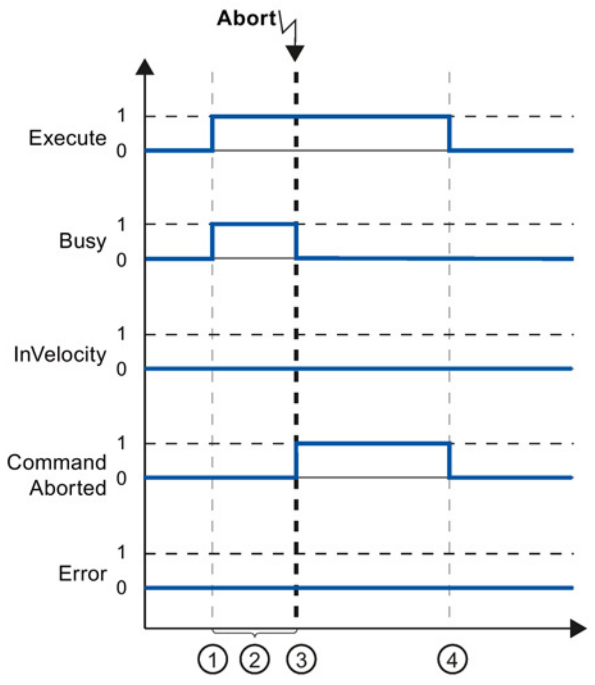
 If "Execute" = FALSE before the configured velocity is reached	 If "Execute" = FALSE after the configured velocity is reached
① The task is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can be reset to the value FALSE event before the parameterized velocity is reached, or alternatively only after it has been reached. ② While the task is active, the output parameter "Busy" indicates the value TRUE. ③ When the parameterized velocity is reached, the output parameter "InVelocity" changes to TRUE. ④ If "Execute" retains the value TRUE even after the parameterized velocity has been reached, the task remains active. "InVelocity" and "Busy" retain the value TRUE and only change their status to FALSE together with "Execute". ⑤ If "Execute" has been reset to FALSE before the parameterized velocity is reached, the task is complete when the parameterized velocity is reached. "InVelocity" indicates the value TRUE for one execution cycle and changes to FALSE together with "Busy".	

Table 10- 87 Example 2 - If the task is aborted prior to reaching the parameterized velocity

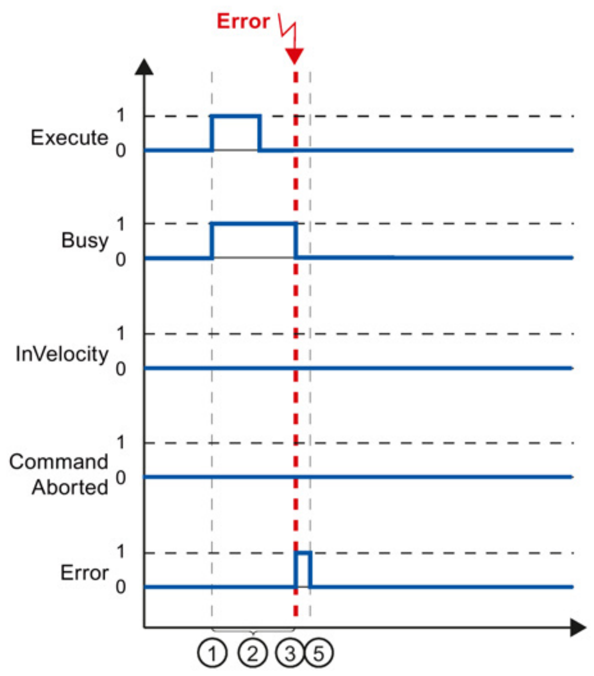
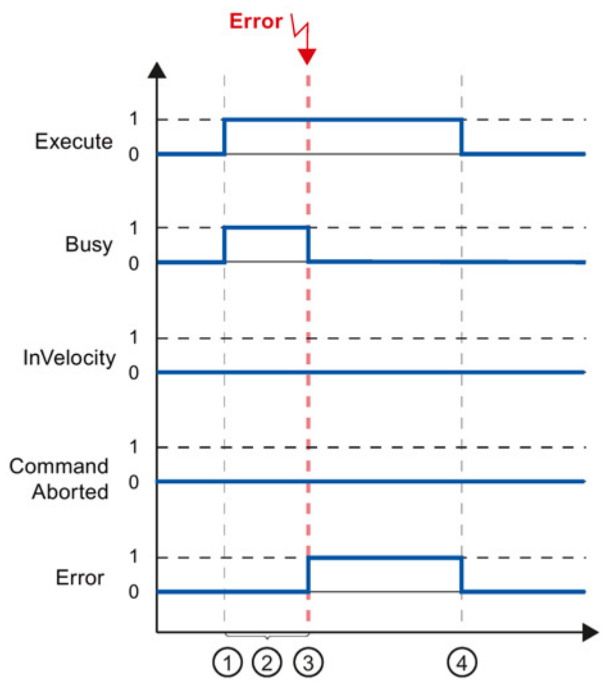
 Abort Execute Busy InVelocity Command Aborted Error 1 0 ① ②③⑤ If "Execute" = FALSE before the task is aborted	 Abort Execute Busy InVelocity Command Aborted Error 1 0 ① ② ③ ④ If "Execute" = FALSE after the task is aborted
① The task is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the task, or the value TRUE can be retained until after the task is aborted. ② While the task is active, the output parameter "Busy" indicates the value TRUE. ③ During task execution, the task is aborted by another motion control task. If the task is aborted, output parameter "Busy" changes to FALSE and "CommandAborted" to TRUE. ④ If "Execute" retains the value TRUE until after the task is aborted, then "CommandAborted" also remains TRUE and changes its status to FALSE together with "Execute". ⑤ If "Execute" has been reset to FALSE before the task is aborted, "CommandAborted" indicates the value TRUE for only one execution cycle.	

Note

Under the following conditions, an abort is not indicated in output parameter "CommandAborted":

- The parameterized velocity has been reached, input parameter "Execute" has the value FALSE, and a new motion control task is initiated.
- When the parameterized velocity is reached and input parameter "Execute" has the value FALSE, the task is complete. Therefore, the start of a new task is not indicated as an abort.

Table 10- 88 Example 3 - If an error occurs prior to reaching the parameterized velocity

 Error Execute Busy InVelocity Command Aborted Error 1 0 ① ② ③ ⑤ If "Execute" = FALSE before the error occurs	 Error Execute Busy InVelocity Command Aborted Error 1 0 ① ② ③ ④ If "Execute" = FALSE after the error occurs
① The task is started with a positive edge at the input parameter "Execute". Depending on the programming, "Execute" can still be reset to the value FALSE during the task, or the value TRUE can be retained until after the error has occurred. ② While the task is active, the output parameter "Busy" indicates the value TRUE. ③ An error occurred during task execution. When the error occurs, the output parameter "Busy" changes to FALSE and "Error" to TRUE. ④ If "Execute" retains the value TRUE until after the error has occurred, then "Error" also remains TRUE and only changes its status to FALSE together with "Execute". ⑤ If "Execute" has been reset to FALSE before the error occurs, "Error" indicates the value TRUE for only one execution cycle.	

Note

Under the following conditions, an error is not indicated in output parameter "Error":

- The parameterized velocity has been reached, input parameter "Execute" has the value FALSE, and an axis error occurs (software limit switch is approached, for example).
- When the parameterized velocity is reached and input parameter "Execute" has the value FALSE, the task is complete. After completion of the task, the axis error is only indicated in the motion control instruction "MC_Power".

10.3.8.3 Monitoring the MC_MoveJog instruction

The tasks of motion control instruction "MC_MoveJog" implement a jog operation.

- The motion control tasks "MC_MoveJog" do not have a defined end. The task objective is fulfilled when the parameterized velocity is reached for the first time and the axis travels at constant velocity. When the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "InVelocity".
- The order is complete when input parameter "JogForward" or "JogBackward" has been set to the value FALSE and the axis has come to a standstill.
- The output parameters "Busy", "CommandAborted", and "Error" signal that the task is still being processed, has been aborted or an error is pending.
  - During processing of the motion control task, the output parameter "Busy" indicates the value TRUE. If the task has been completed, aborted, or stopped by an error, the output parameter "Busy" changes its value to FALSE.
  - The output parameter "InVelocity" indicates the status TRUE, as long as the axis is moving at the parameterized velocity. The output parameters "CommandAborted" and "Error" indicate the status for at least one cycle. These status messages are latched as long as either input parameter "JogForward" or "JogBackward" is set to TRUE.

The behavior of the status bits is presented below for various example situations.

- The first example shows the behavior or the axis if the parameterized velocity is reached and maintained. If the motion control task has been executed by the time the parameterized velocity is reached, this is indicated by the value TRUE in output parameter "InVelocity".
- The second example shows the behavior of the axis if the task is aborted. If the motion control task is aborted during execution, this is indicated by the value TRUE in output parameter "CommandAborted". The behavior is independent of whether or not the parameterized velocity has been reached.
- The third example shows the behavior of the axis if an error occurs. If an error occurs during execution of the motion control task, this is indicated by the value TRUE in output parameter "Error". The behavior is independent of whether or not the parameterized velocity has been reached.

Table 10- 89 Example 1 - If the parameterized velocity is reached and maintained

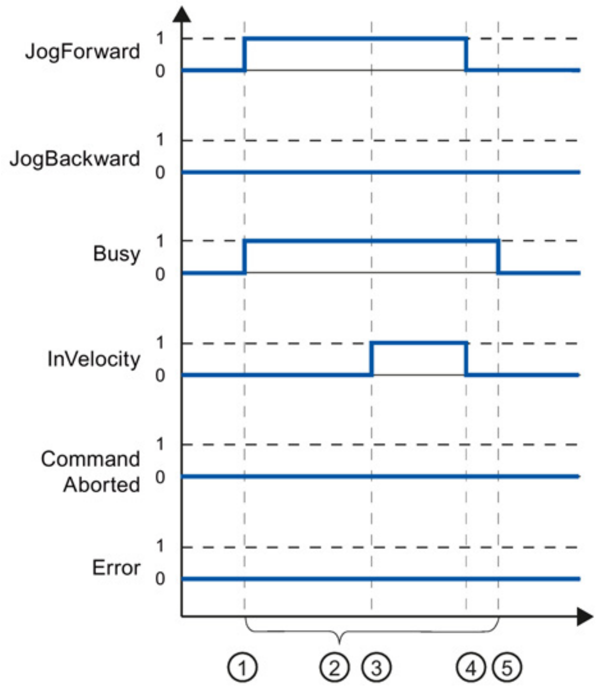
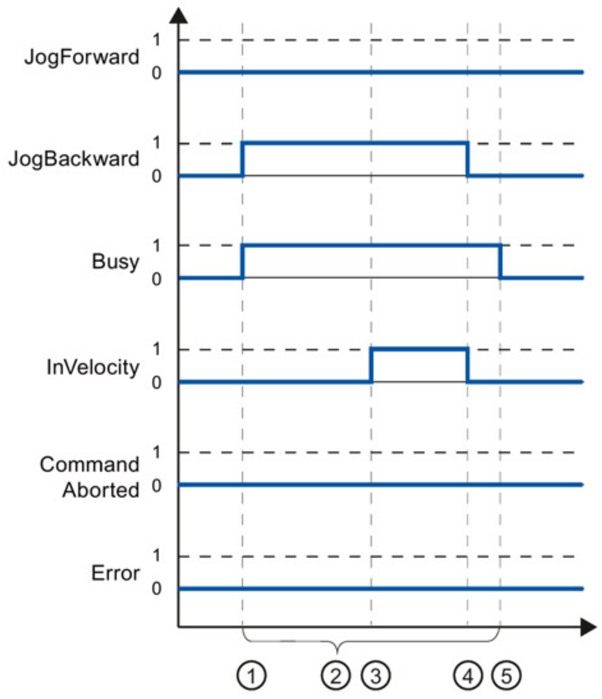
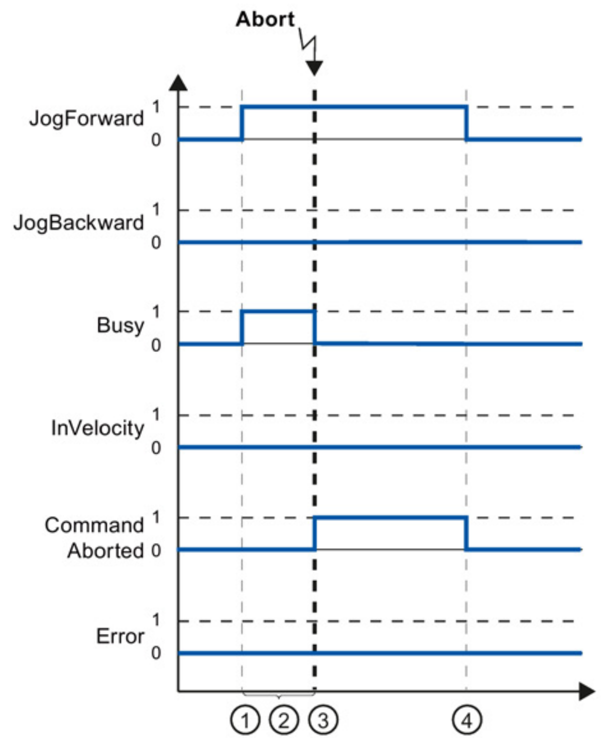
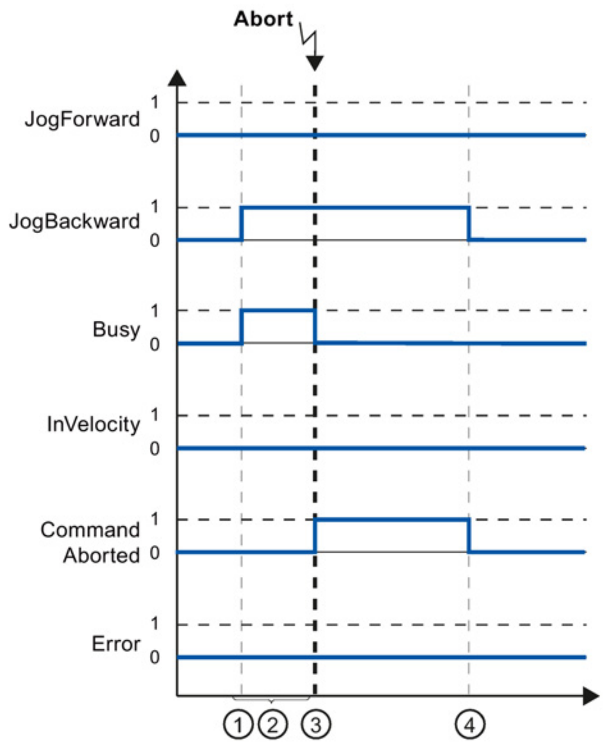
 JogForward JogBackward Busy InVelocity Command Aborted Error 1 0 ① ②③ ④⑤ JogForward	 JogForward JogBackward Busy InVelocity Command Aborted Error 1 0 ① ②③ ④⑤ JogBackward
① The task is started with a positive edge at the input parameter "JogForward" or "JogBackward". ② While the task is active, the output parameter "Busy" indicates the value TRUE. ③ When the parameterized velocity is reached, the output parameter "InVelocity" changes to TRUE. ④ When the input parameter "JogForward" or "JogBackward" is reset to the value FALSE, the axis motion ends. The axis starts to decelerate. As a result, the axis no longer moves at constant velocity and the output parameter "InVelocity" changes its status to FALSE. ⑤ If the axis has come to a standstill, the motion control task is complete and the output parameter "Busy" changes its value to FALSE.	

Table 10- 90 Example 2 - If the task is aborted during execution

 Abort JogForward 1 0 JogBackward 1 0 Busy 1 0 InVelocity 1 0 Command Aborted 1 0 Error 1 0 ① ② ③ ④ JogForward	 Abort JogForward 1 0 JogBackward 1 0 Busy 1 0 InVelocity 1 0 Command Aborted 1 0 Error 1 0 ① ② ③ ④ JogBackward

① The task is started with a positive edge at the input parameter "JogForward" or "JogBackward".

② While the task is active, the output parameter "Busy" indicates the value TRUE.

③ During task execution, the task is aborted by another motion control task. If the task is aborted, output parameter "Busy" changes to FALSE and "CommandAborted" to TRUE.

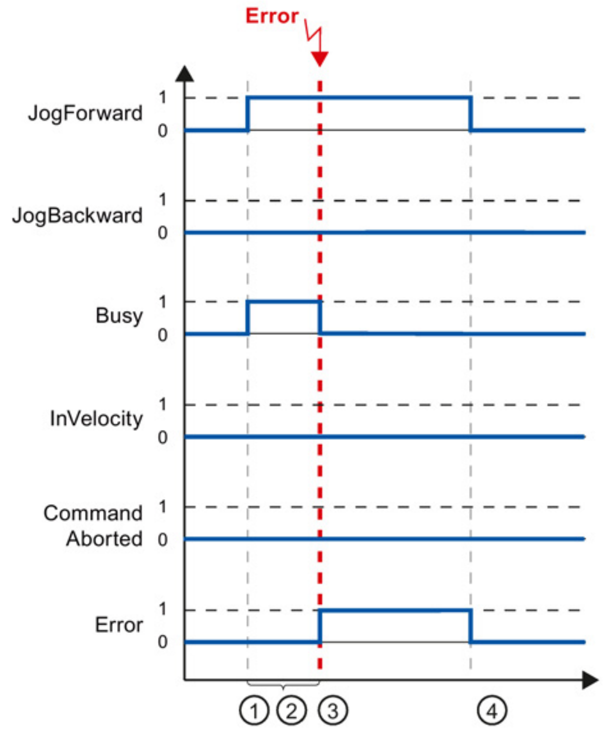
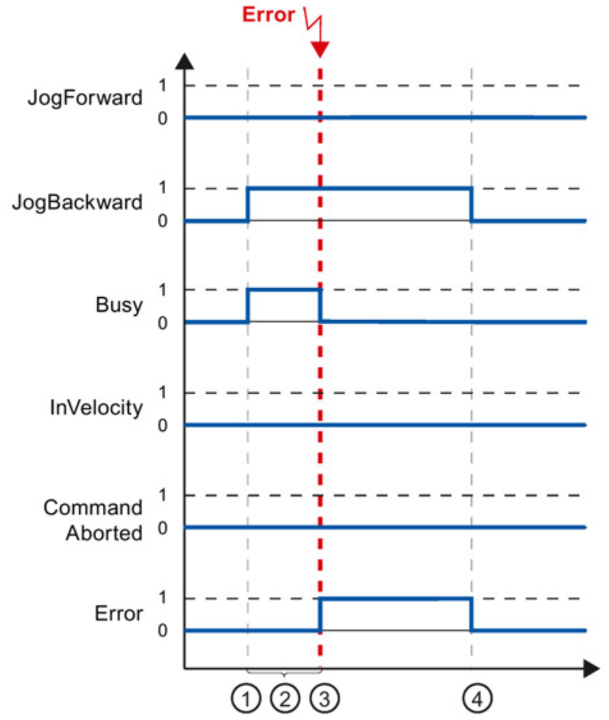
④ When the input parameter "JogForward" or "JogBackward" is reset to the value FALSE, the output parameter "CommandAborted" changes its value to FALSE.

Note

The task abort is indicated in the output parameter "CommandAborted" for only one execution cycle, if all conditions below are met:

The input parameters "JogForward" and "JogBackward" have the value FALSE (but the axis is still decelerating) and a new motion control task is initiated.

Table 10- 91 Example 3 - If an error has occurred during task execution

 Error JogForward 1 0 JogBackward 1 0 Busy 1 0 InVelocity 1 0 Command Aborted 1 0 Error 1 0 ① ② ③ ④ JogForward	 Error JogForward 1 0 JogBackward 1 0 Busy 1 0 InVelocity 1 0 Command Aborted 1 0 Error 1 0 ① ② ③ ④ JogBackward
① The task is started with a positive edge at the input parameter "JogForward" or "JogBackward".	
② While the task is active, the output parameter "Busy" indicates the value TRUE.	
③ An error occurred during task execution. When the error occurs, the output parameter "Busy" changes to FALSE and "Error" to TRUE.	
④ When the input parameter "JogForward" or "JogBackward" is reset to the value FALSE, the output parameter "Error" changes its value to FALSE.	

Note

An error occurrence is indicated in the output parameter "Error" for only one execution cycle, if all the conditions below are met:

The input parameters "JogForward" and "JogBackward" have the value FALSE (but the axis is still decelerating) and a new error occurs (software limit switch is approached, for example).

Communication 11

The S7-1200 offers several types of communication between CPUs and programming devices, HMIs, and other CPUs.

⚠WARNING
If an attacker can physically access your networks, the attacker can possibly read and write data. The TIA Portal, the CPU, and HMIs (except HMIs using GET/PUT) use secure communication that protects against replay and "man-in-the-middle" attacks. Once communication is enabled, the exchange of signed messages takes place in clear text which allows an attacker to read data, but protects against unauthorized writing of data. The TIA Portal, not the communication process, encrypts the data of know-how protected blocks. All other forms of communication (I/O exchange through PROFIBUS, PROFINET, AS-i, or other I/O bus, GET/PUT, T-Block, and communication modules (CM)) have no security features. You must protect these forms of communication by limiting physical access. If an attacker can physically access your networks utilizing these forms of communication, the attacker can possibly read and write data. For security information and recommendations, please see our "Operational Guidelines for Industrial Security" (http://www.industry.siemens.com/topics/global/en/industrial-security/Documents/operational_guidelines_industrial_security_en.pdf) on the Siemens Service and Support site.

PROFINET

PROFINET is used for exchanging data through the user program with other communications partners through Ethernet:

- In the S7-1200, PROFINET supports 16 IO devices with a maximum of 256 submodules, and PROFIBUS allows 3 independent PROFIBUS DP Masters, supporting 32 slaves per DP master, with a maximum of 512 modules per DP master.
- S7 communication
- User Datagram Protocol (UDP) protocol
- ISO on TCP (RFC 1006)
- Transport Control Protocol (TCP)

PROFINET IO controller

As an IO controller using PROFINET IO, the CPU communicates with up to 16 PN devices on the local PN network or through a PN/PN coupler (link). Refer to PROFIBUS and PROFINET International, PI (www.us.profinet.com) for more information.

PROFIBUS

PROFIBUS is used for exchanging data through the user program with other communications partners through the PROFIBUS network:

- With CM 1242-5, the CPU operates as a PROFIBUS DP slave.
- With CM 1243-5, the CPU operates as a PROFIBUS DP master class1.
- PROFIBUS DP Slaves, PROFIBUS DP Masters, and AS-i (the 3 left-side communication modules) and PROFINET are separate communications networks that do not limit each other.

AS-i

The S7-1200 CM 1243-2 AS-i Master allows the attachment of an AS-i network to an S7-1200 CPU.

CPU-to-CPU S7 communication

You can create a communication connection to a partner station and use the GET and PUT instructions to communicate with S7 CPUs.

TeleService communication

In TeleService via GPRS, an engineering station on which STEP 7 is installed communicates via the GSM network and the Internet with a SIMATIC S7-1200 station with a CP 1242-7. The connection runs via a telecontrol server that serves as an intermediary and is connected to the Internet.

IO-Link

The S7-1200 SM 1278 4xIO-Link Master enables IO-Link devices to connect to an S7-1200 CPU.

11.1 V4.1 asynchronous communication connections

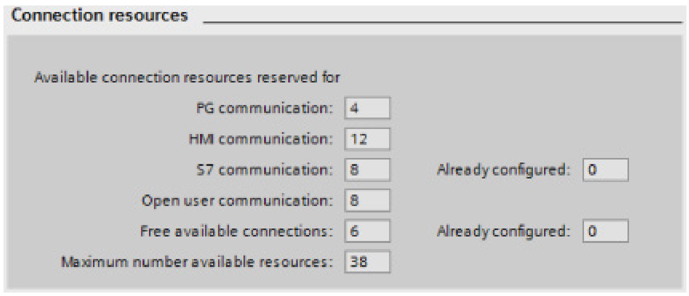
Overview of communication services

The CPU supports the following communication services:

Communication service	Functionality	Using PROFIBUS DP		Using Ethernet
		CM 1243-5 DP master module	CM 1242-5 DP slave module	
PG communication	Commissioning, testing, diagnostics	Yes	No	Yes
HMI communication	Operator control and monitoring	Yes	No	Yes
S7 communication	Data exchange using configured connections	Yes	No	Yes
Routing of PG functions	For example, testing and diagnostics beyond network boundaries	No	No	No
PROFIBUS DP	Data exchange between master and slave	Yes	Yes	No
PROFINET IO	Data exchange between I/O controllers and I/O devices	No	No	Yes
Web server	Diagnostics	No	No	Yes
SNMP (Simple Network Management Protocol)	Standard protocol for network diagnostics and parameterization	No	No	Yes
Open communication over TCP/IP	Data exchange over Industrial Ethernet with TCP/IP protocol (with loadable FBs)	No	No	Yes
Open communication over ISO on TCP	Data exchange over Industrial Ethernet with ISO on TCP protocol (with loadable FBs)	No	No	Yes
Open communication over UDP	Data exchange over Industrial Ethernet with UDP protocol (with loadable FBs)	No	No	Yes

Available connections

The CPU supports the following number of maximum simultaneous, asynchronous communication connections for PROFINET and PROFIBUS. The maximum number of connection resources allocated to each category are fixed; you cannot change these values. However, you can configure the 6 "Free available connections" to increase the number of any category as required by your application.


Connection resources

Available connection resources reserved for

PG communication:	4		
HMI communication:	12		
S7 communication:	8	Already configured:	0
Open user communication:	8		
Free available connections:	6	Already configured:	0
Maximum number available resources:	38		

Based upon the allocated connection resources, the following number of connections per device are available:

	Programming terminal (PG)	Human Machine Interface (HMI)	GET/PUT client/server	Open User Communications	Web browser
Maximum number of connection resources	3 (guaranteed to support 1 PG device)	12 (guaranteed to support 4 HMI devices)	8	8	30 (guaranteed to support 3 web browsers)

For an example, a PG has 3 available connection resources. Depending on the current PG functions in use, the PG might actually use 1, 2, or 3 of its available connection resources. In the S7-1200, you are always guaranteed at least 1 PG; however, no more than 1 PG is allowed.

Another example is the number of HMIs, as shown in the figure below. HMIs have 12 available connection resources. Depending on what HMI type or model that you have and the HMI functions that you use, each HMI might actually use 1, 2, or 3 of its available connection resources. Given the number of available connection resources being used, it may be possible to use more than 4 HMIs at one time. However, you are always guaranteed at least 4 HMIs. An HMI can use its available connection resources (1 each for a total of 3) for the following functions:

- Reading
- Writing
- Alarming plus diagnostics

Example	HMI 1	HMI 2	HMI 3	HMI 4	HMI 5	Total con-nection resources available
Connection resources used	2	2	2	3	3	12

Note

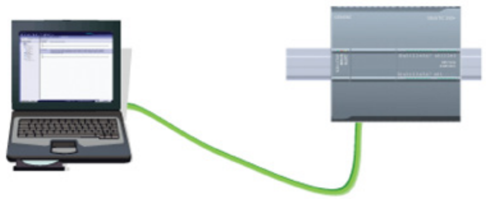
Web server (HTTP) connections: The CPU provides connections for multiple web browsers. The number of browsers that the CPU can simultaneously support depends upon how many connections a given web browser requests/utilizes.

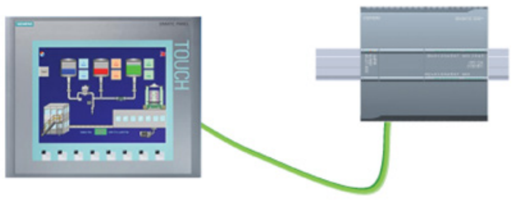
Note

The Open User Communications, S7 connection, HMI, programming device, and Web server (HTTP) communication connections may utilize multiple connection resources based upon the features currently being used.

11.2 PROFINET

The CPU can communicate with other CPUs, with programming devices, with HMI devices, and with non-Siemens devices using standard TCP communications protocols.

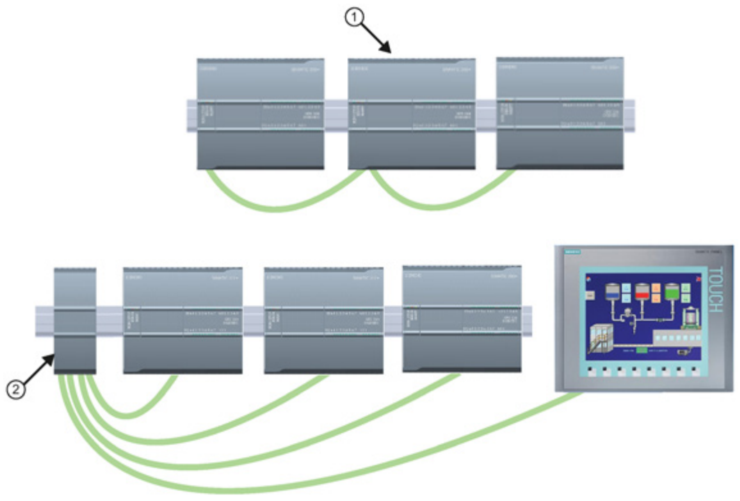

Programming device connected to the CPU


HMI connected to the CPU

A CPU connected to another CPU

Ethernet switching

The PROFINET port on the CPU 1211C, 1212C, and 1214C does not contain an Ethernet switching device. A direct connection between a programming device or HMI and a CPU does not require an Ethernet switch. However, a network with more than two CPUs or HMI devices requires an Ethernet switch.



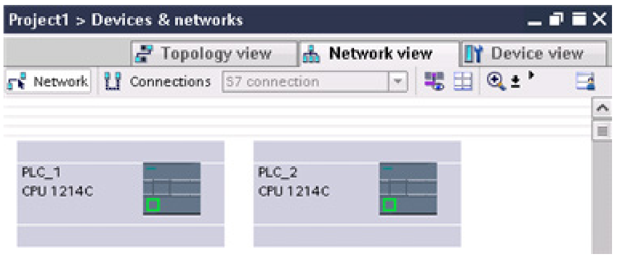
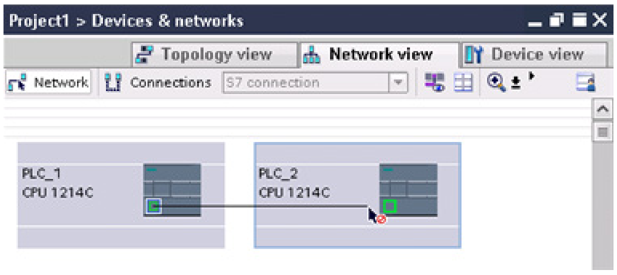
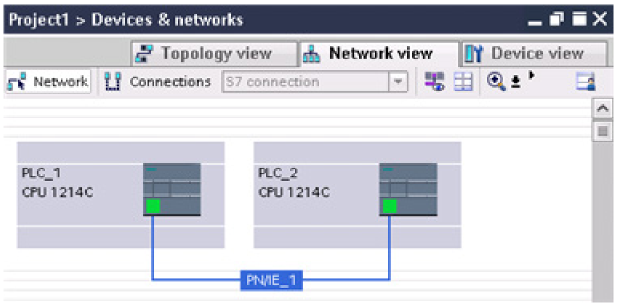
①	CPU 1215C
②	CSM1277 Ethernet switch

The CPU 1215C and the CPU 1217C have a built-in 2-port Ethernet switch. You can have a network with a CPU 1215C and two other S7-1200 CPUs. You can also use the rack-mounted CSM1277 4-port Ethernet switch for connecting multiple CPUs and HMI devices.

11.2.1 Creating a network connection

Use the "Network view" of Device configuration to create the network connections between the devices in your project. After creating the network connection, use the "Properties" tab of the inspector window to configure the parameters of the network.

Table 11- 1 Creating a network connection

Action	Result
Select "Network view" to display the devices to be connected.	 Project1 > Devices & networks Topology view Network view Device view Network Connections S7 connection PLC_1 CPU 1214C PLC_2 CPU 1214C
Select the port on one device and drag the connection to the port on the second device.	 Project1 > Devices & networks Topology view Network view Device view Network Connections S7 connection PLC_1 CPU 1214C PLC_2 CPU 1214C
Release the mouse button to create the network connection.	 Project1 > Devices & networks Topology view Network view Device view Network Connections S7 connection PLC_1 CPU 1214C PLC_2 CPU 1214C PN/IE_1

11.2.2 Configuring the Local/Partner connection path

A Local / Partner (remote) connection defines a logical assignment of two communication partners to establish communication services. A connection defines the following:

- Communication partners involved (One active, one passive)
- Type of connection (for example, a PLC, HMI, or device connection)
- Connection path

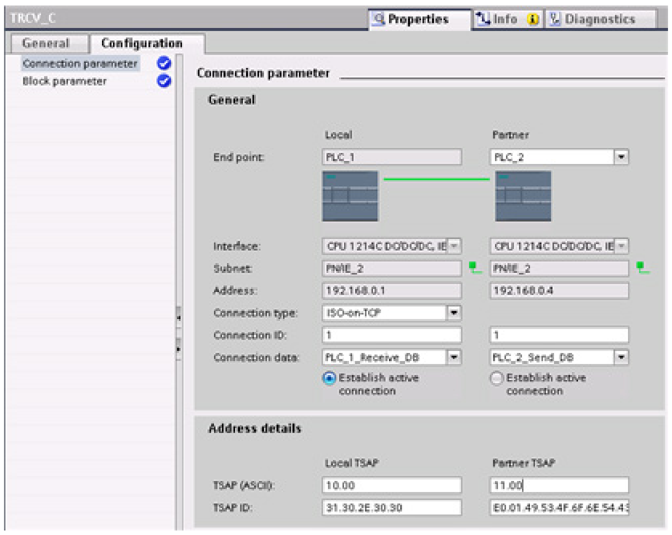
Communication partners execute the instructions to set up and establish the communication connection. You use parameters to specify the active and passive communication end point partners. After the connection is set up and established, it is automatically maintained and monitored by the CPU.

If the connection is terminated (for example, due to a line break), the active partner attempts to re-establish the configured connection. You do not have to execute the communication instruction again.

Connection paths

After inserting a TSEND_C, TRCV_C or TCON instruction into the user program, the inspector window displays the properties of the connection whenever you have selected any part of the instruction. Specify the communication parameters in the "Configuration" tab of the "Properties" for the communication instruction.

Table 11- 2 Configuring the connection path (using the properties of the instruction)

TCP, ISO-on-TCP, and UDP	Connection properties
For the TCP, ISO-on-TCP, and UDP Ethernet protocols, use the "Properties" of the instruction (TSEND_C, TRCV_C, or TCON) to configure the "Local/Partner" connections. The illustration shows the "Connection properties" of the "Configuration tab" for an ISO-on-TCP connection.	

	Local	Partner
End point:	PLC_1	PLC_2
Interface:	CPU 1214C DC/DC/DC, IE	CPU 1214C DC/DC/DC, IE
Subnet:	PN/IE_2	PN/IE_2
Address:	192.168.0.1	192.168.0.4
Connection type:	ISO-on-TCP	
Connection ID:	1	1
Connection data:	PLC_1_Receive_DB	PLC_2_Send_DB
	[✓ Establish active connection]	Establish active connection

Address details

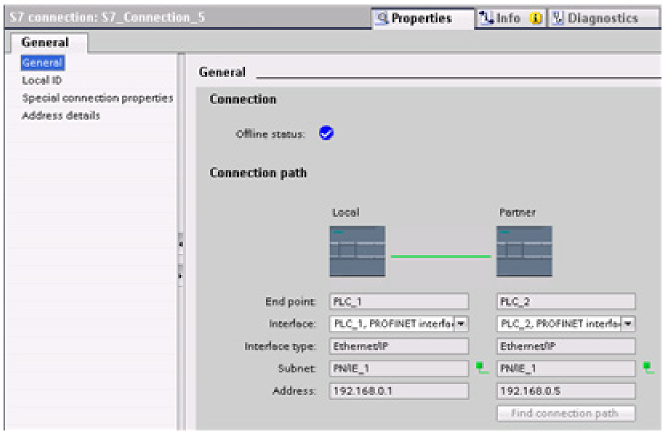
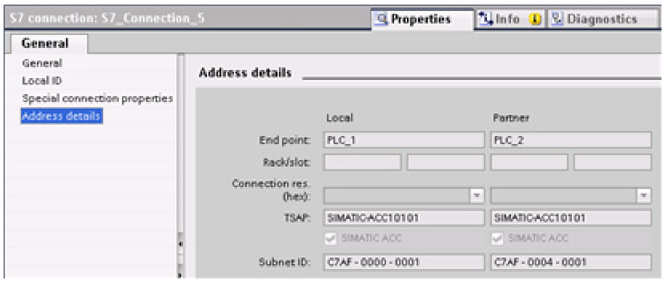
	Local TSAP	Partner TSAP
TSAP (ASCII):	10.00	11.00
TSAP ID:	31.30.2E.30.30	E0.01.49.53.4F.6F.6E.54.4

Note

When you configure the connection properties for one CPU, STEP 7 allows you either to select a specific connection DB in the partner CPU (if one exists), or to create the connection DB for the partner CPU. The partner CPU must already have been created for the project and cannot be an "unspecified" CPU.

You must still insert a TSEND_C, TRCV_C or TCON instruction into the user program of the partner CPU. When you insert the instruction, select the connection DB that was created by the configuration.

Table 11- 3 Configuring the connection path for S7 communication (Device configuration)

S7 communication (GET and PUT)	Connection properties
For S7 communication, use the "Devices & net-works" editor of the network to configure the Local/Partner connections. You can click the "Highlighted: Connection" button to access the "Properties". The "General" tab provides several properties: • "General" (shown) • "Local ID" • "Special connection properties" • "Address details" (shown)	 S7 connection: S7_Connection_5 General Connection Offline status: ✓ Connection path Local / Partner End point: PLC_1 / PLC_2 Interface: PLC_1, PROFINET interfa / PLC_2, PROFINET interfa Interface type: Ethernet/IP / Ethernet/IP Subnet: PN/IE_1 / PN/IE_1 Address: 192.168.0.1 / 192.168.0.5 Properties Info Diagnostics General General Local ID Special connection properties Address details Find connection path
	 S7 connection: S7_Connection_5 Address details Local / Partner End point: PLC_1 / PLC_2 Rack/slot: Connection res. (hex): TSAP: SIMATIC-ACC10101 / SIMATIC-ACC10101 ✓ SIMATIC ACC / ✓ SIMATIC ACC Subnet ID: C7AF - 0000 - 0001 / C7AF - 0004 - 0001 Properties Info Diagnostics General General Local ID Special connection properties Address details

Refer to "Protocols" (Page 634) in the "PROFINET" section or to "Creating an S7 connection" (Page 774) in the "S7 communication" section for more information and a list of available communication instructions.

Table 11- 4 Parameters for the multiple CPU connection

Parameter		Definition
Address		Assigned IP addresses
General	End point	Name assigned to the partner (receiving) CPU
	Interface	Name assigned to the interfaces
	Subnet	Name assigned to the subnets
	Interface type	*S7 communication only*: Type of interface
	Connection type	Type of Ethernet protocol
	Connection ID	ID number
	Connection data	Local and Partner CPU data storage location
	Establish active connection	Radio button to select Local or Partner CPU as the active connection
Address details	End point	*S7 communication only*: Name assigned to the partner (receiving) CPU
	Rack/slot	*S7 communication only*: Rack and slot location
	Connection resource	*S7 communication only*: Component of the TSAP used when configuring an S7 connection with an S7-300 or S7-400 CPU
	Port (decimal):	TCP and UPD: Partner CPU port in decimal format
	TSAP [1] and Subnet ID:	ISO on TCP (RFC 1006) and S7 communication: Local and partner CPU TSAPs in ASCII and hexadecimal formats

[1] When configuring a connection with an S7-1200 CPU for ISO-on-TCP, use only ASCII characters in the TSAP extension for the passive communication partners.

Transport Service Access Points (TSAPs)

Using TSAPs, ISO on TCP protocol and S7 communication allows multiple connections to a single IP address (up to 64K connections). TSAPs uniquely identify these communication end point connections to an IP address.

In the "Address Details" section of the Connection Parameters dialog, you define the TSAPs to be used. The TSAP of a connection in the CPU is entered in the "Local TSAP" field. The TSAP assigned for the connection in your partner CPU is entered under the "Partner TSAP" field.

Port Numbers

With TCP and UDP protocols, the connection parameter configuration of the Local (active) connection CPU must specify the remote IP address and port number of the Partner (passive) connection CPU.

In the "Address Details" section of the Connection Parameters dialog, you define the ports to be used. The port of a connection in the CPU is entered in the "Local Port" field. The port assigned for the connection in your partner CPU is entered under the "Partner Port" field.

11.2.3 Assigning Internet Protocol (IP) addresses

11.2.3.1 Assigning IP addresses to programming and network devices

If your programming device is using an on-board adapter card connected to your plant LAN (and possibly the world-wide web), the IP Address Network ID and subnet mask of your CPU and the programming device's on-board adapter card must be exactly the same. 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 mathematical 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 communicate with the world, you must assign unique IP addresses 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.

WARNING

Unauthorized access to the CPU through the Web server

Unauthorized access to the CPU or changing PLC variables to invalid values could disrupt process operation and could result in death, severe personal injury and/or property damage.

Enabling the Web server allows authorized users to perform operating mode changes, writes to PLC data, and firmware updates, Siemens recommends that you observe the following security practices:

- Enable access to the Web server only with the HTTPS protocol.
- Password-protect Web server user IDs (Page 789) with a strong password. Strong passwords are at least ten characters in length, mix letters, numbers, and special characters, are not words that can be found in a dictionary, and are not names or identifiers that can be derived from personal information. Keep the password secret and change it frequently.
- Do not extend the default minimum privileges of the "Everybody" user.
- Perform error-checking and range-checking on your variables in your program logic because Web page users can change PLC variables to invalid values.

If your programming device is using an Ethernet-to-USB adapter card connected to an isolated network, the IP Address Network ID and subnet mask of your CPU and the programming device's Ethernet-to-USB adapter card must be exactly the same. 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**. The subnet mask, when combined with the device IP address in a mathematical AND operation, defines the boundaries of an IP subnet.

Note

An Ethernet-to-USB adapter card is useful when you do not want your programming device on your company LAN. During initial testing or commissioning tests, this arrangement is particularly useful.

Table 11- 5 Assigning Ethernet addresses

Programming Device Adapter Card	Network Type	Internet Protocol (IP) Address	Subnet Mask
On-board adapter card	Connected to your plant LAN (and possibly the world-wide web)	Network ID of your CPU and the programming device's on-board adapter card must be exactly the same.[1]	The subnet mask of your CPU and the on-board adapter card must be exactly the same. 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 ex-ample, **255.255.254.0**) in order to set up unique subnets.[2]
Ethernet-to-USB adapter card	Connected to an isolated network	Network ID of your CPU and the programming device's Ethernet-to-USB adapter card must be exactly the same.[1]	The subnet mask of your CPU and the Ethernet-to-USB adapter card must be exactly the same. The subnet mask normally has a value of **255.255.255.0**.[2]

[1] 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.)

[2] The subnet mask, when combined with the device IP address in a mathematical AND operation, defines the boundaries of an IP subnet.

Assigning or checking the IP address of your programming device using "My Network Places" (on your desktop)

You can assign or check your programming device's IP address with the following menu selections:

- (Right-click) "My Network Places"
- "Properties"
- (Right-click) "Local Area Connection"
- "Properties"

In the "Local Area Connection Properties" dialog, in the "This connection uses the following items:" field, scroll down to "Internet Protocol (TCP/IP)". Click "Internet Protocol (TCP/IP)", and click the "Properties" button. Select "Obtain an IP address automatically (DHCP)" or "Use the following IP address" (to enter a static IP address).

Note

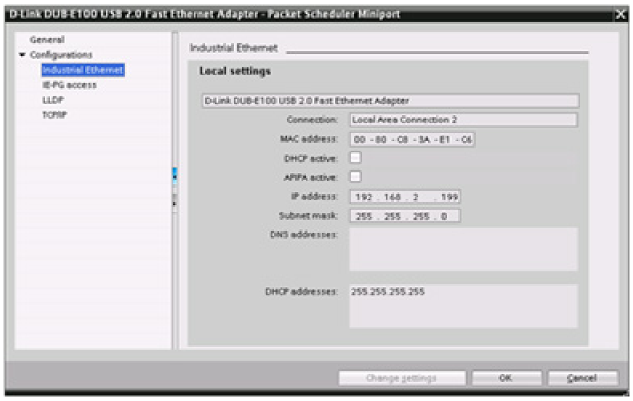
Dynamic Host Configuration Protocol (DHCP) automatically assigns an IP address to your programming device upon power up from the DHCP server.

11.2.3.2 Checking the IP address of your programming device

You can check the MAC and IP addresses of your programming device with the following menu selections:

1. In the "Project tree", expand "Online access".
2. Right-click the required network, and select "Properties".
3. In the network dialog, expand "Configurations", and select "Industrial Ethernet".

The MAC and IP addresses of the programming device are displayed.



Local settings

D-Link DUB-E100 USB 2.0 Fast Ethernet Adapter

Connection: Local Area Connection 2
MAC address: 00 - 80 - C8 - 3A - E1 - C6
DHCP active: ☐
APIPA active: ☐
IP address: 192 . 168 . 2 . 199
Subnet mask: 255 . 255 . 255 . 0
DNS addresses:
DHCP addresses: 255.255.255.255

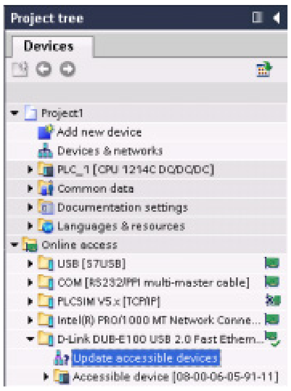
11.2.3.3 Assigning an IP address to a CPU online

You can assign an IP address to a network device online. This is particularly useful in an initial device configuration.

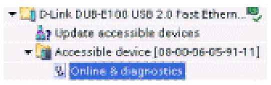
1. In the "Project tree," verify that no IP address is assigned to the CPU, with the following menu selections:

- "Online access"
- <Adapter card for the network in which the device is located>
- "Update accessible devices"

NOTE: If STEP 7 displays a MAC address instead of an IP address, then no IP address has been assigned.

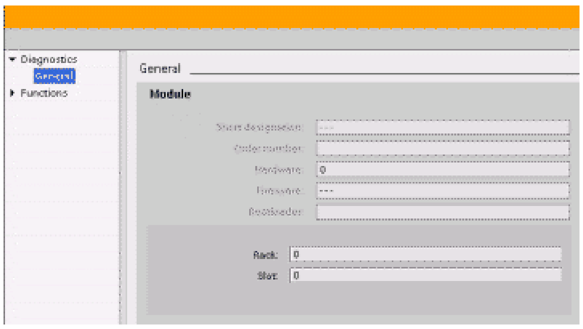

Project tree
Devices
Project1
Add new device
Devices & networks
PLC_1 [CPU 1214C DC/DC/DC]
Common data
Documentation settings
Languages & resources
Online access
USB [S7USB]
COM [RS232/PPI multi-master cable]
PLCSIM V5.x [TCP/IP]
Intel(R) PRO/1000 MT Network Conne...
D-Link DUB-E100 USB 2.0 Fast Ethern...
Update accessible devices
Accessible device [08-00-06-05-91-11]

2. Under the required accessible device, double-click "Online & diagnostics".

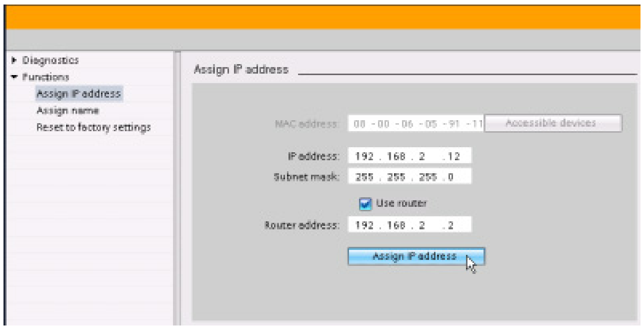

D-Link DUB-E100 USB 2.0 Fast Ethern...
Update accessible devices
Accessible device [08-00-06-05-91-11]
Online & diagnostics

3. In the "Online & diagnostics" dialog, make the following menu selections:

- "Functions"
- "Assign IP address"



4. In the "IP address" field, enter your new IP address, and click the "Assign IP address" button.

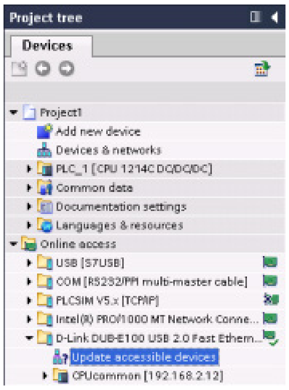


Assign IP address

MAC address: 00 - 00 - 06 - 05 - 91 - 11
IP address: 192 . 168 . 2 . 12
Subnet mask: 255 . 255 . 255 . 0
[✓] Use router
Router address: 192 . 168 . 2 . 2

5. In the "Project tree," verify that your new IP address has been assigned to the CPU, with the following menu selections:

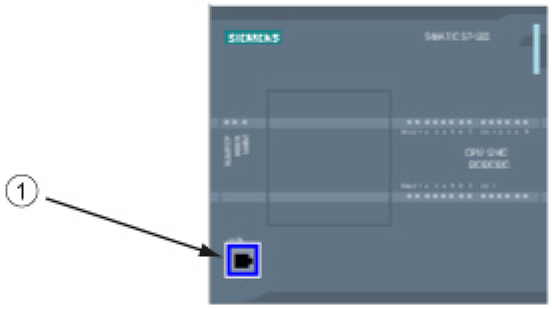
- "Online access"
- <Adapter for the network in which the device is located>
- "Update accessible devices"



11.2.3.4 Configuring an IP address for a CPU in your project

Configuring the PROFINET interface

To configure parameters for the PROFINET interface, select the green PROFINET box on the CPU. The "Properties" tab in the inspector window displays the PROFINET port.



① PROFINET port

Configuring the IP address

Ethernet (MAC) address: In a PROFINET network, each device is assigned a Media Access Control address (MAC address) by the manufacturer for identification. A MAC address consists of six groups of two hexadecimal digits, separated by hyphens (-) or colons (:), in transmission order, (for example, 01-23-45-67-89-AB or 01:23:45:67:89:AB).

IP address: Each device must also have an Internet Protocol (IP) address. This address allows the device to deliver data on a more complex, routed network.

Each IP address is divided into four 8-bit segments and is expressed in a dotted, decimal format (for example, 211.154.184.16). The first part of the IP address is used for the Network ID (What network are you on?), and the second part of the address is for the Host ID (unique for each device on the network). An IP address of 192.168.x.y is a standard designation recognized as part of a private network that is not routed on the Internet.

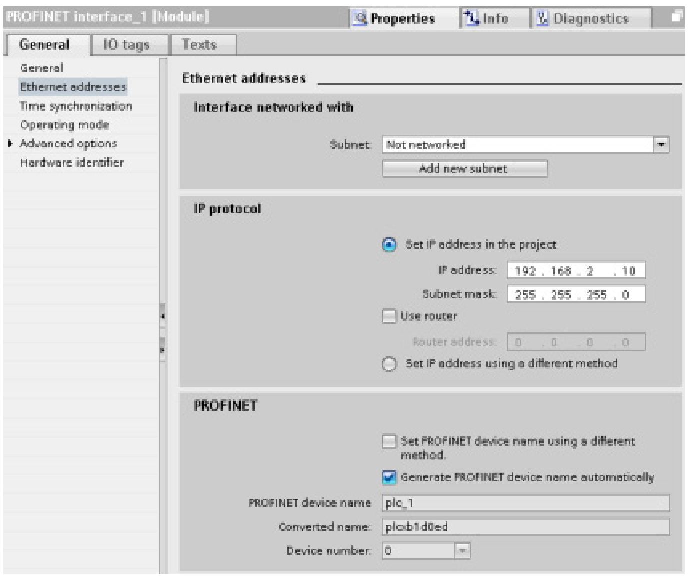
Subnet mask: A subnet is a logical grouping of connected network devices. Nodes on a subnet tend to be located in close physical proximity to each other on a Local Area Network (LAN). A mask (known as the subnet mask or network mask) defines the boundaries of an IP subnet.

A subnet mask of 255.255.255.0 is generally suitable for a small local network. This means that all IP addresses on this network should have the same first 3 octets, and the various devices on this network are identified by the last octet (8-bit field). An example of this is to assign a subnet mask of 255.255.255.0 and an IP addresses of 192.168.2.0 through 192.168.2.255 to the devices on a small local network.

The only connection between different subnets is via a router. If subnets are used, an IP router must be employed.

IP router: Routers are the link between LANs. Using a router, a computer in a LAN can send messages to any other networks, which might have other LANs behind them. If the destination of the data is not within the LAN, the router forwards the data to another network or group of networks where it can be delivered to its destination.

Routers rely on IP addresses to deliver and receive data packets.



Ethernet addresses

Interface networked with

Subnet: Not networked

IP protocol

[✓ Set IP address in the project]

IP address: 192 . 168 . 2 . 10

Subnet mask: 255 . 255 . 255 . 0

Use router

Router address: 0 . 0 . 0 . 0

Set IP address using a different method

PROFINET

Set PROFINET device name using a different method.

[✓ Generate PROFINET device name automatically]

PROFINET device name: plc_1

Converted name: plcxb1d0ed

Device number: 0

IP addresses properties: In the Properties window, select the "Ethernet addresses" configuration entry. STEP 7 displays the Ethernet address configuration dialog, which associates the software project with the IP address of the CPU that will receive that project.

Table 11- 6 Parameters for the IP address

Parameter		Description
Subnet	Name of the Subnet to which the device is connected. Click the "Add new subnet" button to create a new subnet. "Not connected" is the default. Two connection types are possible: • The "Not connected" default provides a local connection. • A subnet is required when your network has two or more devices.	
IP protocol	IP address	Assigned IP address for the CPU
	Subnet mask	Assigned subnet mask
	Use IP router	Click the checkbox to indicate the use of an IP router
	Router address	Assigned IP address for the router, if applicable

Note

All IP addresses are configured when you download the project. If the CPU does not have a pre-configured IP address, you must associate the project with the MAC address of the target device. If your CPU is connected to a router on a network, you must also enter the IP address of the router.

The "Set IP address using a different method" radio button allows you to change the IP address online or by using the "T_CONFIG (Page 704)" instruction after the program is downloaded. This IP address assignment method is for the CPU only.

WARNING

Downloading a hardware configuration with "Set IP address using different method"

After downloading a hardware configuration with the "Set IP address using a different method" option enabled, it is not possible to transition the CPU operating mode from RUN to STOP or from STOP to RUN.

User equipment continues to run under these conditions and can result in unexpected machine or process operations, which could cause death, severe personal injury, or property damage if proper precautions are not taken.

Ensure that your CPU IP address(es) are set before using the CPU in an actual automation environment. This can be done by using your STEP 7 programming package, the S7-1200 Tool, or an attached HMI device in conjunction with the T_CONFIG instruction.

⚠ WARNING

Condition in which PROFINET network might stop

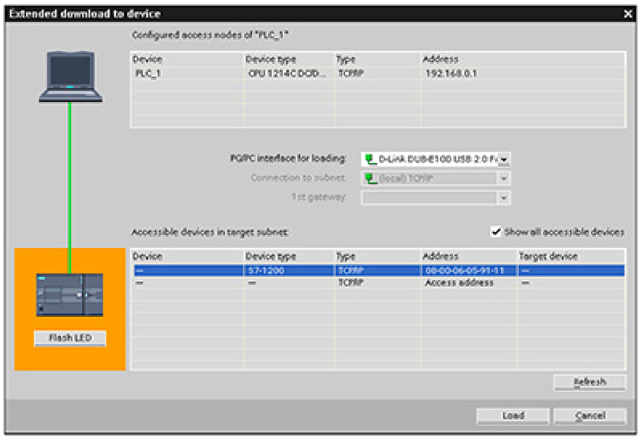
When changing the IP address of a CPU online or from the user program, it is possible to create a condition in which the PROFINET network might stop.

If the IP address of a CPU is changed to an IP address outside the subnet, the PROFINET network will lose communication, and all data exchange will stop. User equipment may be configured to keep running under these conditions. Loss of PROFINET communication may result in unexpected machine or process operations, causing death, severe personal injury, or property damage if proper precautions are not taken.

If an IP address must be changed manually, ensure that the new IP address lies within the subnet.

11.2.4 Testing the PROFINET network

After completing the configuration, download the project (Page 203) to the CPU. All IP addresses are configured when you download the project.


Extended download to device

Configured access nodes of "PLC_1"

Device	Device type	Type	Address
PLC_1	CPU 1214C DC/D...	TCP/IP	192.168.0.1

PG/PC interface for loading: D-Link DUB-E100 USB 2.0 F...
Connection to subnet: (local) TCP/IP
1st gateway:

Accessible devices in target subnet:
✓ Show all accessible devices

Device	Device type	Type	Address	Target device
—	S7-1200	TCP/IP	08-00-06-05-91-11	—
—	—	TCP/IP	Access address	—

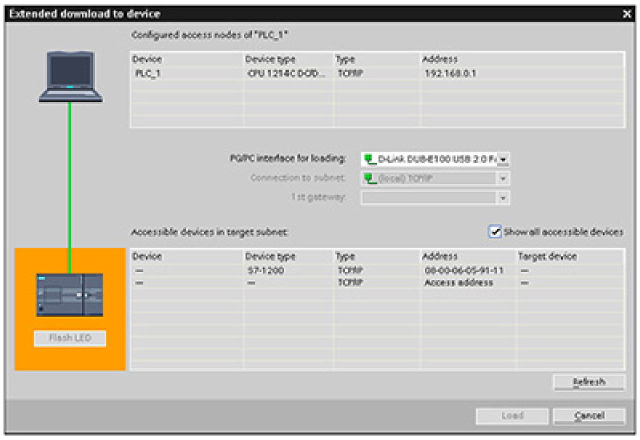
Assigning an IP address to a device online

The S7-1200 CPU does not have a pre-configured IP address. You must manually assign an IP address for the CPU:

- To assign an IP address to a device online, refer to "Device configuration: Assigning an IP address to a CPU online" (Page 625) for this step-by-step procedure.
- To assign an IP address in your project, you must configure the IP address in the Device configuration, save the configuration, and download it to the PLC. Refer to "Device configuration: Configuring an IP address for a CPU in your project" (Page 626) for more information.

Using the "Extended download to device" dialog to test for connected network devices

The S7-1200 CPU "Download to device" function and its "Extended download to device" dialog can show all accessible network devices and whether or not unique IP addresses have been assigned to all devices. To display all accessible and available devices with their assigned MAC or IP addresses, check the "Show all accessible devices" checkbox.


Extended download to device

Configured access nodes of "PLC_1"

Device	Device type	Type	Address
PLC_1	CPU 1214C DC/D...	TCP/IP	192.168.0.1

PG/PC interface for loading: D-Link DUB-E100 USB 2.0 F...
Connection to subnet: (local) TCP/IP
1st gateway:

Accessible devices in target subnet:
✓ Show all accessible devices

Device	Device type	Type	Address	Target device
—	S7-1200	TCP/IP	08-00-06-05-91-11	—
—	—	TCP/IP	Access address	—

If the required network device is not in this list, communications to that device have been interrupted for some reason. The device and network must be investigated for hardware and/or configuration errors.

11.2.5 Locating the Ethernet (MAC) address on the CPU

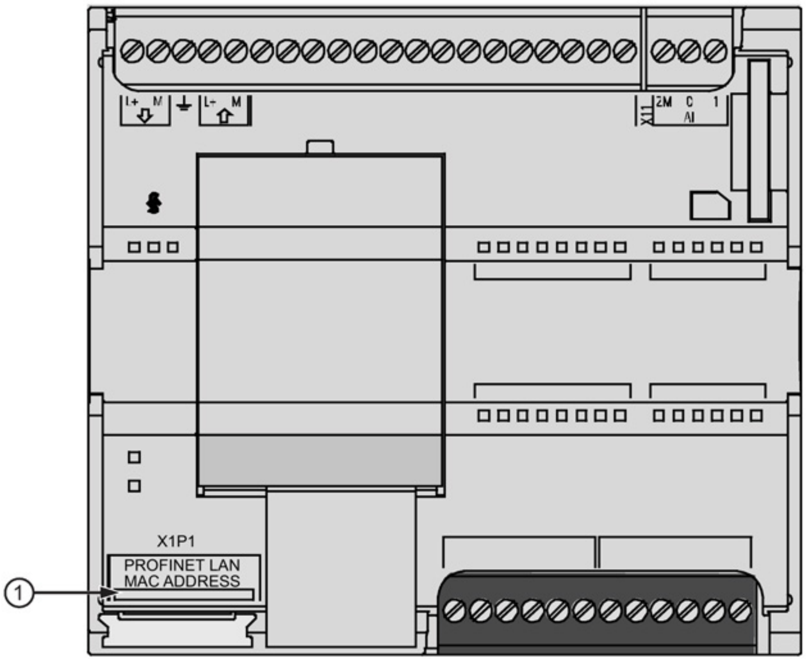
In PROFINET networking, a Media Access Control address (MAC address) is an identifier assigned to the network interface by the manufacturer for identification. A MAC address usually encodes the manufacturer's registered identification number.

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 (:), in transmission order, (for example, 01-23-45-67-89-ab or 01:23:45:67:89:ab).

Note

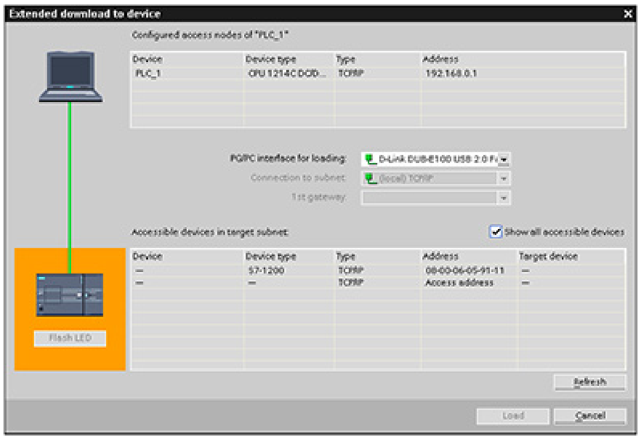
Each CPU is loaded at the factory with a permanent, unique MAC address. You cannot change the MAC address of a CPU.

The MAC address is printed on the front, lower-left corner of the CPU. Note that you have to lift the lower door to see the MAC address information.



① MAC address

Initially, the CPU has no IP address, only a factory-installed MAC address. PROFINET communications requires that all devices be assigned a unique IP address.


Extended download to device

Configured access nodes of "PLC_1"

Device	Device type	Type	Address
PLC_1	CPU 1214C DC/D...	TCP/IP	192.168.0.1

PG/PC interface for loading: D-Link DUB-E100 USB 2.0 F...
Connection to subnet: (local) TCP/IP
1st gateway:

Accessible devices in target subnet: ☑ Show all accessible devices

Device	Device type	Type	Address	Target device
—	S7-1200	TCP/IP	08-00-06-05-91-11	—
—	—	TCP/IP	Access address	—

Use the CPU "Download to device" function and the "Extended download to device" dialog to show all accessible network devices and ensure that unique IP addresses have been assigned to all devices. This dialog displays all accessible and available devices with their assigned MAC or IP addresses. MAC addresses are all-important in identifying devices that are missing the required unique IP address.

11.2.6 Configuring Network Time Protocol synchronization

WARNING

If an attacker can access your networks through Network Time Protocol (NTP) synchronization, the attacker can possibly take limited control of your process by shifting the CPU system time.

The NTP client feature of the S7-1200 CPU is disabled by default, and, when enabled, only allows configured IP addresses to act as an NTP server. The CPU disables this feature by default, and you must configure this feature to allow remotely-controlled CPU system time corrections.

The S7-1200 CPU supports "time of day" interrupts and clock instructions that depend upon accurate CPU system time. If you configure NTP and accept time synchronization from a server, you must ensure that the server is a trusted source. Failure to do so can cause a security breach that allows an unknown user to take limited control of your process by shifting the CPU system time.

For security information and recommendations, please see our "Operational Guidelines for Industrial Security" (http://www.industry.siemens.com/topics/global/en/industrial-security/Documents/operational_guidelines_industrial_security_en.pdf) on the Siemens Service and Support site.

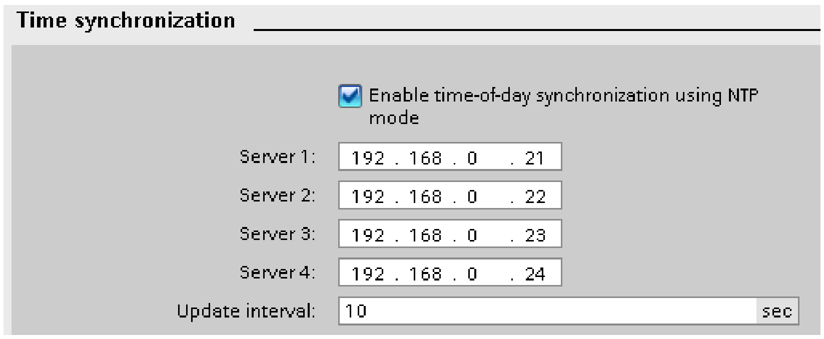
The Network Time Protocol (NTP) is widely used to synchronize the clocks of computer systems to Internet time servers. In NTP mode, the CPU sends time-of-day queries at regular intervals (in the client mode) to the NTP server in the subnet (LAN). Based on the replies from the server, the most reliable and most accurate time is calculated and the time of day on the station is synchronized.

The advantage of this mode is that it allows the time to be synchronized across subnets.

The IP addresses of up to four NTP servers need to be configured. The update interval defines the interval between the time queries (in seconds). The value of the interval ranges between 10 seconds and one day.

In NTP mode, it is generally UTC (Universal Time Coordinated) that is transferred; this corresponds to GMT (Greenwich Mean Time).

In the Properties window, select the "Time synchronization" configuration entry. STEP 7 displays the Time synchronization configuration dialog:


Time synchronization

[✓] Enable time-of-day synchronization using NTP mode

Server 1: 192 . 168 . 0 . 21
Server 2: 192 . 168 . 0 . 22
Server 3: 192 . 168 . 0 . 23
Server 4: 192 . 168 . 0 . 24
Update interval: 10 sec

Note

All IP addresses are configured when you download the project.

Table 11- 7 Parameters for time synchronization

Parameter	Definition
Enable time-of-day synchronization using Network Time Protocol (NTP) servers	Click the checkbox to enable time-of-day synchronization using NTP servers.
Server 1	Assigned IP Address for network time server 1
Server 2	Assigned IP Address for network time server 2
Server 3	Assigned IP Address for network time server 3
Server 4	Assigned IP Address for network time server 4
Time synchronization interval	Interval value (sec)

11.2.7 PROFINET device start-up time, naming, and address assignment

PROFINET IO can extend the start-up time for your system (configurable time-out). More devices and slow devices impact the amount of time it takes to switch to RUN.

In V4.0 and later, you can have a maximum of 16 PROFINET IO devices on your S7-1200 PROFINET network.

Each station (or IO device) starts up independently on start-up, and this affects the overall CPU start-up time. If you set the configurable time-out too low, there may not be a sufficient overall CPU start-up time for all stations to complete start-up. If this situation occurs, false station errors will result.

In the CPU Properties under "Startup", you can find the "Parameter assignment time for distributed I/O" (time-out). The default configurable time-out is 60,000 ms (1 minute); the user can configure this time.

PROFINET device naming and addressing in STEP 7

All PROFINET devices **must** have a Device Name and an IP Address. Use STEP 7 to define the Device Names and to configure the IP addresses. Device names are downloaded to the IO devices using PROFINET DCP (Discovery and Configuration Protocol).

PROFINET address assignment at system start-up

The controller broadcasts the names of the devices to the network, and the devices respond with their MAC addresses. The controller then assigns an IP address to the device using PROFINET DCP protocol:

- If the MAC address has a configured IP address, then the station performs start-up.
- If the MAC address does not have a configured IP address, STEP 7 assigns the address that is configured in the project, and the station then performs start-up.
- If there is a problem with this process, a station error occurs and no start-up takes place. This situation causes the configurable time-out value to be exceeded.

11.2.8 Open user communication

11.2.8.1 Protocols

The integrated PROFINET port of the CPU supports multiple communications standards over an Ethernet network:

- Transport Control Protocol (TCP)
- ISO on TCP (RFC 1006)
- User Datagram Protocol (UDP)

Table 11- 8 Protocols and communication instructions for each

Protocol	**Usage examples**	**Entering data in the receive area**	**Communication instructions**	**Addressing type**
TCP	CPU-to-CPU communication Transport of frames	Ad hoc mode	Only TRCV_C and TRCV (V4.1 and legacy instructions)	Assigns port numbers to the Local (active) and Partner (passive) devices
		Data reception with specified length	TSEND_C, TRCV_C, TCON, TDISCON, TSEND, and TRCV(V4.1 and legacy instructions)	
ISO on TCP	CPU-to-CPU communication Message fragmentation and re-assembly	Ad hoc mode	Only TRCV_C and TRCV (V4.1 and legacy instructions)	Assigns TSAPs to the Local (active) and Partner (passive) devices
		Protocol-controlled	TSEND_C, TRCV_C, TCON, TDISCON, TSEND, and TRCV (V4.1 and legacy instructions)	

Protocol	Usage examples	Entering data in the receive area	Communication instructions	Addressing type
UDP	CPU-to-CPU communication User program communications	User Datagram Protocol	TUSEND and TURCV	Assigns port numbers to the Local (active) and Partner (passive) devices, but is not a dedicated connection
S7 communication	CPU-to-CPU communication Read/write data from/to a CPU	Data transmission and reception with specified length	GET and PUT	Assigns TSAPs to the Local (active) and Partner (passive) devices
PROFINET IO	CPU-to-PROFINET IO device communication	Data transmission and reception with specified length	Built-in	Built-in

11.2.8.2 TCP and ISO on TCP

Transport Control Protocol (TCP) is a standard protocol described by RFC 793: Transmission Control Protocol. The primary purpose of TCP is to provide reliable, secure connection service between pairs of processes. This protocol has the following features:

- An efficient communications protocol since it is closely tied to the hardware
- Suitable for medium-sized to large data amounts (up to 8192 bytes)
- Provides considerably more facilities for applications, notably error recovery, flow control, and reliability
- A connection-oriented protocol
- Can be used very flexibly with third-party systems which exclusively support TCP
- Routing-capable
- Only static data lengths are applicable.
- Messages are acknowledged.
- Applications are addressed using port numbers.
- Most of the user application protocols, such as TELNET and FTP, use TCP.
- Programming effort is required for data management due to the SEND / RECEIVE programming interface.

International Standards Organization (ISO) on Transport Control Protocol (TCP) (RFC 1006) (ISO on TCP) is a mechanism that enables ISO applications to be ported to the TCP/IP network. This protocol has the following features:

- An efficient communications protocol closely tied to the hardware
- Suitable for medium-sized to large data amounts (up to 8192 bytes)
- In contrast to TCP, the messages feature an end-of-data identification and are message-oriented.
- Routing-capable; can be used in WAN

- Dynamic data lengths are possible.
- Programming effort is required for data management due to the SEND / RECEIVE programming interface.

Using Transport Service Access Points (TSAPs), TCP protocol allows multiple connections to a single IP address (up to 64K connections). With RFC 1006, TSAPs uniquely identify these communication end point connections to an IP address.

11.2.8.3 Communication services and used port numbers

The S7-1200 CPU supports the protocols listed in the table below. For each protocol, the CPU assigns the address parameters, the respective communications layer as well as the communications role, and the communications direction.

This information makes it possible to match the security measures for protection of the automation system to the used protocols (for example, firewall). Only the Ethernet or PROFINET networks have security measures. Since PROFIBUS does not have any security measures, the table does not include any PROFIBUS protocols.

The table below shows the different layers and protocols that the CPU uses:

Protocol	**Port number**	**(2) Link layer** **(4) Transport layer**	**Function**	**Description**
PROFINET protocols				
DCP (Discovery and Configuration Protocol)	Not relevant	(2) Ethernet II and IEEE 802.1Q and Ethertype 0x8892 (PROFINET)	Accessible devic-es PROFINET Discovery and configuration	PROFINET uses DCP to discover devices and pro-vide basic set-tings. DCP uses the special multicast MAC address: xx-xx-xx-01-0E-CF, xx-xx-xx = Organi-zationally Unique Indentifier
LLDP (Link Layer Dis-covery Protocol)	Not relevant	(2) Ethernet II and IEEE 802.1Q and Ethertype 0x88CC (PROFINET)	PROFINET Link Layer Discovery protocol	PROFINET uses LLDP to discover and manage neighbor relation-ships between PROFINET devic-es. LLDP uses the special multicast MAC address: 01-80-C2-00-00-0E

11.2.8.4 Ad hoc mode

Typically, TCP and ISO-on-TCP receive data packets of a specified length, ranging from 1 to 8192 bytes. However, the TRCV_C and TRCV communication instructions also provide an "ad hoc" communications mode that can receive data packets of a variable length from 1 to 1472 bytes.

Note

If you store the data in an "optimized" DB (symbolic only), you can receive data only in arrays of Byte, Char, USInt, and SInt data types.

To configure the TRCV_C or TRCV instruction for ad hoc mode, set the ADHOC instruction input parameter.

If you do not call the TRCV_C or TRCV instruction in ad hoc mode frequently, you could receive more than one packet in one call. For example: If you were to receive five 100-byte packets with one call, TCP would deliver these five packets as one 500-byte packet, while ISO-on-TCP would restructure the packets into five 100-byte packets.

11.2.8.5 Connection IDs for the Open user communication instructions

When you insert the TSEND_C, TRCV_C or TCON PROFINET instructions into your user program, STEP 7 creates an instance DB to configure the communications channel (or connection) between the devices. Use the "Properties" (Page 619) of the instruction to configure the parameters for the connection. Among the parameters is the connection ID for that connection.

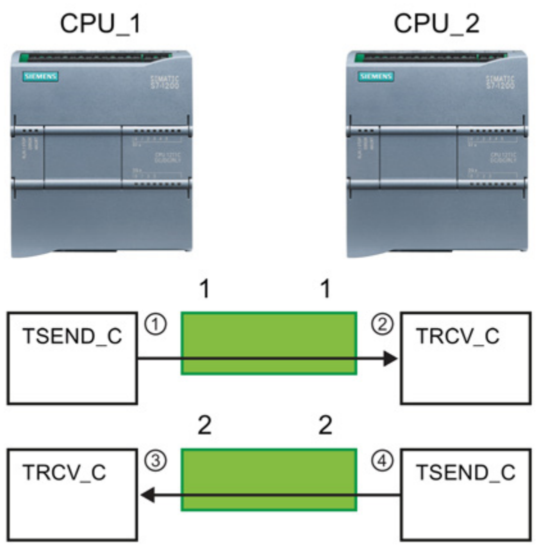
- The connection ID must be unique for the CPU. Each connection that you create must have a different DB and connection ID.
- Both the local CPU and the partner CPU can use the same connection ID number for the same connection, but the connection ID numbers are not required to match. The connection ID number is relevant only for the PROFINET instructions within the user program of the individual CPU.
- You can use any number for the connection ID of the CPU. However, configuring the connection IDs sequentially from "1" provides an easy method for tracking the number of connections in use for a specific CPU.

Note

Each TSEND_C, TRCV_C or TCON instruction in your user program creates a new connection. It is important to use the correct connection ID for each connection.

The following example shows the communication between two CPUs that utilize two separate connections for sending and receiving the data.

- The TSEND_C instruction in CPU_1 links to the TRCV_C in CPU_2 over the first connection ("connection ID 1" on both CPU_1 and CPU_2).
- The TRCV_C instruction in CPU_1 links to the TSEND_C in CPU_2 over the second connection ("connection ID 2" on both CPU_1 and CPU_2).



① TSEND_C on CPU_1 creates a connection and assigns a connection ID to that connection (connection ID 1 for CPU_1).

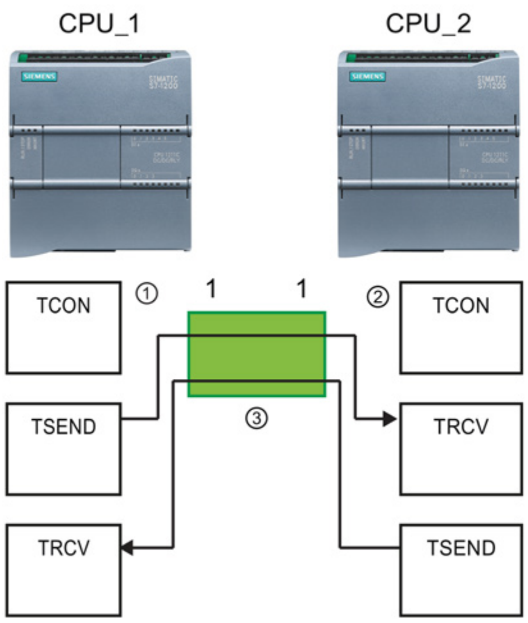
② TRCV_C on CPU_2 creates the connection for CPU_2 and assigns the connection ID (connection ID 1 for CPU_2).

③ TRCV_C on CPU_1 creates a second connection for CPU_1 and assigns a different connection ID for that connection (connection ID 2 for CPU_1).

④ TSEND_C on CPU_2 creates a second connection and assigns a different connection ID for that connection (connection ID 2 for CPU_2).

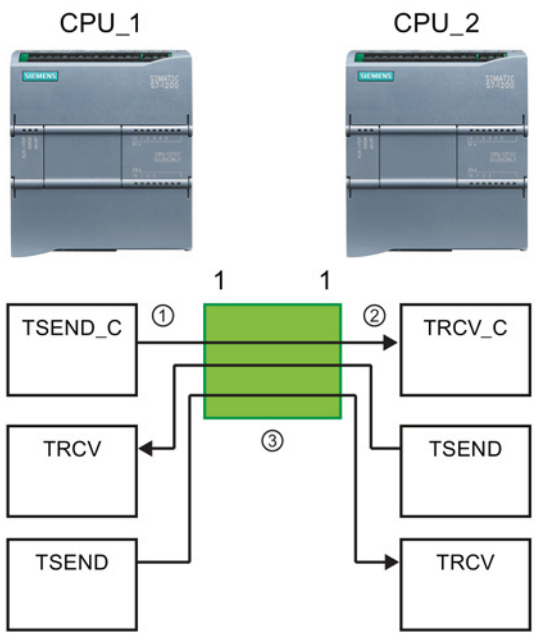
The following example shows the communication between two CPUs that utilize 1 connection for both sending and receiving the data.

- Each CPU uses a TCON instruction to configure the connection between the two CPUs.
- The TSEND instruction in CPU_1 links to the TRCV instruction in CPU_2 by using the connection ID ("connection ID 1") that was configured by the TCON instruction in CPU_1. The TRCV instruction in CPU_2 links to the TSEND instruction in CPU_1 by using the connection ID ("connection ID 1") that was configured by the TCON instruction in CPU_2.
- The TSEND instruction in CPU_2 links to the TRCV instruction in CPU_1 by using the connection ID ("connection ID 1") that was configured by the TCON instruction in CPU_2. The TRCV instruction in CPU_1 links to the TSEND instruction in CPU_2 by using the connection ID ("connection ID 1") that was configured by the TCON instruction in CPU_1.



①	TCON on CPU_1 creates a connection and assigns a connection ID for that connection on CPU_1 (ID=1).
②	TCON on CPU_2 creates a connection and assigns a connection ID for that connection on CPU_2 (ID=1).
③	TSEND and TRCV on CPU_1 use the connection ID created by the TCON on CPU_1 (ID=1). TSEND and TRCV on CPU_2 use the connection ID created by the TCON on CPU_2 (ID=1).

As shown in the following example, you can also use individual TSEND and TRCV instruction to communication over a connection created by a TSEND_C or TRCV_C instruction. The TSEND and TRCV instructions do not themselves create a new connection, so must use the DB and connection ID that was created by a TSEND_C, TRCV_C or TCON instruction.



①	TSEND_C on CPU_1 creates a connection and assigns a connection ID to that connection (ID=1).
②	TRCV_C on CPU_2 creates a connection and assigns the connection ID to that connection on CPU_2 (ID=1).
③	TSEND and TRCV on CPU_1 use the connection ID created by the TSEND_C on CPU_1 (ID=1). TSEND and TRCV on CPU_2 use the connection ID created by the TRCV_C on CPU_2 (ID=1).

11.2.8.6 Parameters for the PROFINET connection

The TSEND_C, TRCV_C and TCON instructions require that connection-related parameters be specified in order to connect to the partner device. These parameters are assigned by the TCON_Param structure for the TCP, ISO-on-TCP, and UDP protocols. Typically, you use the "Configuration" (Page 619) tab of the "Properties" of the instruction to specify these parameters. If the "Configuration" tab is not accessible, then you must specify the TCON_Param structure programmatically.

With V4.1, the TCON_IP_V4 structure assigns parameters for the TCP protocol, and the TCON_IP_RFC structure assigns parameters for the ISO-on-TCP protocol.

TCON_Param

Table 11- 9 Structure of the connection description (TCON_Param)

Byte	Parameter and data type		Description
0 … 1	block_length	UInt	Length: 64 bytes (fixed)
2 … 3	id	CONN_OUC (Word)	Reference to this connection: Range of values: 1 (default) to 4095. Specify the value of this parameter for the TSEND_C, TRCV_C or TCON instruction under ID.
4	connection_type	USInt	Connection type: • 17: TCP (default) • 18: ISO-on-TCP • 19: UDP
5	active_est	Bool	ID for the type of connection: • TCP and ISO-on-TCP: – FALSE: Passive connection – TRUE: Active connection (default) • UDP: FALSE
6	local_device_id	USInt	ID for the local PROFINET or Industrial Ethernet interface: 1 (default)
7	local_tsap_id_len	USInt	Length of parameter local_tsap_id used, in bytes; possible values: • TCP: 0 (active, default) or 2 (passive) • ISO-on-TCP: 2 to 16 • UDP: 2
8	rem_subnet_id_len	USInt	This parameter is not used.
9	rem_staddr_len	USInt	Length of address of partner end point, in bytes: • 0: unspecified (parameter rem_staddr is irrelevant) • 4 (default): Valid IP address in parameter rem_staddr (only for TCP and ISO-on-TCP)
10	rem_tsap_id_len	USInt	Length of parameter rem_tsap_id used, in bytes; possible values: • TCP: 0 (passive) or 2 (active, default) • ISO-on-TCP: 2 to 16 • UDP: 0

Byte	Parameter and data type		Description
11	next_staddr_len	USInt	This parameter is not used.
12 … 27	local_tsap_id	Array [1..16] of Byte	Local address component of connection: • TCP and ISO-on-TCP: local port no. (possible values: 1 to 49151; recommended values: 2000...5000): – local_tsap_id[1] = high byte of port number in hexadecimal notation; – local_tsap_id[2] = low byte of port number in hexadecimal notation; – local_tsap_id[3-16] = irrelevant • ISO-on-TCP: local TSAP-ID: – local_tsap_id[1] = B#16#E0; – local_tsap_id[2] = rack and slot of local end points (bits 0 to 4: slot number, bits 5 to 7: rack number); – local_tsap_id[3-16] = TSAP extension, optional • UDP: This parameter is not used. Note: Make sure that every value of local_tsap_id is unique within the CPU.
28 … 33	rem_subnet_id	Array [1..6] of USInt	This parameter is not used.

34 … 39	rem_staddr	Array [1..6] of USInt	TCP and ISO-on-TCP only: IP address of the partner end point. (Not relevant for passive connections.) For example, IP address 192.168.002.003 is stored in the following elements of the array:
			rem_staddr[1] = 192
			rem_staddr[2] = 168
			rem_staddr[3] = 002
			rem_staddr[4] = 003
			rem_staddr[5-6]= irrelevant

40 … 55	rem_tsap_id	Array [1..16] of Byte	Partner address component of connection • TCP: partner port number. Range: 1 to 49151; Recommended values: 2000 to 5000): – rem_tsap_id[1] = high byte of the port number in hexadecimal notation – rem_tsap_id[2] = low byte of the port number in hexadecimal notation; – rem_tsap_id[3-16] = irrelevant • ISO-on-TCP: partner TSAP-ID: – rem_tsap_id[1] = B#16#E0 – rem_tsap_id[2] = rack and slot of partner end point (bits 0 to 4: Slot number, bits 5 to 7: rack number) – rem_tsap_id[3-16] = TSAP extension, optional • UDP: This parameter is not used.
56 … 61	next_staddr	Array [1..6] of Byte	This parameter is not used.
62 … 63	spare	Word	Reserved: W#16#0000

TCON_IP_V4

Table 11- 10 Structure of the connection description (TCON_IP_V4): For use with TCP

Byte	Parameter and data type		Description
0 … 1	InterfaceId	HW_ANY	HW-identifier of the IE-interface submodule
2 … 3	ID	CONN_OUC (Word)	Reference to this connection: Range of values: 1 (default) to 4095. Specify the value of this parameter for the TSEND_C, TRCV_C, or TCON instruction under ID.
4	ConnectionType	Byte	Connection type: • 11: TCP/IP (default) • 17: TCP/IP (This connection type is included for legacy reasons. It is recommended that you use "11: TCP/IP (default)".) • 19: UDP
5	ActiveEstablished	Bool	Active/passive connection establishment: • TRUE: Active connection (default) • FALSE: Passive connection
	V4 IP address		
6	ADDR[1]	Byte	Octet 1
7	ADDR[1]	Byte	Octet 2
8	ADDR[1]	Byte	Octet 3
9	ADDR[1]	Byte	Octet 4
10 ... 11	RemotePort	UInt	Remote UDP/TCP port number
12 ... 13	LocalPort	UInt	Local UDP/TCP port number

TCON_IP_RFC

Table 11- 11 Structure of the connection description (TCON_IP_RFC): For use with ISO on TCP

Byte	Parameter and data type		Description
0 … 1	InterfaceId	HW_ANY	HW-identifier of the IE-interface submodule
2 … 3	ID	CONN_OUC (Word)	Reference to this connection: Range of values: 1 (default) to 4095. Specify the value of this parameter for the TSEND_C, TRCV_C, or TCON instruction under ID.
4	ConnectionType	Byte	Connection type: • 12: ISO-on-TCP (default) • 17: ISO-on-TCP (This connection type is included for legacy reasons. It is recommended that you use "12: ISO-on-TCP (default)".)
5	ActiveEstablished	Bool	Active/passive connection establishment: • TRUE: Active connection (default) • FALSE: Passive connection

Byte	Parameter and data type		Description
6 ... 7	Spare		Not used
	V4 IP address		
8	ADDR[1]	Byte	Octet 1
9	ADDR[1]	Byte	Octet 2
10	ADDR[1]	Byte	Octet 3
11	ADDR[1]	Byte	Octet 4
	Remote transport selector		
12 ... 13	TSelLength	UInt	Length of TSelector
14 ... 45	TSel	array [1..32] of Byte	Character array for TSAP name
	Local transport selector		
46 ... 47	TSelLength	UInt	Length of TSelector
48 ... 79	TSel	array [1..32] of Byte	Character array for TSAP name

11.2.8.7 TSEND_C and TRCV_C instructions

Version V4.1 of the S7-1200 CPU together with STEP 7 V13 SP1 extends the capability of the TSEND_C and TRCV_C instructions to use connection parameters with structures according to "TCON_IP_v4" and "TCON_IP_RFC".

For this reason, the S7-1200 supports two sets of TSEND_C and TRCV_C instructions:

- Legacy TSEND_C and TRCV_C instructions (Page 655): These TSEND_C and TRCV_C instructions existed prior to version V4.0 of the S7-1200 and only work with connection parameters with structures according to "TCON_Param".
- TSEND_C and TRCV_C instructions (Page 645): These TSEND_C and TRCV_C instructions provide all of the functionality of the legacy instructions, plus the ability to use connection parameters with structures according to "TCON_IP_v4" and "TCON_IP_RFC".

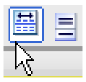
Selecting the version of the TSEND_C and TRCV_C instructions

There are two versions of the TSEND_C and TRCV_C instructions available in STEP 7:

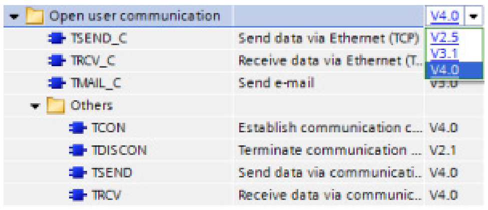
- Versions 2.5 and 3.1 were available in STEP 7 Basic/Professional V13 and earlier.
- Version 4.0 is available in STEP 7 Basic/Professional V13 SP1.

For compatibility and ease of migration, you can choose which instruction version to insert into your user program.

Do not use different instruction versions in the same CPU program.



Click the icon on the instruction tree task card to enable the headers and columns of the instruction tree.



Open user communication		V4.0
TSEND_C	Send data via Ethernet (TCP)	V2.5
TRCV_C	Receive data via Ethernet (T...	V3.1
TMAIL_C	Send e-mail	V4.0
Others		
TCON	Establish communication c...	V4.0
TDISCON	Terminate communication ...	V2.1
TSEND	Send data via communicati...	V4.0
TRCV	Receive data via communic...	V4.0

To change the version of the TSEND_C and TRCV_C instructions, select the version from the drop-down list. You can select the group or individual instructions.

When you use the instruction tree to place a TSEND_C or TRCV_C instruction in your program, a new FB or FC instance, depending on the TSEND_C or TRCV_C instruction selected, is created in the project tree. You can see new FB or FC instance in the project tree under PLC_x > Program blocks > System blocks > Program resources.

To verify the version of a TSEND_C or TRCV_C instruction in a program, you must inspect project tree properties and not the properties of a box displayed in the program editor. Select a project tree TSEND_C or TRCV_C FB or FC instance, right-click, select "Properties", and select the "Information" page to see the TSEND_C or TRCV_C instruction version number.

TSEND_C and TRCV_C (Send and receive data using Ethernet) instructions

The TSEND_C instruction combines the functions of the TCON, TDISCON and TSEND instructions. The TRCV_C instruction combines the functions of the TCON, TDISCON, and TRCV instructions. (Refer to "TCON, TDISCON, TSEND, AND TRCV (Page 664)" for more information on these instructions.)

The minimum size of data that you can transmit (TSEND_C) or receive (TRCV_C) is one byte; the maximum size is 8192 bytes. TSEND_C does not support the transmission of data from Boolean locations, and TRCV_C will not receive data into Boolean locations. For information on transferring data with these instructions, see the section on data consistency (Page 185).

Note

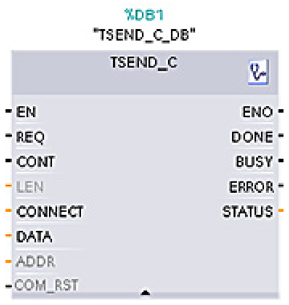
Initializing the communication parameters

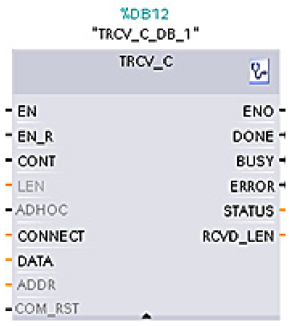
After you insert the TSEND_C or TRCV_C instruction, use the "Properties" of the instruction (Page 619) to configure the communication parameters (Page 641). As you enter the parameters for the communication partners in the inspector window, STEP 7 enters the corresponding data in the DB for the instruction.

If you want to use a multi-instance DB, you must manually configure the DB on both CPUs.

Table 11- 12 TSEND_C and TRCV_C instructions

LAD / FBD	SCL	Description

 %DB1
"TSEND_C_DB"
TSEND_C
EN
REQ
CONT
LEN
CONNECT
DATA
ADDR
COM_RST
ENO
DONE
BUSY
ERROR
STATUS
	"TSEND_C_DB"(
	req:=_bool_in_,
	cont:=_bool_in_,
	len:=_uint_in_,
	done=>_bool_out_,
	busy=>_bool_out_,
	error=>_bool_out_,
	status=>_word_out_,
	connect:=_struct_inout_,
	data:=_variant_inout_,
	com_rst:=_bool_inout_);
		TSEND_C establishes a TCP or ISO on TCP communication connection to a partner station, sends data, and can terminate the connection. After the connection is set up and established, it is automatically maintained and monitored by the CPU.

 %DB12
"TRCV_C_DB_1"
TRCV_C
EN
EN_R
CONT
LEN
ADHOC
CONNECT
DATA
ADDR
COM_RST
ENO
DONE
BUSY
ERROR
STATUS
RCVD_LEN
	"TRCV_C_DB"(
	en_r:=_bool_in_,
	cont:=_bool_in_,
	len:=_uint_in_,
	adhoc:=_bool_in_,
	done=>_bool_out_,
	busy=>_bool_out_,
	error=>_bool_out_,
	status=>_word_out_,
	rcvd_len=>_uint_out_,
	connect:=_struct_inout_,
	data:=_variant_inout_,
	com_rst:=_bool_inout_);
		TRCV_C establishes a TCP or ISO on TCP communication connection to a partner CPU, receives data, and can terminate the connection. After the connection is set up and established, it is automatically maintained and monitored by the CPU.

[1] STEP 7 automatically creates the DB when you insert the instruction.

Table 11- 13 TSEND_C and TRCV_C data types for the parameters

Parameter and type		Data type	Description
REQ (TSEND_C)	IN	Bool	Starts the send job on a rising edge
EN_R (TRCV_C)	IN	Bool	Receive enable
CONT	IN	Bool	Controls the communications connection: • 0: Establish connection automatically. Terminate communications connection after data is sent. • 1: Terminate communications connection after data has been received. The CONT parameter is only evaluated upon a positive edge at REQ or when COM_RST = "1".
LEN	IN	UDInt	Optional parameter (hidden) Maximum number of bytes to be sent (TSEND_C) or received (TRCV_C) with the job. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".

Parameter and type		Data type	Description
ADHOC (TRCV_C)	IN	Bool	Optional parameter (hidden) Ad hoc mode request for connection type TCP.
CONNECT	IN_OUT	TCON_Param	Pointer to the connection description corresponding to the structure of connection to be described. • For TCP or UDP, use the structure TCON_IP_v4 • For description, refer to: "Connection parameters with structure according to TCON_IP_v4". • For ISO-on-TCP, use the structure TCON_IP_RFC • For description, refer to: "Connection parameters with structure according to TCON_IP_RFC". The CONNECT parameter is only evaluated upon a positive edge at REQ (TSEND_C), when connection establishment starts (TRCV_C), or when COM_RST = 1.
DATA	IN_OUT	Variant	Pointer to the send area containing: • Address and length of data to be sent (TSEND_C) • Address and maximum length of received data (TRCV_C)
ADDR	IN_OUT	Variant	Optional parameter (hidden) Pointer to the address of the recipient with the connection type UDP. The address information is mapped in the structure TADDR_Param ###.
COM_RST	IN_OUT	Bool	Optional parameter (hidden) Restarts the instruction: • 0: Irrelevant • 1: Completely restarts the instruction; the existing connection is either terminated or reset and established again in accordance with CONT. The COM_RST parameter is reset after evaluation by the TSEND_C or TRCV_C instruction and should not, therefore, be switched statically.
DONE	OUT	Bool	Status parameter with the following values: • 0: Send job not yet started or is still executing. • 1: Send job executed without errors. This state is only displayed for one cycle.
BUSY	OUT	Bool	Status parameter with the following values: • 0: Send job not yet started or already completed. • 1: Send job not yet completed. A new send job cannot be started.
ERROR	OUT	Bool	Status parameters with the following values: • 0: No error • 1: Error occurred during connection establishment, data transmission, or connection termination.

Parameter and type		Data type	Description
STATUS	OUT	Word	Status of instruction (see the ERROR and STATUS parameters description).
RCVD_LEN (TRCV_C)	OUT	Int	Amount of data actually received (in bytes).

Note

The TSEND_C instruction requires a low-to-high transition at the REQ input parameter to start a send job. The BUSY parameter is then set to 1 during processing. Completion of the send job is indicated by either the DONE or ERROR parameters being set to 1 for one scan. During this time, any low-to-high transition at the REQ input parameter is ignored.

Note

The default setting of the LEN parameter (LEN = 0) uses the DATA parameter to determine the length of the data being transmitted. Ensure that the DATA transmitted by the TSEND_C instruction is the same size as the DATA parameter of the TRCV_C instruction.

TSEND_C operations

The TSEND_C instruction is executed asynchronously and implements the following functions in sequence:

1. Setting up and establishing a communications connection:

   TSEND_C sets up a communication connection and establishes this connection if a rising edge is detected at the REQ parameter and no communication connection is in place yet. Once the connection has been set up and established, it is automatically maintained and monitored by the CPU. The connection description specified at the CONNECT parameter is used to set up the communications connection. The following connection types can be used:

   – TCON_Param structure for the TCP, ISO-on-TCP, and UDP protocols

   – WIth V4.1, TCP/UDP: Connection description using the structure TCON_IP_v4 at the parameter CONNECT

   – WIth V4.1, ISO-on-TCP: Connection description using the structure TCON_IP_RFC at the parameter CONNECT

   An existing connection is terminated and the connection which has been set up is removed when the CPU goes into STOP mode. To set up and establish the connection again, you must execute TSEND_C again. For information on the number of possible communication connections, please refer to the technical specifications for your CPU.

2. Sending data via an existing communications connection:

   The send job is executed when a rising edge is detected at the REQ parameter. As described above, the communications connection is established first. You specify the send area with the DATA parameter. This includes the address and the length of the data to be sent. Do not use a data area with the data type BOOL or Array of BOOL at the DATA parameter. With the LEN parameter, you specify the maximum number of bytes sent with a send job. If using a symbolic name at the DATA parameter, the LEN parameter should have the value "0".

   The data to be sent must not be edited until the send job is completed.

3. Terminating the communications connection:

   The communications connection is terminated after the data has been sent if the CONT parameter had the value "0" at the time of the rising edge at the REQ parameter. Otherwise, the communications connection will be maintained.

If the send job executes successfully, the DONE parameter is set to "1". The communications connection may be terminated before this (see the above description of the dependency on the CONT parameter). Signal state "1" at the DONE parameter is not confirmation that the data sent has already been read by the communications partner.

TSEND_C is reset when the COM_RST parameter is set to "1". Data loss may occur if data is being transferred at this point.

The following scenarios are possible depending on the CONT parameter:

- CONT = "0":

  An existing communications connection is established.

- CONT = "1" and a communications connection was established:

  An existing communications connection is reset and established again.

- CONT = "1" and no communications connection was established.

  No communications connection is established.

The COM_RST parameter is reset following evaluation by the instruction T_SEND. To enable TSEND_C again after the execution (DONE = 1), call the instruction once with REQ = 0

TRCV_C operations

The TRCV_C instruction is executed asynchronously and implements the following functions in sequence:

1. Setting up and establishing a communications connection:

   TRCV_C sets up a communication connection and establishes this connection if the EN_R parameter = "1" and there is no communication connection. Once the connection has been set up and established, it is automatically maintained and monitored by the CPU.

   The connection description specified at the CONNECT parameter is used to set up the communications connection. The following connection types can be used:

   – TCON_Param structure for the TCP, ISO-on-TCP, and UDP protocols

   – With V4.1, TCP / UDP: Connection description via the structure TCON_IP_v4 at the parameter CONNECT

   – With V4.1, ISO-on-TCP: Connection description via the structure TCON_IP_RFC at the parameter CONNECT

   An existing connection is terminated and the connection which has been set up is removed when the CPU goes into STOP mode. To set up and establish the connection again, you must execute TRCV_C again with EN_R = "1".

   If EN_R is set to "0" before the communications connection has been established, the connection will be established and remain in place even if CONT = "0". However, no data will be received (DONE will remain "0").

   For information on the number of possible communication connections, please refer to the technical specifications for your CPU.

2. Receiving data via an existing communications connection:

   Receipt of data is enabled when the EN_R parameter is set to the value "1". As described above, the communications connection is established first. The received data is entered in a receive area. You specify the length of the receive area either with the LEN parameter (if LEN <> 0) or with the length information of the DATA parameter (if LEN = 0), depending on the protocol variant being used. If you use purely symbolic values at the DATA parameter, the LEN parameter must have the value "0".

   If EN_R is set to "0" before data is received for the first time, the communication connection will remain in place even if CONT = 0. However, no data will be received (DONE will remain "0").

3. Terminating the communications connection:

   The communications connection is terminated after data has been received if the CONT parameter had the value "0" when connection established was started. Otherwise, the communications connection will be maintained.

If the receive job executes successfully, the DONE parameter is set to "1". The communications connection may be terminated before this (see the above description of the dependency on the CONT parameter).

TRCV_C is reset when the COM_RST parameter is set. If data is being received when it executes again, this can lead to a loss of data. The following scenarios are possible depending on the CONT parameter:

- CONT = "0":

  An existing communications connection is established.
- CONT = "1" and a communications connection was established:

  An existing communications connection is reset and established again.
- CONT = "1" and no communications connection was established:

  No communications connection is established.

The COM_RST parameter is reset following evaluation by the instruction TRCV_".

TRCV_C handles the same receive modes as the TRCV instruction. The following table shows how data is entered in the receive area:

Protocol variant	Availability of data in the receive area	Connection_type parameter of the connection description	LEN parameter	RCVD_LEN parameter
TCP (Ad hoc mode)	The data is immediately available.	B#16#11	Selected with the TRCV_C instruction ADHOC input	1 to 1472
TCP (data receipt with specified length)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	B#16#11	1 to 8192	Identical to the value at the LEN parameter
ISO on TCP (protocol-controlled data transfer)	The data is available as soon as the data length specified at the LEN parameter has been fully received.	B#16#12	1 to 8192	Identical to the value at the LEN parameter

Note

Ad hoc mode

The "ad hoc mode" is only available with the TCP protocol variant. To configure the TRCV_C instruction for ad hoc mode, set the ADHOC instruction input parameter. The length of the receive area is defined by the pointer at the DATA parameter. The data length actually received is output at the RCVD_LEN parameter. A maximum of 1460 bytes can be received.

Note

Importing of S7-300/400 STEP 7 projects containing "ad hoc mode" into the S7-1200

In S7-300/400 STEP 7 projects, "ad hoc mode" is selected by assigning "0" to the LEN parameter. In the S7-1200, you configure the TRCV_C instruction for ad hoc mode by setting the ADHOC instruction input parameter..

If you import an S7-300/400 STEP 7 project containing "ad hoc mode" into the S7-1200, you must change the LEN parameter to "65535".

Note

TCP (data receipt with specified length)

You use the value of the LEN parameter to specify the length for the data receipt. The data specified at the DATA parameter is available in the receive area as soon as the length specified at the LEN parameter has been completely received.

Note

ISO on TCP (protocol-controlled data transfer)

With the ISO on TCP protocol variant, data is transferred protocol-controlled. The receive area is defined by the LEN and DATA parameters.

BUSY, DONE, and ERROR parameters

Note

Due to the asynchronous processing of TSEND_C, you must keep the data in the sender area consistent until the DONE parameter or the ERROR parameter assumes the value TRUE.

For TSEND_C, a TRUE state at the parameter DONE means that the data was sent successfully. It does not mean that the connection partner CPU actually read the receive buffer.

Due to the asynchronous processing of TRCV_C, the data in the receiver area are only consistent when parameter DONE = 1.

Table 11- 14 TSEND_C and TRCV_C instructions BUSY, DONE, and ERROR parameters

BUSY	DONE	ERROR	Description
1	0	0	The send job is being processed.
0	1	0	The send job was completed successfully.
0	0	1	The connection establishment or the send job was completed with an error. The cause of the error is specified in the STATUS parameter.
0	0	0	No new send job was assigned.

You can check the status of the execution with the BUSY, DONE, ERROR, and STATUS parameters. The BUSY parameter indicates the processing status. With the DONE parameter, you can check whether or not a send job executed successfully. The ERROR parameter is set when errors occurred during execution of TSEND_C or TRCV_C. The error information is output at the STATUS parameter.

Error and Status parameters

Table 11- 15 TSEND_C and TRCV_C condition codes for ERROR and STATUS

ERROR	STATUS (W#16#...)	Description
0	0000	Send (TSEND_C) or receive (TRCV_C) job executed without errors.
0	7000	No active send job execution; no communications connection established.
0	7001	• Start send (TSEND_C) or receive (TRCV_C) job execution. • Establish connection. • Wait for connection partner.
0	7002	Data is being sent (TSEND_C) or received (TRCV_C).
0	7003	Communications connection is being terminated.
0	7004	Communications connection established and monitored; no send (TSEND_C) or receive (TRCV_C) job execution active.
0	7005	Communications connection is being reset.
1	80A0	Group error for error codes W#16#80A1 and W#16#80A2.
1	80A1	• Connection or port already being used by user. • Communication error: – The specified connection has not yet been established. – The specified connection is being terminated. Transfer through this connection is not possible. – The interface is being re-initialized.
1	80A2	Local or remote port is being used by the system.
1	80A3	• Attempt being made to re-establish an existing connection. • Attempt being made to terminate a non-existent connection.
1	80A4	IP address of the remote endpoint of the connection is invalid, which means it corresponds to the IP address of the local partner.
1	80A7	Communication error: You called the instruction with COM_RST = 1 before the send job was complete.
1	80B2	The CONNECT parameter points to a data block that was generated with the attribute "Only store in load memory".
1	80B3	Inconsistent parameter assignment: Group error for error codes W#16#80A0 to W#16#80A2, W#16#80A4, W#16#80B4 to W#16#80B9.
1	80B4	You have violated one or both of the following conditions for passive connection establishment (active_est = FALSE) when using the ISO on TCP protocol variant (connection_type = B#16#12): • local_tsap_id_len >= B#16#02 • local_tsap_id[1] = B#16#E0
1	80B5	Only passive connection establishment is permitted for connection type 13 = UDP.
1	80B6	Parameter assignment error in the connection_type parameter of the data block for connection description.
1	80B7	Error in one of the following parameters of the data block for connection description: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.

ERROR	STATUS (W#16#...)	Description
1	8085	• The LEN parameter is larger than the highest permitted value. • The value at the LEN or DATA parameter was changed after the first call.
1	8086	The ID parameter within the CONNECT parameter is outside the permitted range.
1	8087	Maximum number of connections reached; no additional connection possible.
1	8088	The value at the LEN parameter does not correspond to the receive area set at the DATA parameter.
1	8089	The CONNECT parameter does not point to a data block.
1	8091	Maximum nesting depth exceeded.
1	809A	The CONNECT parameter points to a field that does not correspond to the length of the connection description.
1	809B	The ID of the local device in the connection description does not correspond to the CPU.
1	80C3	• All connection (Page 637) resources are in use. • A block with this ID is already being processed in a different priority group.
1	80C4	Temporary communication error: • The connection cannot be established at this time. • The interface is receiving new parameters or the connection is being established. • The configured connection is being removed by a "TDISCON (Page 664)" instruction. • The connection used is being terminated by a call with COM_RST = 1.
1	8722	Error in the CONNECT parameter: Invalid source area (area not declared in data block).
1	873A	Error in the CONNECT parameter: Access to connection description is not possible (no access to data block).
1	877F	Error in the CONNECT parameter: Internal error
1	8822	TSEND_C: DATA parameter: Invalid source area, the area does not exist in the DB.
1	8824	TSEND_C: DATA parameter: Area error in the VARIANT pointer.
1	8832	TSEND_C: DATA parameter: The DB number is too high.
1	883A	TSEND_C: CONNECT parameter: Access to specified connection data not possible (for example, because the DB does not exist).
1	887F	TSEND_C: DATA parameter: Internal error (for example, invalid VARIANT reference)
1	893A	TSEND_C: DATA parameter: Access to send area not possible (for example, because the DB does not exist).
1	8922	TRCV_C: DATA parameter: Invalid target area; the area does not exist in the DB.
1	8924	TRCV_C: DATA parameter: Area error in the VARIANT pointer.
1	8932	TRCV_C: DATA parameter: The DB number is too high.
1	893A	TRCV_C: CONNECT parameter: Access to specified connection data not possible (for example, because the DB does not exist).
1	897F	TRCV_C: DATA parameter: Internal error (for example, invalid VARIANT reference).
1	8A3A	TRCV_C: DATA parameter: No access to the data area (for example because the data block does not exist).

Note

Error messages of the instructions TCON, TSEND, TRCV, and TDISCON

Internally, the TSEND_C instruction uses the TCON, TSEND, and TDISCON instructions; and the TRCV_C instruction uses the TCON, TRCV, and TDISCON instructions. Refer to "TCON, TDISCON, TSEND, AND TRCV (Page 664)" for more information on error messages of these instructions.

Connection Ethernet protocols

Every CPU has an integrated PROFINET port, which supports standard PROFINET communications. The TSEND_C and TRCV_C and TSEND and TRCV instructions all support the TCP and ISO on TCP Ethernet protocols.

Refer to "Device Configuration: Configuring the Local/Partner connection path (Page 619)" for more information.

11.2.8.8 Legacy TSEND_C and TRCV_C instructions

Prior to the release of STEP 7 V13 SP1 and the S7-1200 V4.1 CPUs, the TSEND_C and TRCV_C instructions could only work with connection parameters with structures according to "TCON_Param". The general concepts apply to both sets of instructions. Refer to the individual legacy TSEND_C and TRCV_C instructions for programming information.

Selecting the version of the TSEND_C and TRCV_C instructions

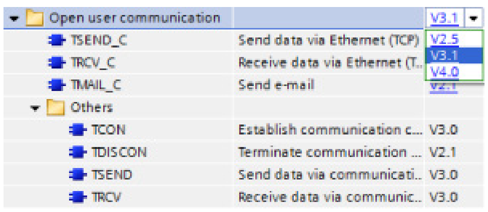
There are two versions of the TSEND_C and TRCV_C instructions available in STEP 7:

- Versions 2.5 and 3.1 were available in STEP 7 Basic/Professional V13 and earlier.
- Version 4.0 is available in STEP 7 Basic/Professional V13, SP1.

For compatibility and ease of migration, you can choose which instruction version to insert into your user program.

Do not use different instruction versions in the same CPU program.

Click the icon on the instruction tree task card to enable the headers and columns of the instruction tree.



Open user communication		V3.1
TSEND_C	Send data via Ethernet (TCP)	V2.5
TRCV_C	Receive data via Ethernet (T...	V3.1
TMAIL_C	Send e-mail	V4.0
Others		
TCON	Establish communication c...	V3.0
TDISCON	Terminate communication ...	V2.1
TSEND	Send data via communicati...	V3.0
TRCV	Receive data via communic...	V3.0

To change the version of the TSEND_C and TRCV_C instructions, select the version from the drop-down list. You can select the group or individual instructions.

When you use the instruction tree to place a TSEND_C or TRCV_C instruction in your program, a new FB or FC instance, depending on the TSEND_C or TRCV_C instruction selected, is created in the project tree. You can see new FB or FC instance in the project tree under PLC_x > Program blocks > System blocks > Program resources.

To verify the version of a TSEND_C or TRCV_C instruction in a program, you must inspect project tree properties and not the properties of a box displayed in the program editor. Select a project tree TSEND_C or TRCV_C FB or FC instance, right-click, select "Properties", and select the "Information" page to see the TSEND_C or TRCV_C instruction version number.

Legacy TSEND_C and TRCV_C (Send and receive data using Ethernet) instructions

The legacy TSEND_C instruction combines the functions of the legacy TCON, TDISCON and TSEND instructions. The TRCV_C instruction combines the functions of the TCON, TDISCON, and TRCV instructions. (Refer to "Legacy TCON, TDISCON, TSEND, and TRCV (TCP communication) instructions (Page 673)" for more information on these instructions.)

The minimum size of data that you can transmit (TSEND_C) or receive (TRCV_C) is one byte; the maximum size is 8192 bytes. TSEND_C does not support the transmission of data from Boolean locations, and TRCV_C will not receive data into Boolean locations. For information on transferring data with these instructions, see the section on data consistency (Page 185).

Note

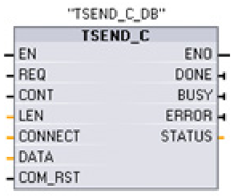
Initializing the communication parameters

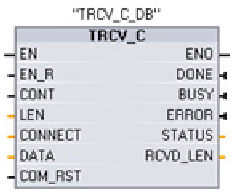
After you insert the TSEND_C or TRCV_C instruction, use the "Properties" of the instruction (Page 619) to configure the communication parameters (Page 641). As you enter the parameters for the communication partners in the inspector window, STEP 7 enters the corresponding data in the DB for the instruction.

If you want to use a multi-instance DB, you must manually configure the DB on both CPUs.

Table 11- 16 TSEND_C and TRCV_C instructions

LAD / FBD	SCL	Description

 "TSEND_C_DB"
TSEND_C
EN
REQ
CONT
LEN
CONNECT
DATA
COM_RST
ENO
DONE
BUSY
ERROR
STATUS
	`"TSEND_C_DB"(`
	`req:=_bool_in_,`
	`cont:=_bool_in_,`
	`len:=_uint_in_,`
	`done=>_bool_out_,`
	`busy=>_bool_out_,`
	`error=>_bool_out_,`
	`status=>_word_out_,`
	`connect:=_struct_inout_,`
	`data:=_variant_inout_,`
	`com_rst:=_bool_inout_);`
		TSEND_C establishes a TCP or ISO on TCP communication connection to a partner station, sends data, and can terminate the connection. After the connection is set up and established, it is automatically maintained and monitored by the CPU.

 "TRCV_C_DB"
TRCV_C
EN
EN_R
CONT
LEN
CONNECT
DATA
COM_RST
ENO
DONE
BUSY
ERROR
STATUS
RCVD_LEN
	`"TRCV_C_DB"(`
	`en_r:=_bool_in_,`
	`cont:=_bool_in_,`
	`len:=_uint_in_,`
	`adhoc:=_bool_in_,`
	`done=>_bool_out_,`
	`busy=>_bool_out_,`
	`error=>_bool_out_,`
	`status=>_word_out_,`
	`rcvd_len=>_uint_out_,`
	`connect:=_struct_inout_,`
	`data:=_variant_inout_,`
	`com_rst:=_bool_inout_);`
		TRCV_C establishes a TCP or ISO on TCP communication connection to a partner CPU, receives data, and can terminate the connection. After the connection is set up and established, it is automatically maintained and monitored by the CPU.

[1] STEP 7 automatically creates the DB when you insert the instruction.

Table 11- 17 TSEND_C and TRCV_C data types for the parameters

Parameter and type		Data type	Description
REQ (TSEND_C)	IN	Bool	REQ = 1 starts the TSEND_C send job on a rising edge with the connection descricribed in CONNECT parameter. (CONT = 1 is also required to establish and maintain the communication connection.
EN_R (TRCV_C)	IN	Bool	When EN_R = 1, TRCV_C is ready to receive. The receive job is processed. (CONT = 1 is also required to establish and maintain the communication connection.)
CONT	IN	Bool	Controls the communication connection: • 0: Disconnect the communication connection • 1: Establish and maintain the communication connection When sending data (TSEND_C) (rising edge at the REQ parameter), the CONT parameter must have the value TRUE in order to establish or maintain a connection. When receiving data (TRCV_C) (rising edge at the EN_R parameter), the CONT parameter must have the value TRUE in order to establish or maintain a connection.

Parameter and type		Data type	Description
LEN	IN	UInt	Maximum number of bytes to be sent (TSEND_C) or received (TRCV_C): • Default = 0: The DATA parameter determines the length of the data to be sent (TSEND_C) or received (TRCV_C). • Ad hoc mode = 65535: A variable length of data is set for reception (TRCV_C).
CONNECT	IN_OUT	TCON_Param	Pointer to the connection description (Page 641)
DATA	IN_OUT	Variant	• Contains address and length of data to be sent (TSEND_C) • Contains start address and maximum length of received data (TRCV_C).
COM_RST	IN_OUT	Bool	Allows restart of the instruction: • 0: Irrelevant • 1: Complete restart of the function block, existing connection will be terminated.
DONE	OUT	Bool	• 0: Job is not yet started or still running. • 1: Job completed without error.
BUSY	OUT	Bool	• 0: Job is completed. • 1: Job is not yet completed. A new job cannot be triggered.
ERROR	OUT	Bool	Status parameters with the following values: • 0: No error • 1: Error occurred during processing. STATUS provides detailed information on the type of error.
STATUS	OUT	Word	Status information including error information. (Refer to the "Error and Status Parameters" table below.)
RCVD_LEN (TRCV_C)	OUT	Int	Amount of data actually received, in bytes

Note

The TSEND_C instruction requires a low-to-high transition at the REQ input parameter to start a send job. The BUSY parameter is then set to 1 during processing. Completion of the send job is indicated by either the DONE or ERROR parameters being set to 1 for one scan. During this time, any low-to-high transition at the REQ input parameter is ignored.

Note

The default setting of the LEN parameter (LEN = 0) uses the DATA parameter to determine the length of the data being transmitted. Ensure that the DATA transmitted by the TSEND_C instruction is the same size as the DATA parameter of the TRCV_C instruction.

TSEND_C operations

The following functions describe the operation of the TSEND_C instruction:

- To establish a connection, execute TSEND_C with CONT = 1.
- After successful establishing of the connection, TSEND_C sets the DONE parameter for one cycle.
- To terminate the communication connection, execute TSEND_C with CONT = 0. The connection will be aborted immediately. This also affects the receiving station. The connection will be closed there and data inside the receive buffer could be lost.
- To send data over an established connection, execute TSEND_C with a rising edge on REQ. After a successful send operation, TSEND_C sets the DONE parameter for one cycle.
- To establish a connection and send data, execute TSEND_C with CONT =1 and REQ = 1. After a successful send operation, TSEND_C sets the DONE parameter for one cycle.

TRCV_C operations

The following functions describe the operation of the TRCV_C instruction:

- To establish a connection, execute TRCV_C with parameter CONT = 1.
- To receive data, execute TRCV_C with parameter EN_R = 1. TRCV_C receives the data continuously when parameters EN_R = 1 and CONT = 1.
- To terminate the connection, execute TRCV_C with parameter CONT = 0. The connection will be aborted immediately, and data could be lost.

TRCV_C handles the same receive modes as the TRCV instruction. The following table shows how data is entered in the receive area:

Table 11- 18 Entering the data into the receive area

Protocol variant	Entering the data in the receive area	Parameter "connection_type"	Value of the LEN parameter	Value of the RCVD_LEN parameter (bytes)
TCP	Ad hoc mode	B#16#11	65535	1 to 1472
TCP	Data reception with specified length	B#16#11	0 (recommended) or 1 to 8192, except 65535	1 to 8192
ISO on TCP	Ad hoc mode	B#16#12	65535	1 to 1472
ISO on TCP	Protocol-controlled	B#16#12	0 (recommended) or 1 to 8192, except 65535	1 to 8192

Note

Ad hoc mode

The "ad hoc mode" exists with the TCP and ISO on TCP protocol variants. You set "ad hoc mode" by assigning "65535" to the LEN parameter. The receive area is identical to the area formed by DATA. The length of the received data will be output to the parameter RCVD_LEN.

If you store the data in an "optimized" DB (symbolic only), you can receive data only in arrays of Byte, Char, USInt, and SInt data types.

Note

Importing of S7-300/400 STEP 7 projects containing "ad hoc mode" into the S7-1200

In S7-300/400 STEP 7 projects, "ad hoc mode" is selected by assigning "0" to the LEN parameter. In the S7-1200, you set "ad hoc mode" by assigning "65535" to the LEN parameter.

If you import an S7-300/400 STEP 7 project containing "ad hoc mode" into the S7-1200, you must change the LEN parameter to "65535".

Note

Must keep the data in the sender area consistent until the DONE parameter or the ERROR parameter assumes the value TRUE

Due to the asynchronous processing of TSEND_C, you must keep the data in the sender area consistent until the DONE parameter or the ERROR parameter assumes the value TRUE.

For TSEND_C, a TRUE state at the parameter DONE means that the data was sent successfully. It does not mean that the connection partner CPU actually read the receive buffer.

Due to the asynchronous processing of TRCV_C, the data in the receiver area are only consistent when parameter DONE = 1.

Table 11- 19 TSEND_C and TRCV_C instructions BUSY, DONE, and ERROR parameters

BUSY	DONE	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	The job is successfully completed.
FALSE	FALSE	TRUE	The job was ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	A new job was not assigned.

TSEND_C and TRCV_C Error and Status condition codes

ERROR	STATUS	Description
0	0000	Job executed without error
0	7000	No job processing active
0	7001	Start job processing, establishing connection, waiting for connection partner
0	7002	Data being sent or received
0	7003	Connection being terminated
0	7004	Connection established and monitored, no job processing active
1	8085	LEN parameter is greater than the largest permitted value.
1	8086	The CONNECT parameter is outside the permitted range.
1	8087	Maximum number of connections reached; no additional connection possible.
1	8088	LEN parameter is not valid for the memory area specified in DATA.
1	8089	The CONNECT parameter does not point to a data block.
1	8091	Maximum nesting depth exceeded.
1	809A	The CONNECT parameter points to a field that does not match the length of the connection description.
1	809B	The local_device_id in the connection description does not match the CPU.
1	80A1	Communications error: • The specified connection was not yet established • The specified connection is currently being terminated; transmission over this connection is not possible • The interface is being reinitialized
1	80A3	Attempt being made to terminate a nonexistent connection
1	80A4	IP address of the remote partner connection is invalid. For example, the remote partner IP address is the same as the local partner IP address.
1	80A5	Connection ID (Page 637) is already in use.
1	80A7	Communications error: You called TDISCON before TSEND_C was complete.
1	80B2	The CONNECT parameter points to a data block that was generated with the keyword UNLINKED.
1	80B3	Inconsistent parameters: • Error in the connection description • Local port (parameter local_tsap_id) is already present in another connection description. • ID in the connection description different from the ID specified as parameter

ERROR	STATUS	Description
1	80B4	When using the ISO on TCP (connection_type = B#16#12) to establish a passive connection, condition code 80B4 alerts you that the TSAP entered did not conform to one of the following address requirements: • For a local TSAP length of 2 and a TSAP ID value of either E0 or E1 (hexadecimal) for the first byte, the second byte must be either 00 or 01. • For a local TSAP length of 3 or greater and a TSAP ID value of either E0 or E1 (hexadecimal) for the first byte, the second byte must be either 00 or 01 and all other bytes must be valid ASCII characters. • For a local TSAP length of 3 or greater and the first byte of the TSAP ID does not have a value of either E0 or E1 (hexadecimal), then all bytes of the TSAP ID must be valid ASCII characters. Valid ASCII characters are byte values from 20 to 7E (hexadecimal).
1	80B7	Data type and/or length of the transmitted data does not fit in the area in the partner CPU in which it is to be written.
1	80C3	All connection resources are in use.
1	80C4	Temporary communications error: • The connection cannot be established at this time • The interface is receiving new parameters • The configured connection is currently being removed by a TDISCON.
1	8722	CONNECT parameter: Source area invalid: area does not exist in DB.
1	873A	CONNECT parameter: Access to connection description is not possible (for example, DB not available)
1	877F	CONNECT parameter: Internal error such as an invalid ANY reference
1	893A	Parameter contains the number of a DB that is not loaded.

Connection Ethernet protocols

Every CPU has an integrated PROFINET port, which supports standard PROFINET communications. The TSEND_C and TRCV_C and TSEND and TRCV instructions all support the TCP and ISO on TCP Ethernet protocols.

Refer to "Device Configuration: Configuring the Local/Partner connection path (Page 619)" for more information.

11.2.8.9 TCON, TDISCON, TSEND, and TRCV instructions

Version V4.1 of the S7-1200 CPU together with STEP 7 V13 SP1 extends the capability of the TCON, TDISCON, TSEND, and TRCV instructions to use connection parameters with structures according to "TCON_IP_v4" and "TCON_IP_RFC".

For this reason, the S7-1200 supports two sets of TCON, TDISCON, TSEND, and TRCV instructions:

- Legacy TCON, TDISCON, TSEND, and TRCV instructions (Page 673): These TCON, TDISCON, TSEND, and TRCV instructions existed prior to version V4.0 of the S7-1200 and only work with connection parameters with structures according to "TCON_Param".
- TCON, TDISCON, TSEND, and TRCV instructions (Page 664): These TCON, TDISCON, TSEND, and TRCV instructions provide all of the functionality of the legacy instructions, plus the ability to use connection parameters with structures according to "TCON_IP_v4" and "TCON_IP_RFC".

Selecting the version of the TCON, TDISCON, TSEND, and TRCV instructions

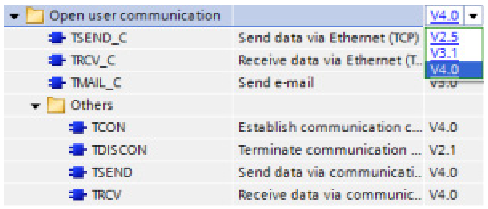
There are two versions of the TCON, TDISCON, TSEND, or TRCV instructions available in STEP 7:

- Versions 2.5 and 3.1 were available in STEP 7 Basic/Professional V13 and earlier.
- Version 4.0 is available in STEP 7 Basic/Professional V13 SP1.

For compatibility and ease of migration, you can choose which instruction version to insert into your user program.

Do not use different instruction versions in the same CPU program.

Click the icon on the instruction tree task card to enable the headers and columns of the instruction tree.



Open user communication		V4.0 (V2.5, V3.1, V4.0)
TSEND_C	Send data via Ethernet (TCP)	
TRCV_C	Receive data via Ethernet (T...	
TMAIL_C	Send e-mail	V3.0
Others		
TCON	Establish communication c...	V4.0
TDISCON	Terminate communication ...	V2.1
TSEND	Send data via communicati...	V4.0
TRCV	Receive data via communic...	V4.0

To change the version of the TCON, TDISCON, TSEND, or TRCV instructions, select the version from the drop-down list. You can select the group or individual instructions.

When you use the instruction tree to place a TCON, TDISCON, TSEND, or TRCV instruction in your program, a new FB or FC instance, depending on the TCON, TDISCON, TSEND, or TRCV instruction selected, is created in the project tree. You can see new FB or FC instance in the project tree under PLC_x > Program blocks > System blocks > Program resources.

To verify the version of a TCON, TDISCON, TSEND, or TRCV instruction in a program, you must inspect project tree properties and not the properties of a box displayed in the program editor. Select a project tree TCON, TDISCON, TSEND, or TRCV FB or FC instance, right-click, select "Properties", and select the "Information" page to see the TCON, TDISCON, TSEND, or TRCV instruction version number.

TCON, TDISCON, TSEND, and TRCV (TCP communication) instructions

Ethernet communication using TCP and ISO on TCP protocols

Note

TSEND_C and TRCV_C instructions

To help simplify the programming of PROFINET/Ethernet communication, the TSEND_C instruction and the TRCV_C instruction combine the functionality of the TCON, TDISCON. TSEND and TRCV instructions:

- TSEND_C combines the TCON, TDISCON and TSEND instructions.
- TRCV_C combines the TCON, TDISCON and TRCV instructions.

The following instructions control the communication process:

- TCON establishes the TCP/IP connection between the client and server (CPU) PC.
- TSEND and TRCV send and receive data.
- TDISCON breaks the connection.

The minimum size of data that you can transmit (TSEND) or receive (TRCV) is one byte; the maximum size is 8192 bytes. TSEND does not support the transmission of data from Boolean locations, and TRCV will not receive data into Boolean locations. For information transferring data with these instructions, see the section on data consistency (Page 185).

TCON, TDISCON, TSEND, and TRCV operate asynchronously, which means that the job processing extends over multiple instruction executions. For example, you start a job for setting up and establishing a connection by executing an instruction TCON with parameter REQ = 1. Then you use additional TCON executions to monitor the job progress and test for job completion with parameter DONE.

The following table shows the relationships between BUSY, DONE, and ERROR. Use the table to determine the current job status:

Table 11- 20 Interactions between the BUSY, DONE, and ERROR parameters

BUSY	DONE	ERROR	Description
1	0	0	The job is being processed.
0	1	0	The job successfully completed.
0	0	1	The job ended with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new job assigned.

TCON and TDISCON

Note

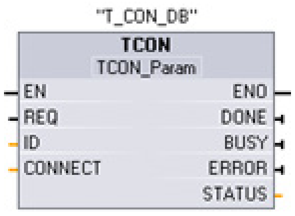
Initializing the communication parameters

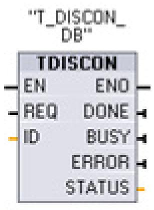
After you insert the TCON instruction, use the "Properties" of the instruction (Page 619) to configure the communication parameters (Page 641). As you enter the parameters for the communication partners in the inspector window, STEP 7 enters the corresponding data in the instance DB for the instruction.

If you want to use a multi-instance DB, you must manually configure the DB on both CPUs.

Table 11- 21 TCON and TDISCON instructions

LAD / FBD		Description

 "T_CON_DB"
TCON
TCON_Param
EN
REQ
ID
CONNECT
ENO
DONE
BUSY
ERROR
STATUS
	`"TCON_DB"(`
	`req:=_bool_in_,`
	`ID:=_undef_in_,`
	`done=>_bool_out_,`
	`busy=>_bool_out_,`
	`error=>_bool_out_,`
	`status=>_word_out_,`
	`connect:=_struct_inout_);`
		TCP and ISO on TCP: TCON initiates a communications connection from the CPU to a communication partner.

 "T_DISCON_DB"
TDISCON
EN
REQ
ID
ENO
DONE
BUSY
ERROR
STATUS
	`"TDISCON_DB"(`
	`req:=_bool_in_,`
	`ID:=_word_in_,`
	`done=>_bool_out_,`
	`busy=>_bool_out_,`
	`error=>_bool_out_,`
	`status=>_word_out_);`
		TCP and ISO on TCP: TDISCON terminates a communications connection from the CPU to a communication partner.

[1] STEP 7 automatically creates the DB when you insert the instruction.

Table 11- 22 Data types for the parameters of TCON and TDISCON

Parameter	Declaration	Data type	Description
REQ	IN	Bool	Starts the job to establish the connection specified in the ID upon a rising edge.
ID	IN	CONN_OUC (Word)	Reference to the assigned connection. Range of values: W#16#0001 to W#16#0FFF

Parameter	Declaration	Data type	Description
CONNECT (TCON)	IN_OUT	VARIANT	Pointer to the connection description • For TCP or UDP, use the structure TCON_IP_v4 For description, refer to: "Connection parameters with structure according to TCON_IP_v4" in the TIA Portal. • For ISO-on-TCP, use the structure TCON_IP_RFC For description, refer to: "Connection parameters with structure according to TCON_IP_RFC" in the TIA Portal.
DONE	OUT	Bool	Status parameter with the following values: • 0: Job not yet started or still in progress • 1: Job executed without errors
BUSY	OUT	Bool	Status parameter with the following values: • 0: Job not yet started or already completed • 1: Job not yet completed. A new job cannot be started
ERROR	OUT	Bool	Status parameter ERROR: • 0: No error • 1: Error occurred
STATUS	OUT	Word	Status of the instruction

Both communication partners execute the TCON instruction to set up and establish the communication connection. You use parameters to specify the active and passive communication end point partners. After the connection is set up and established, it is automatically maintained and monitored by the CPU.

If the connection is terminated due to a line break or due to the remote communications partner, for example, the active partner attempts to re-establish the configured connection. You do not have to execute TCON again.

An existing connection is terminated and the set-up connection is removed when the TDISCON instruction is executed or when the CPU has gone into STOP mode. To set up and re-establish the connection, you must execute TCON again.

Table 11- 23 ERROR and STATUS condition codes for TCON and TDISCON

ERROR	STATUS (W#16#...)	Explanation
0	0000	Connection successfully established.
0	7000	No job processing active
0	7001	Start job execution; establish connection (TCON) or terminate connection (TDISCON).
0	7002	Connection is being established (REQ irrelevant); establish connection (TCON) or terminate connection (TDISCON).
1	8085	TCON: Connection ID is already in use.
1	8086	TCON: The ID parameter is outside the valid range.
1	8087	TCON: Maximum number of connections reached; no additional connection possible
1	8089	TCON: The CONNECT parameter does not point to a connection description or the connection description was created manually.

ERROR	STATUS (W#16#...)	Explanation
1	809A	TCON: The structure at the CONNECT parameter is not supported or the length is invalid.
1	809B	TCON: The ID of the local device in the connection description does not correspond to the CPU or the CP, or it is "0".
1	80A0	Group error for error codes W#16#80A1 and W#16#80A2.
1	80A1	TCON: For TCP/UDP (TCON_IP_v4): Connection or port is already is use.
1	80A2	TCON: Local or remote port is being used by the system.
1	80A3	TCON: Value at the ID parameter is already being used by a connection (TCON) that was created using the user program. The connection uses the identical ID, but different connection settings at the parameter CONNECT.
1	80A4	TCON: IP address of the remote endpoint of the connection is invalid or it corresponds to the IP address of the local partner.
1	80A5	TCON: Connection ID is already in use.
1	80A7	TCON: Communication error: You executed "TDISCON" before "TCON" had completed.
1	80B2	TCON: The CONNECT parameter points to a data block that was generated with the attribute "Only store in load memory".
1	80B3	Inconsistent parameter assignment: Group error for error codes W#16#80A0 to W#16#80A2, W#16#80A4, W#16#80B4 to W#16#80B9.
1	80B4	TCON: Only with TCON_IP_RFC The local T selector was not specified or the first byte does not contain the value 0x0E or the local T selector starts with "SIMATIC-".
1	80B5	TCON: Only passive connection establishment is permitted for connection type 13 = UDP (Parameter active_est of the structure TCON_IP_v4 has the value TRUE)..
1	80B6	TCON: Parameter assignment error in the connection_type parameter of the data block for connection description. • Only valid with TCON_IP_v4: 0x11, 0x0B and 0x13. • Only valid with TCON_IP_RFC: 0x0C and 0x12
1	80B7	TCON: With TCON_IP_v4: • TCP (active connection establishment): Remote port is "0". • TCP (passive connection establishment): Local port is "0". • UDP: Local port is "0". TCON: With TCON_IP_RFC: • Local (local_tselector) or remote (remote_tselector) T selector was specified with a length of more than 32 bytes. • For TSelLength of the T selector (local or remote), a length greater than 32 was entered. • Error in the length of the IP address of the specific connection partner.
1	80B8	TCON: Parameter ID in the local connection description (structure at parameter CONNECT) and parameter ID of the instruction are different.
1	80C3	TCON: All connection (Page 637) resources are in use.
1	80C4	Temporary communication error: • The connection cannot be established at this time (TCON). • The interface is currently receiving new parameters (TCON and TDISCON). • The configured connection is currently being removed by a "TDISCON" instruction (TCON).
1	80C5	TCON: The remote partner refuses to establish the connection, has terminated the connection or actively ended it.

ERROR	STATUS (W#16#...)	Explanation
1	80C6	TCON: The remote partner cannot be reached (network error).
1	80C7	TCON: Execution timeout.
1	80C8	TCON: ID is used by a connection created by the user program, which uses the same connection description at the CONNECT parameter.
1	80C9	TCON: Validation of the remote partner failed. The remote partner that wants to establish the connection does not match the defined partner of the structure at the CONNECT parameter.
1	80CE	TCON: The IP address of the local interface is 0.0.0.0.

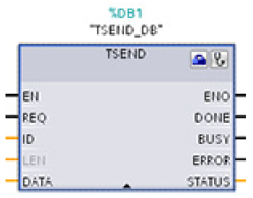
TSEND and TRCV

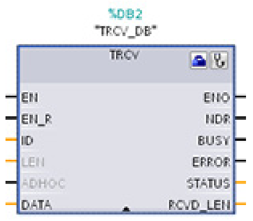
Note

When using PROFINET Open User communication, if you execute a TSEND instruction without a corresponding TRCV instruction executing on the remote device, then the TSEND instruction may reside indefinitely in a "Busy State", waiting for the TRCV instruction to receive the data. In this state, the TSEND instruction "Busy" output is set, and the "Status" output has a value of "0x7002". This condition may occur if you are transferring more than 4096 bytes of data. The issue is resolved at the next execution of the TRCV instruction.

Table 11- 24 TSEND and TRCV instructions

LAD / FBD	SCL	Description

 %DB1
"TSEND_DB"
TSEND
EN
REQ
ID
LEN
DATA
ENO
DONE
BUSY
ERROR
STATUS
	`"TSEND_DB"(`
	`req:=_bool_in_,`
	`ID:=_word_in_,`
	`len:=_udint_in_,`
	`done=>_bool_out_,`
	`busy=>_bool_out_,`
	`error=>_bool_out_,`
	`status=>_word_out_,`
	`data:=_variant_inout_);`
		TCP and ISO on TCP: TSEND sends data through a communication connection from the CPU to a partner station.

 %DB2
"TRCV_DB"
TRCV
EN
EN_R
ID
LEN
ADHOC
DATA
ENO
NDR
BUSY
ERROR
STATUS
RCVD_LEN
	`"TRCV_DB"(`
	`en_r:=_bool_in_,`
	`ID:=_word_in_,`
	`len:=_udint_in_,`
	`adhoc:=_bool_in_,`
	`ndr=>_bool_out_,`
	`busy=>_bool_out_,`
	`error=>_bool_out_,`
	`status=>_word_out_,`
	`rcvd_len=>_udint_out_,`
	`data:=_variant_inout_);`
		TCP and ISO on TCP: TRCV receives data through a communication connection from a partner station to the CPU.

1 STEP 7 automatically creates the DB when you insert the instruction.

Table 11- 25 Data types for the parameters of TSEND and TRCV

Parameter and type		Data type	Description
REQ	IN	Bool	TSEND: Starts the send job on a rising edge. The data is transferred from the area specified by DATA and LEN.
EN_R	IN	Bool	TRCV: Enables the CPU to receive; with EN_R = 1, the TRCV is ready to receive. The receive job is processed.
ID	IN	CONN_OUC (Word)	Reference to the associated connection. ID must be identical to the associated parameter ID in the local connection description. Value range: W#16#0001 to W#16#0FFF
LEN	IN	UDInt	Maximum number of bytes to be sent (TSEND) or received (TRCV): • Default = 0: The DATA parameter determines the length of the data to be sent (TSEND) or received (TRCV). • Ad hoc mode = 65535: A variable length of data is set for reception (TRCV).
ADHOC	IN	Bool	TRCV: Optional parameter (hidden) Ad hoc mode request for connection type TCP.
DATA	IN_OUT	Variant	Pointer to send (TSEND) or receive (TRCV) data area; data area contains the address and length. The address refers to I memory, Q memory, M memory, or a DB.
DONE	OUT	Bool	TSEND: • 0: Job not yet started or still running. • 1: Job executed without error.
NDR	OUT	Bool	TRCV: • NDR = 0: Job not yet started or still running. • NDR = 1: Job successfully completed.
BUSY	OUT	Bool	• BUSY = 1: The job is not yet complete. A new job cannot be triggered. • BUSY = 0: Job is complete.
ERROR	OUT	Bool	ERROR = 1: Error occurred during processing. STATUS provides detailed information on the type of error
STATUS	OUT	Word	Status information including error information. (Refer to the Error and Status condition codes in the table below.)
RCVD_LEN	OUT	UDInt	TRCV: Amount of data actually received in bytes

Note

The TSEND instruction requires a low-to-high transition at the REQ input parameter to start a send job. The BUSY parameter is then set to 1 during processing. Completion of the send job is indicated by either the DONE or ERROR parameters being set to 1 for one scan. During this time, any low-to-high transition at the REQ input parameter is ignored.

TRCV Operations

The TRCV instruction writes the received data to a receive area that is specified by the following two variables:

- Pointer to the start of the area
- Length of the area or the value supplied at the LEN input if not 0

Note

The default setting of the LEN parameter (LEN = 0) uses the DATA parameter to determine the length of the data being transmitted. Ensure that the DATA transmitted by the TSEND instruction is the same size as the DATA parameter of the TRCV instruction.

As soon as all the job data has been received, TRCV transfers it to the receive area and sets NDR to 1.

Table 11- 26 Entering the data into the receive area

Protocol variant	Entering the data in the receive area	Parameter "connection_type"	Value of the LEN parameter	Value of the RCVD_LEN parameter (bytes)
TCP	Ad hoc mode	B#16#11	Selected with the TRCV instruction ADHOC input	1 to 1472
TCP	Data reception with specified length	B#16#11	0 (recommended) or 1 to 8192, except 65535	1 to 8192
ISO on TCP	Ad hoc mode	B#16#12	65535	1 to 1472
ISO on TCP	protocol-controlled	B#16#12	0 (recommended) or 1 to 8192, except 65535	1 to 8192

Note

Ad hoc mode

The "ad hoc mode" exists with the TCP and ISO on TCP protocol variants. To configure the TRCV instruction for ad hoc mode, set the ADHOC instruction input parameter. The receive area is identical to the area formed by DATA. The length of the received data will be output to the parameter RCVD_LEN. Immediately after receiving a block of data, TRCV enters the data in the receive area and sets NDR to 1.

If you store the data in an "optimized" DB (symbolic only), you can receive data only in arrays of Byte, Char, USInt, and SInt data types.

Note

Importing of S7-300/400 STEP 7 projects containing "ad hoc mode" into the S7-1200

In S7-300/400 STEP 7 projects, "ad hoc mode" is selected by assigning "0" to the LEN parameter. In the S7-1200, you configure the TRCV instruction for ad hoc mode by setting the ADHOC instruction input parameter.

If you import an S7-300/400 STEP 7 project containing "ad hoc mode" into the S7-1200, you must change the LEN parameter to "65535".

Table 11- 27 ERROR and STATUS condition codes for TSEND and TRCV

ERROR	STATUS	Description
0	0000	• Send job completed without error (TSEND) • New data accepted: The current length of the received data is shown in RCVD_LEN (TRCV).
0	7000	• No job processing active (TSEND) • Block not ready to receive (TRCV)
0	7001	• Start of job processing, data being sent: During this processing the operating system accesses the data in the DATA send area (TSEND). • Block is ready to receive, receive job was activated (TRCV).
0	7002	• Follow-on instruction execution (REQ irrelevant), job being processed: The operating system accesses the data in the DATA send area during this processing (TSEND). • Follow-on instruction execution, receive job being processed: Data is written to the receive area during this processing. For this reason, an error could result in inconsistent data in the receive area (TRCV).
1	8085	• LEN parameter is greater than the largest permitted value (TSEND) and (TRCV). • LEN or DATA parameter changed since the first instruction execution (TRCV).
1	8086	The ID parameter is not in the permitted address range.
1	8088	The LEN parameter is larger than the memory area specified in DATA.
1	80A1	Communications error: • The specified connection has not yet established (TSEND and TRCV). • The specified connection is currently being terminated. Transmission or a receive job over this connection is not possible (TSEND and TRCV). • The interface is being reinitialized (TSEND). • The interface is receiving new parameters (TRCV).
1	80C3	Internal lack of connection (Page 637) resources: A block with this ID is already being processed in a different priority class.
1	80C4	Temporary communications error: • The connection to the communications partner cannot be established at this time. • The interface is receiving new parameter settings, or the connection is currently being established.

Connection Ethernet protocols

Every CPU has an integrated PROFINET port, which supports standard PROFINET communications. The TSEND_C, TRCV_C, TSEND and TRCV instructions all support the TCP and ISO on TCP Ethernet protocols.

Refer to "Device Configuration: Configuring the Local/Partner connection path (Page 619)" for more information.

11.2.8.10 Legacy TCON, TDISCON, TSEND, and TRCV instructions

Prior to the release of STEP 7 V13 SP1 and the S7-1200 V4.1 CPUs, the TCON, TDISCON, TSEND, and TRCV instructions could only work with connection parameters with structures according to "TCON_Param". The general concepts apply to both sets of instructions. Refer to the individual legacy TCON, TDISCON, TSEND, and TRCV instructions for programming information.

Selecting the version of the TCON, TDISCON, TSEND, and TRCV instructions

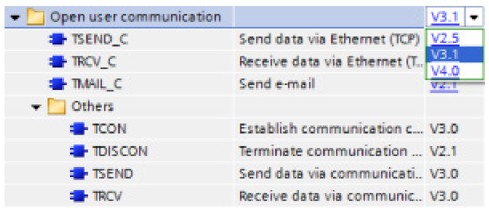
There are two versions of the TCON, TDISCON, TSEND, or TRCV instructions available in STEP 7:

- Versions 2.5 and 3.1 were available in STEP 7 Basic/Professional V13 and earlier.
- Version 4.0 is available in STEP 7 Basic/Professional V13, SP1.

For compatibility and ease of migration, you can choose which instruction version to insert into your user program.

Do not use different instruction versions in the same CPU program.

Click the icon on the instruction tree task card to enable the headers and columns of the instruction tree.



Open user communication		V3.1
TSEND_C	Send data via Ethernet (TCP)	V2.5
TRCV_C	Receive data via Ethernet (T...	V3.1
TMAIL_C	Send e-mail	V4.0
Others		
TCON	Establish communication c...	V3.0
TDISCON	Terminate communication ...	V2.1
TSEND	Send data via communicati...	V3.0
TRCV	Receive data via communic...	V3.0

To change the version of the TCON, TDISCON, TSEND, or TRCV instructions, select the version from the drop-down list. You can select the group or individual instructions.

When you use the instruction tree to place a TCON, TDISCON, TSEND, or TRCV instruction in your program, a new FB or FC instance, depending on the TCON, TDISCON, TSEND, or TRCV instruction selected, is created in the project tree. You can see new FB or FC instance in the project tree under PLC_x > Program blocks > System blocks > Program resources.

To verify the version of a TCON, TDISCON, TSEND, or TRCV instruction in a program, you must inspect project tree properties and not the properties of a box displayed in the program editor. Select a project tree TCON, TDISCON, TSEND, or TRCV FB or FC instance, right-click, select "Properties", and select the "Information" page to see the TCON, TDISCON, TSEND, or TRCV instruction version number.

Legacy TCON, TDISCON, TSEND, and TRCV (TCP communication) instructions

Ethernet communication using TCP and ISO on TCP protocols

Note

TSEND_C and TRCV_C instructions

To help simplify the programming of PROFINET/Ethernet communication, the TSEND_C instruction and the TRCV_C instruction combine the functionality of the TCON, TDISCON. TSEND and TRCV instructions:

- TSEND_C combines the TCON, TDISCON and TSEND instructions.
- TRCV_C combines the TCON, TDISCON and TRCV instructions.

The following instructions control the communication process:

- TCON establishes the TCP/IP connection between the client and server (CPU) PC.
- TSEND and TRCV send and receive data.
- TDISCON breaks the connection.

The minimum size of data that you can transmit (TSEND) or receive (TRCV) is one byte; the maximum size is 8192 bytes. TSEND does not support the transmission of data from Boolean locations, and TRCV will not receive data into Boolean locations. For information transferring data with these instructions, see the section on data consistency (Page 185).

TCON, TDISCON, TSEND, and TRCV operate asynchronously, which means that the job processing extends over multiple instruction executions. For example, you start a job for setting up and establishing a connection by executing an instruction TCON with parameter REQ = 1. Then you use additional TCON executions to monitor the job progress and test for job completion with parameter DONE.

The following table shows the relationships between BUSY, DONE, and ERROR. Use the table to determine the current job status:

Table 11- 28 Interactions between the BUSY, DONE, and ERROR parameters

BUSY	DONE	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	The job successfully completed.
FALSE	FALSE	TRUE	The job was ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	A new job was not assigned.

TCON and TDISCON

Note

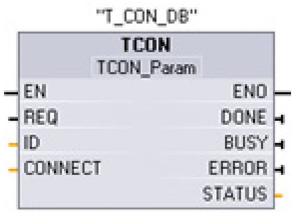
Initializing the communication parameters

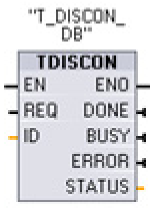
After you insert the TCON instruction, use the "Properties" of the instruction (Page 619) to configure the communication parameters (Page 641). As you enter the parameters for the communication partners in the inspector window, STEP 7 enters the corresponding data in the instance DB for the instruction.

If you want to use a multi-instance DB, you must manually configure the DB on both CPUs.

Table 11- 29 TCON and TDISCON instructions

LAD / FBD		Description

 "T_CON_DB"
TCON
TCON_Param
EN
ENO
REQ
DONE
ID
BUSY
CONNECT
ERROR
STATUS
	`"TCON_DB"(`
	`req:=_bool_in_,`
	`ID:=_undef_in_,`
	`done=>_bool_out_,`
	`busy=>_bool_out_,`
	`error=>_bool_out_,`
	`status=>_word_out_,`
	`connect:=_struct_inout_);`
		TCP and ISO on TCP: TCON initiates a communications connection from the CPU to a communication partner.

 "T_DISCON_DB"
TDISCON
EN
ENO
REQ
DONE
ID
BUSY
ERROR
STATUS
	`"TDISCON_DB"(`
	`req:=_bool_in_,`
	`ID:=_word_in_,`
	`done=>_bool_out_,`
	`busy=>_bool_out_,`
	`error=>_bool_out_,`
	`status=>_word_out_);`
		TCP and ISO on TCP: TDISCON terminates a communications connection from the CPU to a communication partner.

[1] STEP 7 automatically creates the DB when you insert the instruction.

Table 11- 30 Data types for the parameters of TCON and TDISCON

Parameter and type		Data type	Description
REQ	IN	Bool	Control parameter REQ starts the job by establishing the connection specified by ID. The job starts at rising edge.
ID	IN	CONN_OUC (Word)	Reference to the connection to be established (TCON) or terminated (TDISCON) to the remote partner, or between the user program and the communication layer of the operating system. The ID must be identical to the associated parameter ID in the local connection description. Value range: W#16#0001 to W#16#0FFF
CONNECT (TCON)	IN_OUT	TCON_Param	Pointer to the connection description (Page 641)

Parameter and type		Data type	Description
DONE	OUT	Bool	• 0: Job is not yet started or still running. • 1: Job completed without error.
BUSY	OUT	Bool	• 0: Job is completed. • 1: Job is not yet completed. A new job cannot be trig-gered.
ERROR	OUT	Bool	Status parameters with the following values: • 0: No error • 1: Error occurred during processing. STATUS provides detailed information on the type of error.
STATUS	OUT	Word	Status information including error information. (Refer to the Error and Status condition codes in the table below.)

Both communication partners execute the TCON instruction to set up and establish the communication connection. You use parameters to specify the active and passive communication end point partners. After the connection is set up and established, it is automatically maintained and monitored by the CPU.

If the connection is terminated due to a line break or due to the remote communications partner, for example, the active partner attempts to re-establish the configured connection. You do not have to execute TCON again.

An existing connection is terminated and the set-up connection is removed when the TDISCON instruction is executed or when the CPU has gone into STOP mode. To set up and re-establish the connection, you must execute TCON again.

Table 11- 31 ERROR and STATUS condition codes for TCON and TDISCON

ERROR	STATUS	Description
0	0000	Connection was established successfully.
0	7000	No job processing active
0	7001	Start job processing; establishing connection (TCON) or terminating connection (TDISCON)
0	7002	Follow-on call (REQ irrelevant); establishing connection (TCON) or terminating connec-tion (TDISCON)
1	8086	The ID parameter is outside the permitted address range.
1	8087	TCON: Maximum number of connections reached; no additional connection possible.
1	809B	TCON: The local_device_id in the connection description does not match the CPU.
1	80A1	TCON: Connection or port is already occupied by user.
1	80A2	TCON: Local or remote port is occupied by the system.
1	80A3	Attempt being made to re-establish an existing connection (TCON) or terminate a non-existent connection (TDISCON).
1	80A4	TCON: IP address of the remote connection end point is invalid; it matches the local partner IP address.
1	80A5	TCON: Connection ID (Page 637) is already in use.
1	80A7	TCON: Communications error: You executed a TDISCON before the TCON complet-ed.The TDISCON must first completely terminate the connection referenced by the ID.

ERROR	STATUS	Description
1	80B2	TCON: The CONNECT parameter points to a data block that was generated with the attribute "Only store in load memory".
1	80B4	TCON: When using the ISO on TCP (connection_type = B#16#12) to establish a passive connection, condition code 80B4 alerts you that the TSAP entered did not conform to one of the following address requirements: • For a local TSAP length of 2 and a TSAP ID value of either E0 or E1 (hexadecimal) for the first byte, the second byte must be either 00 or 01. • For a local TSAP length of 3 or greater and a TSAP ID value of either E0 or E1 (hexadecimal) for the first byte, the second byte must be either 00 or 01 and all other bytes must be valid ASCII characters. • For a local TSAP length of 3 or greater and the first byte of the TSAP ID does not have a value of either E0 or E1 (hexadecimal), then all bytes of the TSAP ID must be valid ASCII characters. Valid ASCII characters are byte values from 20 to 7E (hexadecimal).
1	80B5	TCON: Connection type "13 = UDP" permits only passive connection establishment.
1	80B6	TCON: Parameter assignment error in CONNECTION_TYPE parameter of the SDT TCON_Param.
1	80B7	TCON: Error in one of the following parameters of the data block for connection description: • block_length • local_tsap_id_len • rem_subnet_id_len • rem_staddr_len • rem_tsap_id_len • next_staddr_len Note: When operating TCON in TCP passive mode, the LOCAL_TSAP_ID_LEN must be "2" and the REM_TSAP_ID_LEN must be "0".
1	80B8	TCON: Parameter in the local connection description and Parameter ID are different.
1	80C3	TCON: All connection resources are in use.
1	80C4	Temporary communications error: • The connection cannot be established at this time (TCON). • The configured connection is currently being removed by TDISCON (TCON). • The connection is currently being established (TDISCON). • The interface is receiving new parameters (TCON and TDISCON).

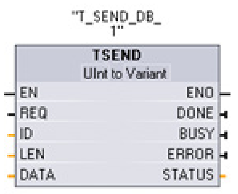
TSEND and TRCV

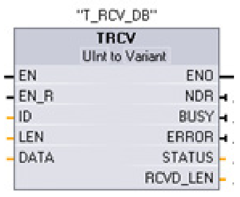
Note

When using PROFINET Open User communication, if you execute a TSEND instruction without a corresponding TRCV instruction executing on the remote device, then the TSEND instruction may reside indefinitely in a "Busy State", waiting for the TRCV instruction to receive the data. In this state, the TSEND instruction "Busy" output is set, and the "Status" output has a value of "0x7002". This condition may occur if you are transferring more than 4096 bytes of data. The issue is resolved at the next execution of the TRCV instruction.

Table 11- 32 TSEND and TRCV instructions

LAD / FBD	SCL	Description

 "T_SEND_DB_1"
TSEND
UInt to Variant
EN
ENO
REQ
DONE
ID
BUSY
LEN
ERROR
DATA
STATUS
	`"TSEND_DB"(`
	`    req:=_bool_in_,`
	`    ID:=_word_in_,`
	`    len:=_udint_in_,`
	`    done=>_bool_out_,`
	`    busy=>_bool_out_,`
	`    error=>_bool_out_,`
	`    status=>_word_out_,`
	`    data:=_variant_inout_);`
		TCP and ISO on TCP: TSEND sends data through a communication connection from the CPU to a partner station.

 "T_RCV_DB"
TRCV
UInt to Variant
EN
ENO
EN_R
NDR
ID
BUSY
LEN
ERROR
DATA
STATUS
RCVD_LEN
	`"TRCV_DB"(`
	`    en_r:=_bool_in_,`
	`    ID:=_word_in_,`
	`    len:=_udint_in_,`
	`    ndr=>_bool_out_,`
	`    busy=>_bool_out_,`
	`    error=>_bool_out_,`
	`    status=>_word_out_,`
	`    rcvd_len=>_udint_out_,`
	`    data:=_variant_inout_);`
		TCP and ISO on TCP: TRCV receives data through a communication connection from a partner station to the CPU.

[1] STEP 7 automatically creates the DB when you insert the instruction.

Table 11- 33 Data types for the parameters of TSEND and TRCV

Parameter and type		Data type	Description
REQ	IN	Bool	TSEND: Starts the send job on a rising edge. The data is transferred from the area specified by DATA and LEN.
EN_R	IN	Bool	TRCV: Enables the CPU to receive; with EN_R = 1, the TRCV is ready to receive. The receive job is processed.
ID	IN	CONN_OUC (Word)	Reference to the associated connection. ID must be identical to the associated parameter ID in the local connection description. Value range: W#16#0001 to W#16#0FFF

Parameter and type		Data type	Description
LEN	IN	UInt	Maximum number of bytes to be sent (TSEND) or received (TRCV): • Default = 0: The DATA parameter determines the length of the data to be sent (TSEND) or received (TRCV). • Ad hoc mode = 65535: A variable length of data is set for reception (TRCV).
DATA	IN_OUT	Variant	Pointer to send (TSEND) or receive (TRCV) data area; data area contains the address and length. The address refers to I memory, Q memory, M memory, or a DB.
DONE	OUT	Bool	TSEND: • 0: Job not yet started or still running. • 1: Job executed without error.
NDR	OUT	Bool	TRCV: • NDR = 0: Job not yet started or still running. • NDR = 1: Job successfully completed.
BUSY	OUT	Bool	• BUSY = 1: The job is not yet complete. A new job cannot be triggered. • BUSY = 0: Job is complete.
ERROR	OUT	Bool	ERROR = 1: Error occurred during processing. STATUS provides detailed information on the type of error
STATUS	OUT	Word	Status information including error information. (Refer to the Error and Status condition codes in the table below.)
RCVD_LEN	OUT	Int	TRCV: Amount of data actually received in bytes

Note

The TSEND instruction requires a low-to-high transition at the REQ input parameter to start a send job. The BUSY parameter is then set to 1 during processing. Completion of the send job is indicated by either the DONE or ERROR parameters being set to 1 for one scan. During this time, any low-to-high transition at the REQ input parameter is ignored.

TRCV Operations

The TRCV instruction writes the received data to a receive area that is specified by the following two variables:

- Pointer to the start of the area
- Length of the area or the value supplied at the LEN input if not 0

Note

The default setting of the LEN parameter (LEN = 0) uses the DATA parameter to determine the length of the data being transmitted. Ensure that the DATA transmitted by the TSEND instruction is the same size as the DATA parameter of the TRCV instruction.

As soon as all the job data has been received, TRCV transfers it to the receive area and sets NDR to 1.

Table 11- 34 Entering the data into the receive area

Protocol variant	Entering the data in the receive area	Parameter "connection_type"	Value of the LEN parameter	Value of the RCVD_LEN parameter (bytes)
TCP	Ad hoc mode	B#16#11	65535	1 to 1472
TCP	Data reception with specified length	B#16#11	0 (recommended) or 1 to 8192, except 65535	1 to 8192
ISO on TCP	Ad hoc mode	B#16#12	65535	1 to 1472
ISO on TCP	protocol-controlled	B#16#12	0 (recommended) or 1 to 8192, except 65535	1 to 8192

Note

Ad hoc mode

The "ad hoc mode" exists with the TCP and ISO on TCP protocol variants. You set "ad hoc mode" by assigning "65535" to the LEN parameter. The receive area is identical to the area formed by DATA. The length of the received data will be output to the parameter RCVD_LEN. Immediately after receiving a block of data, TRCV enters the data in the receive area and sets NDR to 1.

If you store the data in an "optimized" DB (symbolic only), you can receive data only in arrays of Byte, Char, USInt, and SInt data types.

Note

Importing of S7-300/400 STEP 7 projects containing "ad hoc mode" into the S7-1200

In S7-300/400 STEP 7 projects, "ad hoc mode" is selected by assigning "0" to the LEN parameter. In the S7-1200, you set "ad hoc mode" by assigning "65535" to the LEN parameter.

If you import an S7-300/400 STEP 7 project containing "ad hoc mode" into the S7-1200, you must change the LEN parameter to "65535".

TSEND and TRCV Error and Status condition codes

ERROR	STATUS	Description
0	0000	• Send job completed without error (TSEND) • New data accepted: The current length of the received data is shown in RCVD_LEN (TRCV).
0	7000	• No job processing active (TSEND) • Block not ready to receive (TRCV)
0	7001	• Start of job processing, data being sent: During this processing the operating system accesses the data in the DATA send area (TSEND). • Block is ready to receive, receive job was activated (TRCV).
0	7002	• Follow-on instruction execution (REQ irrelevant), job being processed: The operating system accesses the data in the DATA send area during this processing (TSEND). • Follow-on instruction execution, receive job being processed: Data is written to the receive area during this processing. For this reason, an error could result in inconsistent data in the receive area (TRCV).
1	8085	• LEN parameter is greater than the largest permitted value (TSEND) and (TRCV). • LEN or DATA parameter changed since the first instruction execution (TRCV).
1	8086	The ID parameter is not in the permitted address range.
1	8088	The LEN parameter is larger than the memory area specified in DATA.
1	80A1	Communications error: • The specified connection has not yet established (TSEND and TRCV). • The specified connection is currently being terminated. Transmission or a receive job over this connection is not possible (TSEND and TRCV). • The interface is being reinitialized (TSEND). • The interface is receiving new parameters (TRCV).
1	80C3	Internal lack of resources: A block with this ID is already being processed in a different priority class.
1	80C4	Temporary communications error: • The connection to the communications partner cannot be established at this time. • The interface is receiving new parameter settings, or the connection is currently being established.

Connection Ethernet protocols

Every CPU has an integrated PROFINET port, which supports standard PROFINET communications. The TSEND_C, TRCV_C, TSEND and TRCV instructions all support the TCP and ISO on TCP Ethernet protocols.

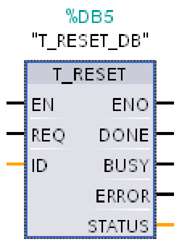
Refer to "Device Configuration: Configuring the Local/Partner connection path (Page 619)" for more information.

11.2.8.11 T_RESET (Terminate and re-establish an existing connection) instruction

The "T_RESET" instruction terminates and then reestablishes an existing connection:

Table 11- 35 T_RESET instruction

LAD / FBD	SCL	Description

 %DB5
"T_RESET_DB"
T_RESET
EN
ENO
REQ
DONE
ID
BUSY
ERROR
STATUS
	`"T_RESET_DB"(`
	`    req:=_bool_in_,`
	`    id:=_word_in_,`
	`    done=>_bool_out_,`
	`    error=>_bool_out_,`
	`    sta-`
	`tus=>_word_out_);`
		Use the T_RESET instruction to terminate and then reestablish an existing connection.

The local end points of the connection are retained. They are generated automatically:

- If a connection has been configured and loaded to the CPU.
- If a connection has been generated by the user program, for example, by calling the instruction "TCON (Page 664)".

The "T_RESET" instruction can be executed for all connection types regardless of whether the local interface of the CPU or the interface of a CM/CP was used for the connection. An exception to this is connections for data transfer in ad-hoc mode with TCP, as such connections cannot be referenced with a connection ID.

Once the instruction "T_RESET" has been called using the REQ parameter, the connection specified with the ID parameter is terminated and, if necessary, the data send and receive buffer cleared. Canceling the connection also cancels any data transfer in progress. There is therefore a risk of losing data if data transfer is in progress. The CPU defined as the active connection partner will then automatically attempt to restore the interrupted communication connection. You therefore do not need to call the "TCON (Page 664)" instruction to reestablish the communication connection.

The output parameters DONE, BUSY, and STATUS indicate the status of the job.

Data types for the parameters

The following table shows the parameters of the "T_RESET" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant	Control parameter REQUEST starts the job for terminating the connection specified by ID. The job starts on a rising edge.
ID	Input	CONN_OUC (WORD)	L, D or constant	Reference to the connection to the passive partner which is to be terminated. ID must be identical to the corresponding parameter ID in the local connection description. Range of values: W#16#0001 to W#16#0FFF
DONE	Output	BOOL	I, Q, M, D, L	Status parameter DONE • 0: Job not yet started or still executing. • 1: Job executed without errors.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter BUSY • 0: Job is complete. • 1: Job is not yet complete.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter ERROR • 0: No error occurred. • 1: Error occurred during processing. The STATUS parameter supplies detailed information on the type of error
STATUS	Output	WORD	I, Q, M, D, L	Status parameter STATUS Error information (see "STATUS parameter" table).

STATUS parameter

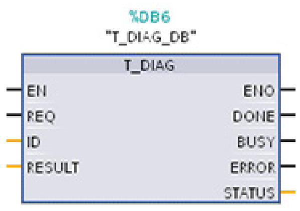
Error bit	STATUS* (W#16#...)	Description
0	0000	No error.
0	0001	Connection has not been established.
0	7001	Connection termination launched.
0	7002	Connection being terminated.
1	8081	Unknown connection specified at the ID parameter.

11.2.8.12 T_DIAG (Checks the status of connection and reads information) instruction

The "T_DIAG" instruction checks the status of a connection and reads further information on the local end point of this connection:

Table 11- 36 T_DIAG instruction

LAD / FBD	SCL	Description

 %DB6
"T_DIAG_DB"
T_DIAG
EN
REQ
ID
RESULT
ENO
DONE
BUSY
ERROR
STATUS
	`"T_DIAG_DB"(`
	`    req:=_bool_in_,`
	`    id:=_word_in_,`
	`    done=>_bool_out_,`
	`    error=>_bool_out_,`
	`    sta-`
	`tus=>_dword_out_);`
		Use the T_DIAG instruction to check the status of a connection and read further information on the local end point of this connection.

The "T_DIAG" instruction operates as follows:

- The connection is referenced by the ID parameter. You can read both connection end points configured in the connection editor and programmed connection end points (e.g. with the "TCON" instruction).

  Temporary connection end points (for example end points created when you connect to an engineering station) cannot be diagnosed, as no connection ID is generated in this process.
- The connection information read is stored in a structure referenced by the RESULT parameter.
- The output parameter STATUS indicates whether it was possible to read the connection information. The connection information in the structure at the RESULT parameter is only valid if the "T_DIAG" instruction has been completed with STATUS = W#16#0000 and ERROR = FALSE.

  Connection information cannot be evaluated if an error occurs.

Possible connection information

The "TDiag_Status" structure can be used to read the connection information at the RESULT parameter. The TDiag_Status structure only contains the most important information about a connection end point (for example, the protocol used, the connection status, and the number of data bytes sent or received).

The structure and parameters of the TDiag_Status structure are described below (see the "TDIAG_Status structure" table).

Data types for the parameters

The following table shows the parameters of the "T_DIAG" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant	Starts the instruction to check the connection specified in the ID parameter when there is a positive edge.
ID	Input	CONN_OUC (WORD)	L, D or constant	Reference to the assigned connection. Range of values: W#16#0001 to W#16#0FFF
RESULT	InOut	VARIANT	D	Pointer to the structure in which the connection information is stored. The structure TDiag_Status can be used at the RESULT parameter (for a description, see the "TDIAG_Status structure" table).
DONE	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: Instruction not yet started or still in progress. • 1: Instruction executed without errors.
BUSY	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: Instruction not yet started or already completed. • 1: Instruction not yet completed. A new job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: No error. • 1: Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

Parameters BUSY, DONE, and ERROR

You can check the status of "T_DIAG" instruction execution with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. You use the DONE parameter to check whether or not an instruction has been executed successfully. The ERROR parameter is set if errors occur during execution of "T_DIAG".

The following table shows the relationship between the BUSY, DONE, and ERROR parameters:

BUSY	DONE	ERROR	Description
1	-	-	The instruction is being processed.
0	1	0	The instruction has been executed successfully. The data in the structure referenced by RESULT are only valid if this is the case.
0	0	1	Instruction completed with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new instruction has been assigned.

STATUS parameter

The following table shows the meaning of the values at the STATUS parameter:

Error bit	STATUS* (W#16#...)	Description
0	0000	The instruction "T_DIAG" has been executed successfully. The data in the structure referenced at the RESULT parameter can be evaluated.
0	7000	No instruction processing active.
0	7001	Instruction processing launched.
0	7002	Connection information is being read (REQ parameter irrelevant).
1	8086	The value at the ID parameter is outside the valid range (W#16#0001 ... W#16#0FFF).
1	8089	The RESULT parameter points to an invalid data type (structures TDIAG_Status and TDIAG_StatusExt only).
1	80A3	The ID parameter references a connection end point which does not exist. With programmed connections, this error can also occur after the "TDISCON" instruction is called.
1	80C4	Internal error. Access to the connection end point is temporarily unavailable.

TDIAG_Status Structure

The table below details the form of the TDIAG_Status structure. The value of each element is only valid if the instruction has been executed without errors. If an error occurs, the content of the parameters will not change:

Name	Data type	Description
The following parameters are in the TDIAG_Status structure:		
InterfaceID	HW_ANY	Interface ID (LADDR) of the CPU or the CM/CP.
ID	CONN_OUC	ID of the connection diagnosed. Following a successful call, the value of this element is identical to the parameter ID of the "T_DIAG" instruction.
ConnectionType	BYTE	Protocol type used for connection: • 0x01: Not used. • ... • 0x0B: TCP protocol (IP_v4) • 0x0C: ISO-on-TCP protocol (RFC1006) • 0x0D: TCP protocol (DNS) • 0x0E: Dial-in protocol • 0x0F: WDC protocol • 0x10: SMTP protocol • 0x11: TCP protocol • 0x12: TCP and ISO-on-TCP protocol (RFC1006) • 0x13: UDP protocol • 0x14: Reserved • 0x15: PROFIBUS bus access protocol (FDL) • 0x16: ISO 8073 transport protocol (ISO native) • ... • 0x20: SMTP or SMTPS protocol - based on IPv4 • 0x21: SMTP or SMTPS protocol - based on IPv6 • 0x22: SMTP or SMTPS protocol - based on FQDN (**F**ully **Q**ualified **D**omain **N**ame) • ... • 0x70: S7 connection • Other: Reserved
ActiveEstablished	BOOL	• FALSE: Locally, the passive connection end point • TRUE: Locally, the active connection end point

Name	Data type	Description
State	BYTE	Current status of the connection end point • 0x00: Not used. • 0x01: Connection terminated. Temporary status, for example, after the "T_RESET" instruction is called. The system then automatically attempts to reestablish the connection. • 0x02: The active connection end point is attempting to establish a connection to the remote communication partner. • 0x03: The passive connection end point is waiting for establishment of the connection to the remote communication partner. • 0x04: Connection established. • 0x05: The connection is being terminated. This may be because the "T_RESET" or "T_DISCON" instruction has been called. Other possible reasons are protocol errors and line breaks. • 0x06..0xFF: Not used.
Kind	BYTE	Mode of the connection end point: • 0x00: Not used. • 0x01: Configured, static connection which has been configured and loaded to the CPU. • 0x02: Configured, dynamic connection which has been configured and loaded to the CPU (not currently supported). • 0x03: Programmed connection generated in the user program with the instruction "TCON". A call of the instruction "TDISCON" or a transition to CPU STOP status has destroyed the connection end point. • 0x04: Temporary, dynamic connection established by the engineering station (ES) or operator station (OS), for example. (this connection type cannot currently be diagnosed as there is no ID). • 0x05..0xFF: Not used.
SentBytes	UDINT	Number of data bytes sent.
ReceivedBytes	UDINT	Number of data bytes received.

11.2.8.13 TMAIL_C (Send an email using the Ethernet interface of the CPU) instruction

Overview

You use the "TMAIL_C" instruction to send an e-mail using the Ethernet interface of the S7-1200 CPU.

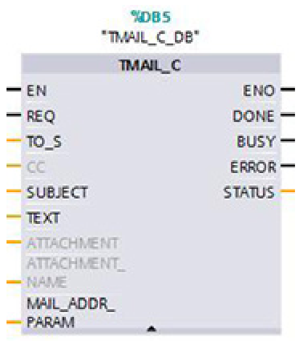
The TMAIL_C instruction has two functionalities:

- Email over the CPU Interface (only SMTP without SSL)
- Email over a CP Interface (either SMTP without SSL or SMTP with SSL) If you want to use the SSL functionality, you must set the TMAIL_C input parameter CERTINDEX = 1 and use the CP Interface. Also, the correct certificate must be stored in CP cert storage.

The instruction can only be used once the hardware has been configured and if the network infrastructure allows for a communication connection to the mail server.

Table 11- 37 TMAIL_C instruction

LAD / FBD	SCL	Description

 %DB5
"TMAIL_C_DB"
TMAIL_C
EN
REQ
TO_S
CC
SUBJECT
TEXT
ATTACHMENT
ATTACHMENT_NAME
MAIL_ADDR_PARAM
ENO
DONE
BUSY
ERROR
STATUS
	`"TMAIL_C_DB"(`
	`req:=_bool_in_,`
	`to_s:=_string_in_,`
	`cc:=_string_in_,`
	`subject:=_string_in_,`
	`text:=_string_in_,`
	`attachment:=_variant_in_,`
	`attach-`
	`ment_name:=_string_in_,`
	
	`mail_addr_param:=_string_i`
	`n_,`
	`done=>_bool_out_,`
	`busy=>_bool_out_,`
	`error=>_bool_out_,`
	`status=>_word_out_);`
		The TMAIL_C instruction sends an e-mail using the Ethernet interface of the S7-1200 CPU.

[1] STEP 7 automatically creates the DB when you insert the instruction.

You define the content of the e-mail, and the connection data, using the following parameters:

- You define the recipient addresses with the parameters TO_S and CC.
- You define the content of the e-mail with the parameters SUBJECT and TEXT.
- You can define an attachment using VARIANT pointers at the ATTACHMENT and ATTACHMENT_NAME parameters.
- The connection data are defined, and addressing and authentication for the mail server executed, using the system data type Tmail_v4 or Tmail_FQDN at the MAIL_ADDR_PARAM parameter. If you are using the interface of the S7-1200 CPU, the system data type Tmail_v4 must be used. In this case, the e-mail can only be sent using SMTP.

- You start the sending of an e-mail with an edge change from "0" to "1" for the REQ parameter.
- The job status is indicated by the output parameters "BUSY", "DONE", "ERROR" and "STATUS".

You cannot send an SMS directly with the "TMAIL_C" instruction. Whether or not the e-mail can be forwarded by the mail server as an SMS depends on your telecommunications provider.

Operation of the instruction

The "TMAIL_C" instruction works asynchronously, which means its execution extends over multiple calls. You must specify an instance when you call the instruction "TMAIL_C".

In the following cases, the connection to the mail server will be lost:

- If the CPU switches to STOP while "TMAIL_C" is active.
- If communication problems occur at the Industrial Ethernet bus. In this case, the transfer of the e-mail will be interrupted and it will not reach its recipient.

The connection is also canceled once the instruction has been successfully executed and the e-mail sent.

NOTICE

Changing user programs

You can change the parts of your user program that directly affect calls of "TMAIL_C" only when:

- The CPU is in "STOP" mode.
- No e-mail is being sent (REQ = 0 and BUSY = 0).

This relates, in particular, to deleting and replacing program blocks that contain "TMAIL_C" calls or calls for the instance of "TMAIL_C".

Ignoring this restriction can tie up connection resources. The automation system can change to an undefined status with the TCP/IP communication functions via Industrial Ethernet.

A warm or cold restart of the CPU is required after the changes are transferred.

Data consistency

The TO_S, CC, SUBJECT, TEXT, ATTACHMENT and MAIL_ADDR_PARAM parameters are applied by the "TMAIL_C" instruction while it is running, which means that they may only be changed after the job has been completed (BUSY = 0).

SMTP authentication

Authentication refers here to a procedure for verifying identity, for example, with a password query.

If you are using the S7-1200 CPU interface, the instruction "TMAIL_C" supports the SMTP authentication procedure AUTH-LOGIN which is required by most mail servers. For information about the authentication procedure of your mail server, please refer to your mail server manual or the website of your Internet service provider.

- Before you can use the AUTH-LOGIN authentication procedure, the "TMAIL_C" instruction requires the user name with which it is to log on to the mail server. This user name corresponds to the user name with which you set up a mail account on your mail server. It is transferred via the UserName parameter to the structure at parameter MAIL_ADDR_PARAM.

  If no user name is specified at the MAIL_ADDR_PARAM parameter, the AUTH-LOGIN authentication procedure is not used. The e-mail is then sent without authentication.
- To log on, the "TMAIL_C" instruction also requires the associated password. This password corresponds to the password you specified when you set up your mail account. It is transferred via the PassWord parameter to the structure at parameter MAIL_ADDR_PARAM.

Data types for the parameters

The following table shows the parameters of the "TMAIL_C" instruction:

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L, T, C or constant	Control parameter REQUEST: Activates the sending of an e-mail upon a rising edge.
TO_S (Page 694)	Input	STRING	D	Recipient addresses STRING with a maximum length of 180 characters (bytes). For the e-mail address format, please see the example in the parameter description.
CC (Page 694)	Input	STRING	D	CC recipient addresses (optional) STRING with a maximum length of 180 characters (bytes). Same e-mail address format as for the TO_S parameter. If an empty string is assigned here, the e-mail is not sent to a CC recipient.
SUBJECT	Input	STRING	D	Subject of the e-mail STRING with a maximum length of 180 characters (bytes).

Parameter	Declaration	Data type	Memory area	Description
TEXT	Input	STRING	D	Text of the e-mail (optional) STRING with a maximum length of 180 characters (bytes). If an empty string is assigned at this parameter, the e-mail is sent without text.
ATTACHMENT	Input	VARIANT	D	E-mail attachment (optional) Reference to a byte/word/double word field (ArrayOfByte, ArrayOfWord or ArrayOfDWord) with a maximum length of 64 bytes. If no value is assigned, the e-mail is sent without an attachment.
ATTACHMENT_NAME	Input	VARIANT	D	E-mail attachment name (optional) Reference to a character string with a maximum length of 50 characters (bytes) to define the file name of the attachment. If an empty string is assigned at this parameter, the e-mail attachment will be sent with the file name "attachment.bin".
MAIL_ADDR_PARAM (Page 692)	Input	VARIANT	D	Connection parameter and address of the e-mail server To define the connection parameters, use the structure Tmail_v4 or Tmail_FQDN (see parameter description).
DONE (Page 694)	Output	BOOL	I, Q, M, D, L	Status parameter • DONE = 0: Job not yet started or still executing. • DONE = 1: Job was executed without errors.
BUSY (Page 694)	Output	BOOL	I, Q, M, D, L	Status parameter • BUSY=0: The processing of "TMAIL_C" was stopped. • BUSY = 1: E-mail transmission is not yet complete.
ERROR (Page 694)	Output	BOOL	I, Q, M, D, L	Status parameter • ERROR = 0: No error has occurred. • ERROR = 1: An error occurred during processing. STATUS supplies detailed information on the type of error.
STATUS (Page 694)	Output	WORD	I, Q, M, D, L	Status parameter Return value or error information of the "TMAIL_C" instruction (see parameter description).

You will find more detailed information on valid data types in "Overview of valid data types".

Note

Optional parameters

The optional parameters CC, TEXT, and ATTACHMENT are only sent with the e-mail if the corresponding parameters contain a string of length > 0.

MAIL_ADDR_PARAM parameter

At the MAIL_ADDR_PARAM parameter, you define the connection for sending the e-mail in the structure Tmail_v4 or Tmail_FQDN, and save the e-mail server address and login details.

The structure you use at the MAIL_ADDR_PARAM parameter depends on the format in which the e-mail server is to be addressed:

- Tmail_v4: Addressing by IP address (IPv4).
- Tmail_FQDN: Addressing by fully qualified domain name (FQDN).

Which structure you can use depends on the interface addressed at the InterfaceId parameter. If you want to use the "TMAIL_C" instruction with the internal interface, the structure Tmail_v4 must be used at the MAIL_ADDR_PARAM parameter.

Table 11- 38 **Tmail_v4: Addressing the mail server by IP address (IPv4)**

Parameter	Data type	Description
Tmail_v4	Struct	
InterfaceId	LADDR	Hardware identifier of the interface
ID	CONN_OUC	Connection ID
ConnectionType	BYTE	Connection type. Select 16#20 as the connection type for IPv4.
ActiveEstablished	BOOL	Status bit. Set to "1" once the connection is established.
CertIndex	BYTE	=0: SMTP used (**S**imple **M**ail **T**ransfer **P**rotocol). SMTP must be used if the e-mail is being sent via the interface of an S7-1200 CPU.
WatchDogTime	TIME	Execution watchdog. Use this parameter to define the maximum execution time for the send operation. Note: Connection establishment can take longer (approx. one minute) if the connection is slow. When you specify the WATCH_DOG_TIME parameter, remember to allow for the time required to establish the connection. The connection is terminated once the specified time has elapsed.
MailServerAddress	IP_v4	IP address of the mail server. IPv4 in the following format: XXX.XXX.XXX.XXX (decimal). Example: 192.142.131.237.
UserName	STRING[254]	Mail server login name
PassWord	STRING[254]	Mail server password

Parameter	Data type	Description
From	EMAIL_ADDR	E-mail sender address, which is defined using the following two STRING parameters. For example: "my-name@mymailserver.com".
Local-PartPlusAtSign	STRING[64]	Local part of sender address, including @ sign. Example: "myname@".
FullQualified-DomainName	STRING[254]	Fully Qualified Domain Name (FQDN for short) of the mail server. Example: "mymailserver.com".

Table 11- 39 **Tmail_FQDN: Addressing the mail server by FQDN**

Parameter	Data type	Description
Tmail_v6	Struct	
Tmail_FQDN	LADDR	Hardware identifier of the interface
ID	CONN_OUC	Connection ID
ConnectionType	BYTE	Connection type. Select 16#22 as the connection type for FQDN.
ActiveEstablished	BOOL	Status bit. Set to "1" once the connection is established.
CertIndex	BYTE	=0: SMTP used (**S**imple **M**ail **T**ransfer **P**rotocol). SMTP must be used if the e-mail is being sent via the interface of an S7-1200 CPU.
WatchDogTime	TIME	Execution watchdog. Use this parameter to define the maximum execution time for the send operation. Note: Connection establishment can take longer (approx. one minute) if the connection is slow. When you specify the WATCH_DOG_TIME parameter, remember to allow for the time required to establish the connection. The connection is terminated once the specified time has elapsed.
MailServerAddress	STRING[254]	FQDN (**F**ully **Q**ualified **D**omain **N**ame) of the mail server. The mail server is addressed using the fully qualified domain name. Example: "www.mymailserver.com.".
UserName	STRING[254]	Mail server login name
PassWord	STRING[254]	Mail server password
From	Struct	E-mail sender address, which is defined using the following two STRING parameters. For example: "my-name@mymailserver.com".
Local-PartPlusAtSign	STRING[64]	Local part of sender address, including @ sign. Example: "myname@".
FullQualified-DomainName	STRING[254]	Fully Qualified Domain Name (FQDN for short) of the mail server. Example: "mymailserver.com".

TO_S and CC parameters

The TO_S and CC parameters are strings, for example, with the following content:

- <wenna@mydomain.com>, <ruby@mydomain.com>
- <admin@mydomain.com>, <judy@mydomain.com>

Note the following rules when entering the parameters:

- A space and an opening pointed bracket "<" must be entered before each address.
- A closing pointed bracket ">" must be entered after each address.
- A comma must be entered between the addresses in TO and CC.

For runtime and memory space reasons, the "TMAIL_C" instruction does not perform a syntax check of parameter TO_S or CC

DONE, BUSY and ERROR parameters

The output parameters DONE, BUSY and ERROR are each displayed for only one cycle if the status of the BUSY output parameter changes from "1" to "0".

The following table shows the relationship between DONE, BUSY, and ERROR. Using this table, you can determine the current status of the instruction "TMAIL_C and when the sending of the e-mail is complete.

DONE	BUSY	ERROR	Description
0	1	0	The job is being processed.
1	0	0	Job successfully completed.
0	0	1	The job ended with an error. The cause of the error can be found in the STATUS (Page 694) parameter.
0	0	0	The "TMAIL_C" instruction was not assigned a (new) job.

STATUS parameter

The following table shows the return values of TMAIL_C at the STATUS parameter:

Return val-ue STATUS* (W#16#...):	Explanation	Notes
0000	The processing of TMAIL_C was completed without errors.	Error-free completion of TMAIL_C does not mean that the e-mail sent will necessarily arrive. Incorrectly entering the recipient addresses does not generate a status error of the TMAIL_C instruction. In this case, there is no guarantee that the e-mail will be sent to other recipients, even if these were entered correct-ly.
7001	TMAIL_C is active (BUSY = 1).	First call: Job triggered.

Return value STATUS* (W#16#...):	Explanation	Notes
7002	TMAIL_C is active (BUSY = 1).	Intermediate call: Job already active.
8xxx	The processing of TMAIL_C was completed with an error code of the communication instructions called internally.	For detailed information, refer to the STATUS parameter descriptions for the TCON, TDISCON, TSEND and TRCV (Page 664) communication instructions.
8010	Error during connection establishment	You will find further information on evaluation in the SFB_STATUS parameter of the instance data block. The error code displayed at the SFB_STATUS parameter is explained in the STATUS parameter description for the TCON (Page 664) instruction.
8011	Error sending the data	You will find further information on evaluation in the SFB_STATUS parameter of the instance data block. The error code displayed at the SFB_STATUS parameter is explained in the STATUS parameter description for the TSEND (Page 664) instruction.
8012	Error receiving the data	You will find further information on evaluation in the SFB_STATUS parameter of the instance data block. The error code displayed at the SFB_STATUS parameter is explained in the STATUS parameter description for the TRCV (Page 664) instruction.
8013	Error during connection establishment	You will find further information on evaluation in the SFB_STATUS parameter of the instance data block. The error code displayed at the SFB_STATUS parameter is explained in the STATUS parameter description for the TCON (Page 664) and TDISCON (Page 664) instructions.
8014	Establishment of a connection is not possible.	You may have entered an incorrect mail server IP address (MailServerAddress (Page 692)) or too short a time span (WatchDogTime (Page 692)) for connection establishment. It is also possible that the CPU has no connection to the network or that the CPU configuration is incorrect.
8015	Incorrect data type for MAIL_ADDR_PARAM	The only valid data types are the system data types (structures) Tmail_v4 and TMail_FQDN.
8016	Incorrect data type for the ATTACHMENT parameter	The only valid data types are ArrayOfByte, ArrayOfWord and ArrayOfDWord.
8017	Incorrect data length for the ATTACHMENT parameter	Data length must be <= 65534 bytes.

Return value STATUS* (W#16#...):	Explanation	Notes
82xx, 84xx, or 85xx	The error message originates from the mail server and corresponds, except for the "8", to the error number of the SMTP protocol. The following lines list several error codes that can occur.	You will find more detailed information on the SMTP error code and other error codes in the SMTP protocol on the Internet or in the error documentation of the mail server. You can also view the most recent error message from the mail server in your instance DB in the BUFFER1 parameter. You will find the last data sent by the TMAIL_C instruction under DATEN in the instance DB.
8450	Action not executed: Mailbox not available/cannot be reached	Try again later.
8451	Action aborted: Local processing error	Try again later.
8500	Syntax error: Error not recognized. This also includes the error when a command string is too long. This can also occur when the e-mail server does not support the LOGIN authentication procedure.	Check the parameters of TMAIL_C. Try to send an e-mail without authentication. To do this, replace the content of the UserName parameter with an empty string. If no user name is specified, the LOGIN authentication procedure is not used.
8501	Syntax error: Incorrect input at a parameter	Possible cause: Incorrect address at the TO_S or CC parameter (see also: TO_S and CC parameters (Page 694)).
8502	Command unknown or not implemented	Check your entries, in particular the FROM parameter. It may be incomplete and you may have forgotten the "@" or "." (see also: TO_S and CC parameters (Page 694)).
8535	SMTP authentication incomplete	You have possibly entered an incorrect user name or incorrect password.
8550	Mail server cannot be reached. You have no access rights.	You may have entered an incorrect user name or password, or the mail server may not support your login. Another cause of error could be a mistake in the domain name after the "@" at the TO_S or CC parameter (see also: TO_S and CC parameters (Page 694)).
8552	Action aborted: Assigned memory size has been exceeded	Try again later.
8554	Transfer failed	Try again later.
* You can display error codes as integer or hexadecimal values in the program editor.		

11.2.8.14 UDP

UDP is a standard protocol described by RFC 768: User Datagram Protocol. UDP provides a mechanism for one application to send a datagram to another; however, delivery of data is not guaranteed. This protocol has the following features:

- A quick communications protocol, because it is very hardware-intimate
- Suitable for small-sized to medium data amounts (up to 1472 bytes)
- UDP is a simpler transport control protocol than TCP, with a thin layer that yields low overheads
- Can be used very flexibly with many third-party systems
- Routing-capable
- Uses port numbers to direct the datagrams
- Messages are unacknowledged: The application is required to take responsibility for error recovery and security
- Programming effort is required for data management due to the SEND / RECEIVE programming interface

UDP supports broadcast communication. To use broadcast, you must configure the IP address portion of the ADDR configuration. For example: A CPU with an IP address of 192.168.2.10 and subnet mask of 255.255.255.0 would use a broadcast address of 192.168.2.255.

11.2.8.15 TUSEND and TURCV

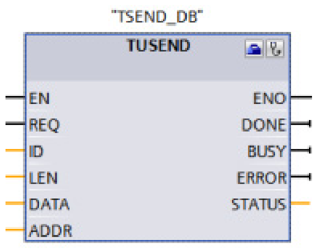
The following instructions control the UDP communication process:

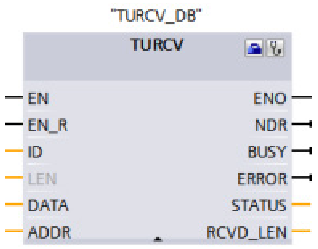
- TCON establishes the communication between the client and server (CPU) PC.
- TUSEND and TURCV send and receive data.
- TDISCON disconnects the communication between the client and server.

Refer to TCON, TDISCON, TSEND, and TRCV (Page 664) in the "TCP and ISO-on-TCP" section for more information on the TCON and TDISCON communication instructions.

Table 11- 40 TUSEND and TURCV instructions

LAD / FBD	SCL	Description

 "TSEND_DB"
TUSEND
EN
REQ
ID
LEN
DATA
ADDR
ENO
DONE
BUSY
ERROR
STATUS
	`"TUSEND_DB"(`
	`    req:=_bool_in_,`
	`    ID:=_word_in_,`
	`    len:=_udint_in_,`
	`    done=>_bool_out_,`
	`    busy=>_bool_out_,`
	`    error=>_bool_out_,`
	`    status=>_word_out_,`
	`    data:=_variant_inout_);`
		The TUSEND instruction sends data via UDP to the remote partner specified by the parameter ADDR.
		To start the job for sending data, call the TUSEND instruction with REQ = 1.

 "TURCV_DB"
TURCV
EN
EN_R
ID
LEN
DATA
ADDR
ENO
NDR
BUSY
ERROR
STATUS
RCVD_LEN
	`"TURCV_DB"(`
	`    en_r:=_bool_in_,`
	`    ID:=_word_in_,`
	`    len:=_udint_in_,`
	`    ndr=>_bool_out_,`
	`    busy=>_bool_out_,`
	`    error=>_bool_out_,`
	`    status=>_word_out_,`
	`    rcvd_len=>_udint_out_,`
	`    data:=_variant_inout_);`
		The TURCV instruction receives data via UDP. The parameter ADDR shows the address of the sender. After successful completion of TURCV, the parameter ADDR contains the address of the re-mote partner (the sender).
		TURCV does not support ad hoc mode.
		To start the job for receiving data, call the TURCV instruction with EN_R = 1.

[1] STEP 7 automatically creates the DB when you insert the instruction.

TCON, TDISCON, TUSEND, and TURCV operate asynchronously, which means that the job processing extends over multiple instruction executions.

Table 11- 41 TUSEND and TURCV data types for the parameters

Parameter and type		Data type	Description
REQ (TUSEND)	IN	Bool	Starts the send job on a rising edge. The data is transferred from the area specified by DATA and LEN.
EN_R (TURCV)	IN	Bool	• 0: CPU cannot receive. • 1: Enables the CPU to receive. The TURCV instruction is ready to receive, and the receive job is processed.
ID	IN	Word	Reference to the associated connection between the user program and the communication level of the operating system. ID must be identical to the associated parameter ID in the local connection description. Range of values: W#16#0001 to W#16#0FFF.
LEN	IN	UDInt	Number of bytes to be sent (TUSEND) or received (TURCV). • Default = 0. The DATA parameter determines the length of the data to be sent or received. • Otherwise, range of values: 1 to 1472
DONE (TUSEND)	IN	Bool	Status parameter DONE (TUSEND): • 0: Job is not yet started or still running. • 1: Job completed without error.
NDR (TURCV)	OUT	Bool	Status parameter NDR (TURCV): • 0: Job not yet started or still running. • 1: Job has successfully completed.
BUSY	OUT	Bool	• 1: Job is not yet completed. A new job cannot be triggered. • 0: Job has completed.
ERROR	OUT	Bool	Status parameters with the following values: • 0: No error • 1: Error occurred during processing. STATUS provides detailed information on the type of error.
STATUS	OUT	Word	Status information including error information. (Refer to the Error and Status condition codes in the table below.)
RCVD_LEN	OUT	UDInt	Number of bytes received (TURCV)

Parameter and type		Data type	Description
DATA	IN_OUT	Variant	Address of the sender area (TUSEND) or receive area (TURCV): • The process image input table • The process image output table • A memory bit • A data block
ADDR	IN_OUT	Variant	Pointer to the address of the receiver (for TUSEND) or sender (for TURCV) (for example, P#DB100.DBX0.0 byte 8). The pointer may point to any memory area. A structure of 8 bytes is required as follows: • First 4 bytes contain the remote IP address. • Next 2 bytes specify the remote port number. • Last 2 bytes are reserved.

The job status is indicated at the output parameters BUSY and STATUS. STATUS corresponds to the RET_VAL output parameter of asynchronously functioning instructions.

The following table shows the relationships between BUSY, DONE (TUSEND), NDR (TURCV), and ERROR. Using this table, you can determine the current status of the instruction (TUSEND or TURCV) or when the sending (transmission) / receiving process is complete.

Table 11- 42 Status of BUSY, DONE (TUSEND) / NDR (TURCV), and ERROR parameters

BUSY	DONE / NDR	ERROR	Description
TRUE	irrelevant	irrelevant	The job is being processed.
FALSE	TRUE	FALSE	The job was completed successfully.
FALSE	FALSE	TRUE	The job was ended with an error. The cause of the error can be found in the STATUS parameter.
FALSE	FALSE	FALSE	The instruction was not assigned a (new) job.

[1] Due to the asynchronous function of the instructions: For TUSEND, you must keep the data in the sender area consistent until the DONE parameter or the ERROR parameter assumes the value TRUE. For TURCV, the data in the receiver area are only consistent when the NDR parameter assumes the value TRUE.

Table 11- 43 TUSEND and TURCV condition codes for ERROR and STATUS

ERROR	STATUS	Description
0	0000	• Send job completed without error (TUSEND). • New data were accepted. The current length of the received data is shown in RCVD_LEN (TURCV).
0	7000	• No job processing active (TUSEND) • Block not ready to receive (TURCV)
0	7001	• Start of job processing, data being sent (TUSEND): During this processing, the operating system accesses the data in the DATA send area. • Block is ready to receive, receive job was activated (TURCV).
0	7002	• Follow-on instruction execution (REQ irrelevant), job being processed (TUSEND): During this processing, the operating system accesses the data in the DATA send area. • Follow-on instruction execution, job being processed: During this processing, the TURCV instruction writes data to the receive area. For this reason, an error could result in inconsistent data in the receive area.
1	8085	LEN parameter is greater than the largest permitted value, has the value 0 (TUSEND), or you changed the value of the LEN or DATA parameter since the first instruction execution (TURCV).
1	8086	The ID parameter is not in the permitted address range.
1	8088	• LEN parameter is larger than the memory area (TUSEND) or receive area (TURCV) specified in DATA. • Receive area is too small (TURCV).
1	8089	ADDR parameter does not point to a data block.
1	80A1	Communications error: • The specified connection between user program and communications layer of the operating system has not yet been established. • The specified connection between the user program and the communication layer of the operating system is currently being terminated. Transmission (TUSEND) or a receive job (TURCV) over this connection is not possible. • The interface is being reinitialized.
1	80A4	IP address of the remote connection end point is invalid; it is possible that it matches the local IP address (TUSEND).
1	80B3	• The set protocol variant (connection_type parameter in the connection description) is not UDP. Please use the TSEND or TRCV instruction. • ADDR parameter: Invalid settings for port number (TUSEND)
1	80C3	• A block with this ID is already being processed in a different priority class. • Internal lack of resources
1	80C4	Temporary communications error: • The connection between the user program and the communication level of the operating system cannot be established at this time (TUSEND). • The interface is receiving new parameters (TUSEND). • The connection is currently being reinitiated (TURCV).

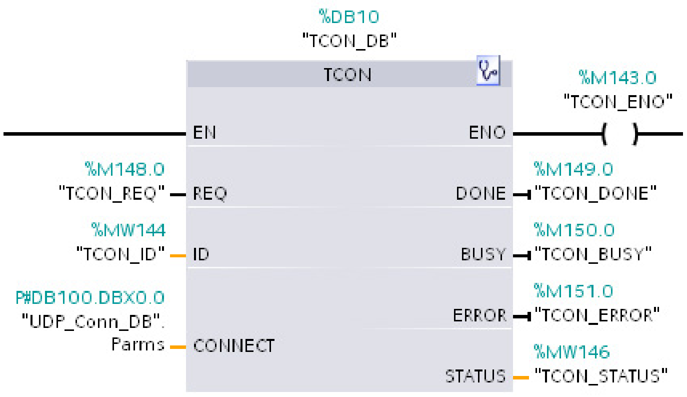
Connection Ethernet protocols

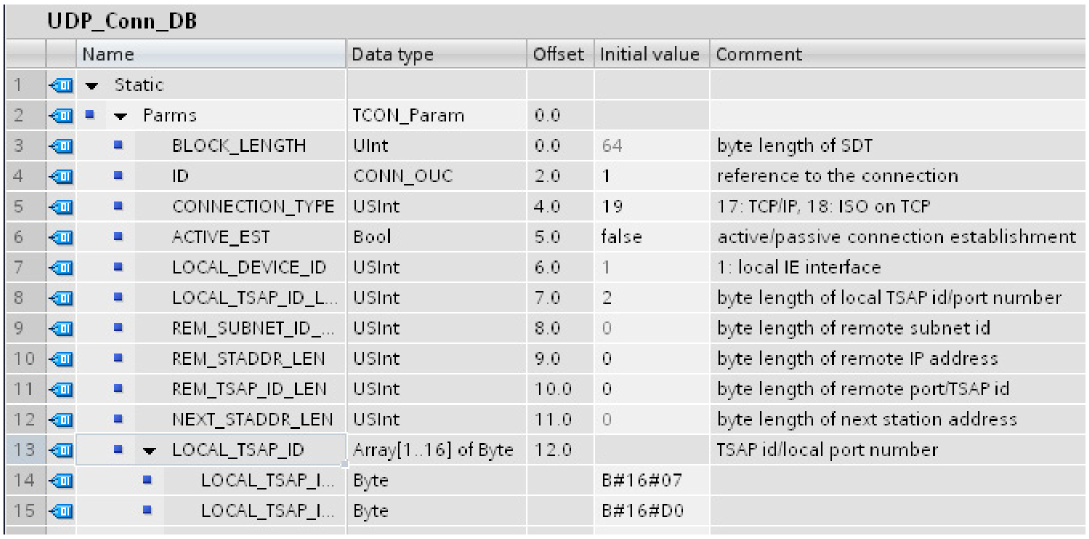
Every CPU has an integrated PROFINET port, which supports standard PROFINET communications. The TUSEND and TURCV instructions support the UDP Ethernet protocol.

Refer to "Configuring the Local/Partner connection path" (Page 619)" in the "Device configuration" chapter for more information.

Operations

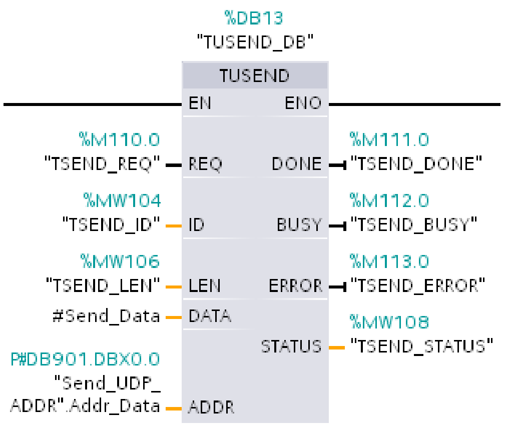
Both partners are passive in UDP communication. Typical parameter start values for the "TCON_Param" data type are shown in the following figures. Port numbers (LOCAL_TSAP_ID) are written in a 2-byte format. All ports except for 161, 34962, 34963, and 34964 are allowed.

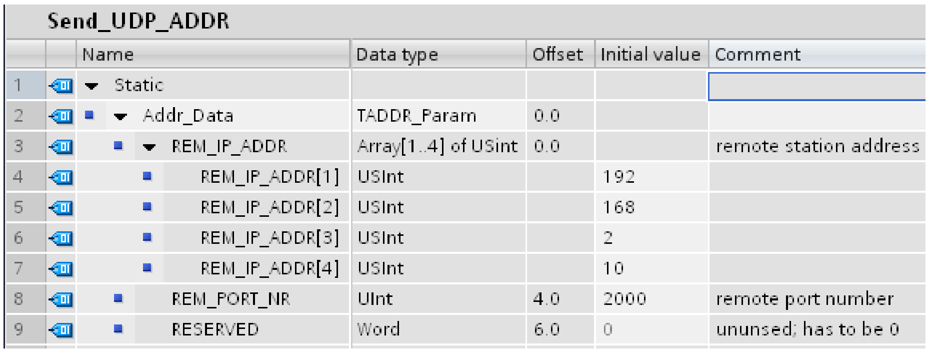



UDP_Conn_DB

	Name	Data type	Offset	Initial value	Comment
1	Static				
2	Parms	TCON_Param	0.0		
3	BLOCK_LENGTH	UInt	0.0	64	byte length of SDT
4	ID	CONN_OUC	2.0	1	reference to the connection
5	CONNECTION_TYPE	USInt	4.0	19	17: TCP/IP, 18: ISO on TCP
6	ACTIVE_EST	Bool	5.0	false	active/passive connection establishment
7	LOCAL_DEVICE_ID	USInt	6.0	1	1: local IE interface
8	LOCAL_TSAP_ID_L...	USInt	7.0	2	byte length of local TSAP id/port number
9	REM_SUBNET_ID_...	USInt	8.0	0	byte length of remote subnet id
10	REM_STADDR_LEN	USInt	9.0	0	byte length of remote IP address
11	REM_TSAP_ID_LEN	USInt	10.0	0	byte length of remote port/TSAP id
12	NEXT_STADDR_LEN	USInt	11.0	0	byte length of next station address
13	LOCAL_TSAP_ID	Array[1..16] of Byte	12.0		TSAP id/local port number
14	LOCAL_TSAP_I...	Byte		B#16#07	
15	LOCAL_TSAP_I...	Byte		B#16#D0	

The TUSEND instruction sends data through UDP to the remote partner specified in the "TADDR_Param" data type. The TURCV instruction receives data through UDP. After a successful execution of the TURCV instruction, the "TADDR_Param" data type shows the address of the remote partner (the sender), as shown in the figures below.




Send_UDP_ADDR

	Name	Data type	Offset	Initial value	Comment
1	Static				
2	Addr_Data	TADDR_Param	0.0		
3	REM_IP_ADDR	Array[1..4] of USInt	0.0		remote station address
4	REM_IP_ADDR[1]	USInt		192	
5	REM_IP_ADDR[2]	USInt		168	
6	REM_IP_ADDR[3]	USInt		2	
7	REM_IP_ADDR[4]	USInt		10	
8	REM_PORT_NR	UInt	4.0	2000	remote port number
9	RESERVED	Word	6.0	0	unused; has to be 0

11.2.8.16 T_CONFIG

The T_CONFIG instruction changes the IP configuration parameters of the PROFINET port from the user program, allowing the permanent change or setting of the following features:

- Station name
- IP address
- Subnet mask
- Router address

Note

Located in the CPU "Properties", "Ethernet address" page, the "Set IP address using a different method" (Page 710) radio button allows you to change the IP address online or by using the "T_CONFIG" instruction after the program is downloaded. This IP address assignment method is for the CPU only.

Located in the CPU "Properties", "Ethernet address" page, the "Set PROFINET device name using a different method" (Page 711) radio button allows you to change the PROFINET device name online or by using the "T_CONFIG" instruction after the program is downloaded. This PROFINET device name assignment method is for the CPU only.

⚠ WARNING

Changing IP configuration parameter with T_CONFIG causes a CPU restart.

After you use the T_CONFIG to change an IP configuration parameter, the CPU restarts. The CPU goes to STOP mode, performs a warm restart, and returns to RUN mode.

Do not use the T_CONFIG instruction in a production environment. Control devices can fail in an unsafe condition, resulting in unexpected operation of controlled equipment. Such unexpected operations could result in death or serious injury to personnel, and/or damage to equipment.

Ensure that your process will go to a safe state when the CPU performs a warm restart as a result of T_CONFIG instruction execution. A warm restart does not reset memory. A warm restart does initialize all non-retentive system and user data and retains the values of all retentive user data.

Note

You should not attempt to execute more than one T_CONFIG instruction at a time.

Table 11- 44 T_CONFIG instruction

LAD / FBD	SCL	Description

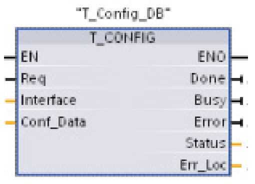
 "T_Config_DB"
T_CONFIG
EN
ENO
Req
Done
Interface
Busy
Conf_Data
Error
Status
Err_Loc
	`"T_CONFIG_DB"(`
	`    req:=_bool_in_,`
	`    interface:=_word_in_,`
	`    conf_Data:=_variant_in_,`
	`    done=>_bool_out_,`
	`    busy=>_bool_out_,`
	`    error=>_bool_out_,`
	`    status=>_dword_out_,`
	`    err_loc=>_word_out_);`
		Use the T_CONFIG instruction to change the IP configuration parameters from your user program.
		T_CONFIG works asynchronously. The execution extends over multiple calls.

Table 11- 45 T_CONFIG data types for the parameters

Parameter and type		Data type	Description
REQ	Input	Bool	Starts the instruction on the rising edge.
INTERFACE	Input	HW_Interface	ID of network interface
CONF_DATA	Input	Variant	Reference to the structure of the configuration data; CONF_DATA is defined by a System Data Type (SDT).
DONE	Output	Bool	• 0: Job has not yet started or is still running. • 1: Job was executed without error.
BUSY	Output	Bool	• 0: The job is complete. • 1: The job is not yet complete. A new job cannot be triggered.
ERROR	Output	Bool	Status parameters with the following values: • 0: No error • 1: Error occurred during processing. STATUS provides detailed information on the type of error.
STATUS	Output	DWord	Status information including error information. (Refer to the Error and Status condition codes in the table below.)
ERR_LOC	Output	DWord	Fault location (field ID and subfield ID of the error parameter)

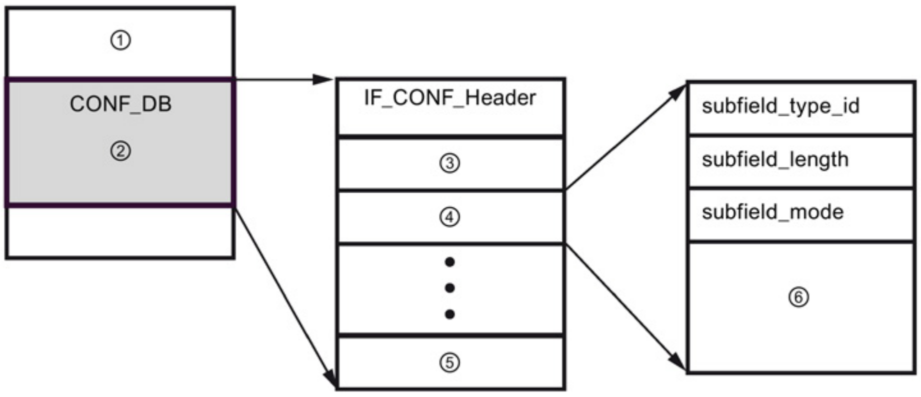
The IP configuration information is placed in the CONF_DATA data block, along with a Variant pointer on parameter CONF_DATA referenced above. The successful execution of the T_CONFIG instruction ends with the handover of the IP configuration data to the network interface. Errors are assigned to the STATUS output parameter.

Table 11- 46 Condition codes for ERROR and STATUS

ERROR	STATUS (DW#16#...)	Description
0	00000000	No error Note: If the instruction executes successfully, the "no error" status may not be returned.
0	00700000	The job is not finished (BUSY = 1).
0	00700100	Start of job execution
0	00700200	Intermediate call (REQ irrelevant)
1	C08xyy00	General failure
1	C0808000	LADDR parameters for identification of the interface are invalid.
1	C0808100	LADDR parameters for identification of the interface have been assigned a non-supported hardware interface.
1	C0808200	CONF_DATA parameter error: Data type of the Variant pointer does not match the data type Byte.
1	C0808300	CONF_DATA parameter error: The area pointer is not in the DB of the Variant pointer.
1	C0808400	CONF_DATA parameter error: The Variant pointer is the wrong length.
1	C0808600	Reserved
1	C0808700	Inconsistency in the CONF_DATA data block length to the IP configuration
1	C0808800	The parameters of the CONF_DATA data block field_type_id are invalid. (Only field_type_id = 0 is allowed.)
1	C0808900	The parameters of the CONF_DATA data block field_type_id are invalid or have been used several times.
1	C0808A00	LEN length of the IP configuration parameters or subfield_cnt errors
1	C0808B00	The IP configuration ID parameter is invalid or unsupported.
1	C0808C00	The Sub-block of the IP configuration is incorrectly placed (Sub-block wrong, wrong order, or used multiple times).
1	C0808D00	The length of a statement LEN Sub-blocks is invalid.
1	C0808E00	The value of the parameter in Sub-blocks mode is invalid.
1	C0808F00	Sub-block conflict between the IP configuration and a previous Sub-block.
1	C0809000	The parameters of the subfield are write-protected (for example: parameters are specified by configuration, or PNIO mode is enabled).
1	C0809100	Reserved
1	C0809400	A parameter in the Sub-block IP configuration has not been defined or may not be used.
1	C0809500	There is an inconsistency between a parameter of the Sub-block IP configuration and other parameters.
1	C080C200	Instruction cannot be executed. This error can occur if, for example, communication with the interface has been lost.
1	C080C300	There are not enough resources. This error can occur if, for example, the instruction is called multiple times with different parameters
1	C080C400	Communication failure. The error can occur temporarily and will require a repeat of the user program.
1	C080D200	Execution of the instruction is not supported by the PROFINET interface.

CONF_DATA Data block

The following diagram shows how the configuration data to be transferred is stored in the configuration DB.



①	Configuration DB	④	Subfield 2
②	Configuration data	⑤	Subfield *n*
③	Subfield 1	⑥	Subfield-specific parameters

The configuration data of CONF_DB consists of a field that contains a header (IF_CONF_Header) and several subfields. IF_CONF_Header provides the following elements:

- field_type_id (data type UInt): Zero
- field_id (data type UInt): Zero
- subfield_cnt (data type UInt): Number of subfields

Each subfield consists of a header (subfield_type_id, subfield_length, subfield_mode) and the subfield-specific parameters. Each subfield must consist of an even number of bytes. The subfield_mode supports a value of 1.

Note

Only one field (IF_CONF_Header) is currently allowed. Its parameters field_type_id and field_id must have the value zero. Other fields with different values for field_type_id and field_id are subject to future extensions.

In the IF_CONF_Header field, only two subfields, "addr" (IP address) and "nos" (Name of station) are currently allowed.

Table 11- 47 Subfields supported

subfield_type_id	Data type	Explanation
30	IF_CONF_V4	IP parameters: IP address, subnet mask, router address
40	IF_CONF_NOS	PROFINET IO device name (Name of station)

Table 11- 48 Elements of the IF_CONF_V4 data type

Name		Data type	Start value	Description
Id		UInt	30	subfield_type_id
len		UInt	18	subfield_length
mode		UInt	1	subfield_mode (1: permanent)
InterfaceAddress		IP_V4	-	Interface address
ADDR		Array [1..4] of Byte		
	ADDR[1]	Byte	b#16#C8	IP address high byte: 200
	ADDR[2]	Byte	b#16#0C	IP address high byte: 12
	ADDR[3]	Byte	b#16#01	IP address low byte: 1
	ADDR[4]	Byte	b#16#90	IP address low byte: 144
SubnetMask		IP_V4	-	Subnet mask
ADDR		Array [1..4] of Byte		
	ADDR[1]	Byte	b#16#FF	Subnet mask high byte: 255
	ADDR[2]	Byte	b#16#FF	Subnet mask high byte: 255
	ADDR[3]	Byte	b#16#FF	Subnet mask low byte: 255
	ADDR[4]	Byte	b#16#00	Subnet mask low byte: 0
DefaultRouter		IP_V4	-	Default router
ADDR		Array [1..4] of Byte		
	ADDR[1]	Byte	b#16#C8	Router high byte: 200
	ADDR[2]	Byte	b#16#0C	Router high byte: 12
	ADDR[3]	Byte	b#16#01	Router low byte: 1
	ADDR[4]	Byte	b#16#01	Router low byte: 1

Table 11- 49 Elements of the IF_CONF_NOS data type

Name	Data type	Start value	Description
id	UInt	40	subfield_type_id
len	UInt	246	subfield_length
mode	UInt	1	subfield_mode (1: permanent)
Nos (Name of station)	Array[1..240] of Byte	0	Station name: You must occupy the ARRAY from the first byte. If the ARRAY is longer than the station name to be assigned, you must enter a zero byte after the actual station name (in conformity with IEC 61158-6-10). Otherwise, nos is rejected and the "T_CONFIG (Page 704)" instruction enters the error code DW#16#C0809400 in STATUS. If you occupy the first byte with zero, the station name is deleted.

The station name is subject to the following limitations:

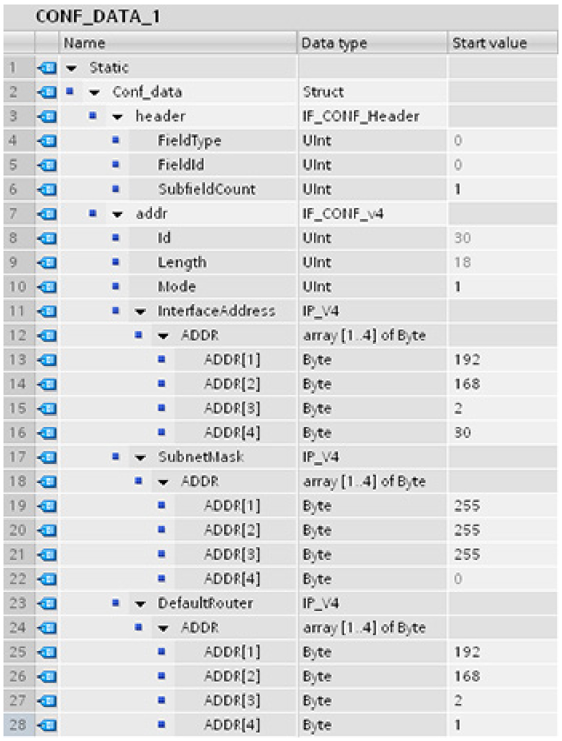
- A name component within the station name, i.e., a character string between two dots, must not exceed 63 characters.
- No special characters such as umlauts, brackets, underscore, slash, blank space, etc. The only special character permitted is the dash.
- The station name must not begin or end with the "-" character.
- The station name must not begin with a number.
- The station name form n.n.n.n (n = 0, ... 999) is not permitted.
- The station name must not begin with the string "port-xyz" or "port-xyz-abcde" (a, b, c, d, e, x, y, z = 0, ... 9).

Note

You can also create an ARRAY "nos" that is shorter then 240 bytes, but not less than 2 bytes. In this case, you must adjust the "len" (length of subfield) tag accordingly.

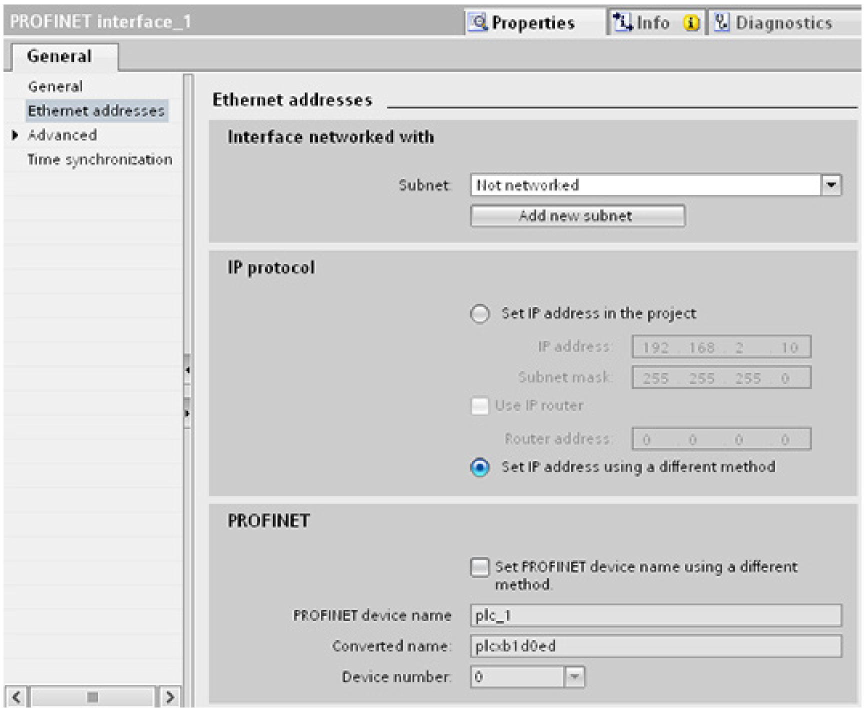
Example: Using the T_CONFIG instruction to change IP parameters

In the following example, in the "addr" subfield, the "InterfaceAddress" (IP address), "SubnetMask", and "DefaultRouter" (IP router) are changed. In the CPU "Properties", "Ethernet address" page, the "Set IP address using a different method" radio button must be clicked to enable you to change the IP address using the "T_CONFIG" instruction after the program is downloaded.



CONF_DATA_1

	Name	Data type	Start value
1	Static		
2	Conf_data	Struct	
3	header	IF_CONF_Header	
4	FieldType	UInt	0
5	FieldId	UInt	0
6	SubfieldCount	UInt	1
7	addr	IF_CONF_v4	
8	Id	UInt	30
9	Length	UInt	18
10	Mode	UInt	1
11	InterfaceAddress	IP_V4	
12	ADDR	array [1..4] of Byte	
13	ADDR[1]	Byte	192
14	ADDR[2]	Byte	168
15	ADDR[3]	Byte	2
16	ADDR[4]	Byte	30
17	SubnetMask	IP_V4	
18	ADDR	array [1..4] of Byte	
19	ADDR[1]	Byte	255
20	ADDR[2]	Byte	255
21	ADDR[3]	Byte	255
22	ADDR[4]	Byte	0
23	DefaultRouter	IP_V4	
24	ADDR	array [1..4] of Byte	
25	ADDR[1]	Byte	192
26	ADDR[2]	Byte	168
27	ADDR[3]	Byte	2
28	ADDR[4]	Byte	1



PROFINET interface_1 | Properties | Info | Diagnostics

General

- General
- Ethernet addresses
- Advanced
- Time synchronization

Ethernet addresses

Interface networked with

Subnet: Not networked

Add new subnet

IP protocol

- ○ Set IP address in the project
  - IP address: 192 . 168 . 2 . 10
  - Subnet mask: 255 . 255 . 255 . 0
  - ☐ Use IP router
  - Router address: 0 . 0 . 0 . 0
- ● Set IP address using a different method

PROFINET

☐ Set PROFINET device name using a different method.

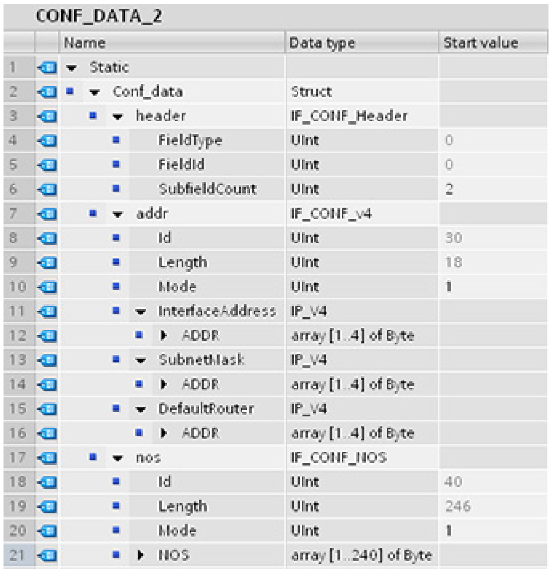
PROFINET device name: plc_1

Converted name: plcxb1d0ed

Device number: 0

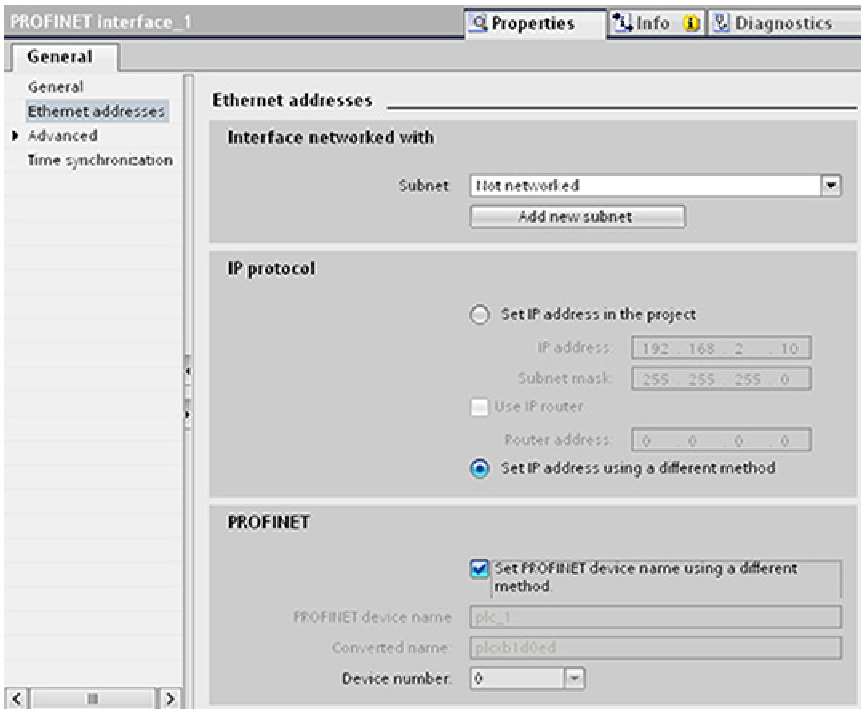
Example: Using the T_CONFIG instruction to change IP parameters and PROFINET IO device names

In the following example, both the "addr" and "nos" (Name of station) subfields are changed. In the CPU "Properties", "Ethernet address" page, the "Set PROFINET device name using a different method" radio button must be clicked to enable you to change the PROFINET device name using the "T_CONFIG" instruction after the program is downloaded.



CONF_DATA_2

	Name	Data type	Start value
1	Static		
2	Conf_data	Struct	
3	header	IF_CONF_Header	
4	FieldType	UInt	0
5	FieldId	UInt	0
6	SubfieldCount	UInt	2
7	addr	IF_CONF_v4	
8	Id	UInt	30
9	Length	UInt	18
10	Mode	UInt	1
11	InterfaceAddress	IP_V4	
12	ADDR	array [1..4] of Byte	
13	SubnetMask	IP_V4	
14	ADDR	array [1..4] of Byte	
15	DefaultRouter	IP_V4	
16	ADDR	array [1..4] of Byte	
17	nos	IF_CONF_NOS	
18	Id	UInt	40
19	Length	UInt	246
20	Mode	UInt	1
21	NOS	array [1..240] of Byte	



Ethernet addresses

Interface networked with

Subnet: Not networked

IP protocol

( ) Set IP address in the project

IP address: 192 . 168 . 2 . 10

Subnet mask: 255 . 255 . 255 . 0

[ ] Use IP router

Router address: 0 . 0 . 0 . 0

(●) Set IP address using a different method

PROFINET

[✓] Set PROFINET device name using a different method.

PROFINET device name: plc_1

Converted name: plcxb1d0ed

Device number: 0

11.2.8.17 Common parameters for instructions

REQ input parameter

Many of the Open User Communication instructions use the REQ input to initiate the operation on a low to high transition. The REQ input must be high (TRUE) for one execution of an instruction, but the REQ input can remain TRUE for as long as desired. The instruction does not initiate another operation until it has been executed with the REQ input FALSE so that the instruction can reset the history state of the REQ input. This is required so that the instruction can detect the low to high transition to initiate the next operation.

When you place one of these instructions in your program, STEP 7 prompts you to identify the instance DB. Use a unique DB for each instruction call. This ensures that each instruction properly handles inputs such as REQ.

ID input parameter

This is a reference to the "Local ID (hex)" on the "Network view" of "Devices and networks" in STEP 7 and is the ID of the network that you want to use for this communication block. The ID must be identical to the associated parameter ID in the local connection description.

DONE, NDR, ERROR, and STATUS output parameters

These instructions provide outputs describing the completion status:

Table 11- 50 Open User Communication instruction output parameters

Parameter	Data type	Default	Description
DONE	Bool	FALSE	Is set TRUE for one execution to indicate that the last request completed without errors; otherwise, FALSE.
NDR	Bool	FALSE	Is set TRUE for one execution to indicate that the requested action has completed without error and new data has been received; otherwise, FALSE.
BUSY	Bool	FALSE	Is set TRUE when active to indicate that: • The job is not yet complete. • A new job cannot be triggered. Is set FALSE when job is complete.
ERROR	Bool	FALSE	Is set TRUE for one execution to indicate that the last request completed with errors, with the applicable error code in STATUS; otherwise, FALSE.
STATUS	Word	0	Result status: • If the DONE or NDR bit is set, then STATUS is set to 0 or to an informational code. • If the ERROR bit is set, then STATUS is set to an error code. • If none of the above bits are set, then the instruction returns status results that describe the current state of the function. STATUS retains its value for the duration of the execution of the function.

Note

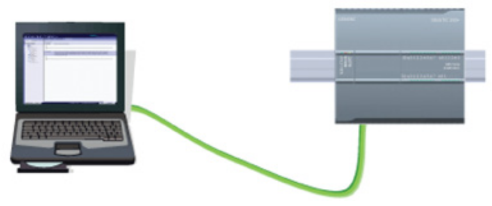
Note that DONE, NDR, and ERROR are set for one execution only.

Restricted TSAPs and port numbers for passive ISO and TCP communication

If you use the "TCON" instruction to set up and establish a passive communications connection, the following port addresses are restricted and should not be used:

- ISO TSAP (passive):
  - 01.00, 01.01, 02.00, 02.01, 03.00, 03.01
  - 10.00, 10.01, 11.00, 11.01, ... BF.00, BF.01
- TCP port (passive): 5001, 102, 123, 20, 21, 25, 34962, 34963, 34964, 80
- UDP port (passive): 161, 34962, 34963, 34964

11.2.9 Communication with a programming device



A CPU can communicate with a STEP 7 programming device on a network.

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

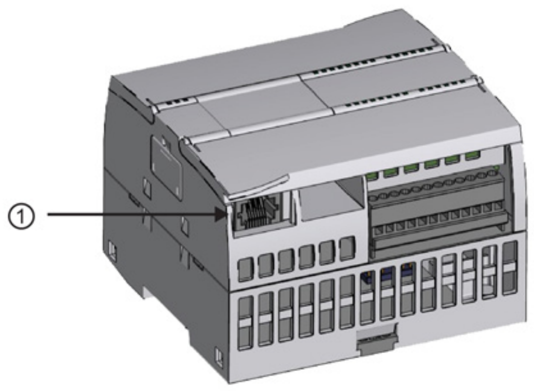
- Configuration/Setup: Hardware configuration is required.
- No Ethernet switch is required for one-to-one communications; an Ethernet switch is required for more than two devices in a network.

11.2.9.1 Establishing the hardware communications connection

The PROFINET 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 (Page 58).
2. Plug the Ethernet cable into the PROFINET port shown below.
3. Connect the Ethernet cable to the programming device.



① PROFINET port

An optional strain relief is available to strengthen the PROFINET connection. For ordering information, see Spare parts and other hardware (Page 58).

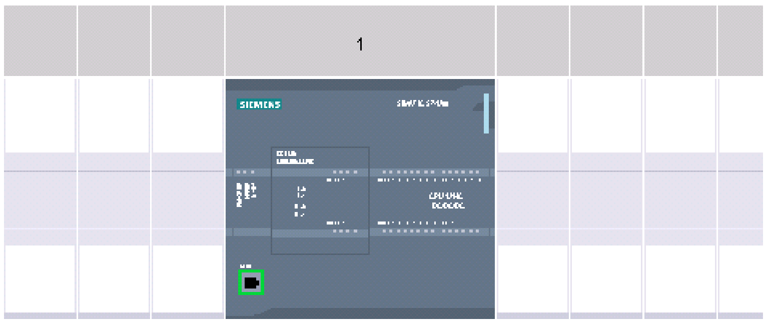
See also

Spare parts and other hardware (Page 1282)

11.2.9.2 Configuring the devices

If you have already created a project with a CPU, open your project in STEP 7.

If not, create a project and insert a CPU (Page 146) into the rack. In the project below, a CPU is shown in the "Device View".



11.2.9.3 Assigning Internet Protocol (IP) addresses

Assigning the IP addresses

In a PROFINET network, each device must also have an Internet Protocol (IP) address. This address allows the device to deliver data on a more complex, routed network:

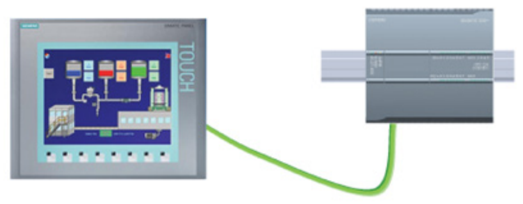
- If you have programming or other network devices that use an on-board adapter card connected to your plant LAN or an Ethernet-to-USB adapter card connected to an isolated network, you must assign IP addresses to them. Refer to "Assigning IP addresses to programming and network devices" (Page 622) for more information.
- You can also assign an IP address to a CPU or network device online. This is particularly useful in an initial device configuration. Refer to "Assigning an IP address to a CPU online" (Page 622) for more information.
- After you have configured your CPU or network device in your project, you can configure parameters for the PROFINET interface, to include its IP address. Refer to "Configuring an IP address for a CPU in your project" (Page 625) for more information.

11.2.9.4 Testing your PROFINET network

After completing the configuration, you must download your project to the CPU. All IP addresses are configured when you download the project.

The CPU "Download to device" function and its "Extended download to device" dialog can show all accessible network devices and whether or not unique IP addresses have been assigned to all devices. Refer to "Testing the PROFINET network" (Page 629) for more information.

11.2.10 HMI-to-PLC communication



The CPU supports PROFINET communications connections to HMIs (Page 30). The following requirements must be considered when setting up communications between CPUs and HMIs:

Configuration/Setup:

- The PROFINET port of the CPU must be configured to connect with the HMI.
- The HMI must be setup and configured.
- The HMI configuration information is part of the CPU project and can be configured and downloaded from within the project.
- No Ethernet switch is required for one-to-one communications; an Ethernet switch is required for more than two devices in a network.

Note

The rack-mounted CSM1277 4-port Ethernet switch can be used to connect your CPUs and HMI devices. The PROFINET port on the CPU does not contain an Ethernet switching device.

Supported functions:

- The HMI can read/write data to the CPU.
- Messages can be triggered, based upon information retrieved from the CPU.
- System diagnostics

Table 11- 51 Required steps in configuring communications between an HMI and a CPU

Step	Task
1	Establishing the hardware communications connection A PROFINET 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. An Ethernet switch is not required to connect an HMI and a CPU. Refer to "Communication with a programming device: Establishing the hardware communications connection" (Page 714) for more information.
2	Configuring the devices Refer to "Communication with a programming device: Configuring the devices" (Page 715) for more information.
3	Configuring the logical network connections between an HMI and a CPU Refer to "HMI-to-PLC communication: Configuring the logical network connections between two devices" (Page 717) for more information.

Step	Task
4	Configuring an IP address in your project Use the same configuration process; however, you must configure IP addresses for the HMI and the CPU. Refer to "Device configuration: Configuring an IP address for a CPU in your project" (Page 626) for more information.
5	Testing the PROFINET network You must download the configuration for each CPU and HMI device. Refer to "Device configuration: Testing the PROFINET network" (Page 629) for more information.

11.2.10.1 Configuring logical network connections between two devices

After you configure the rack with the CPU, you are now ready to configure your network connections.

In the Devices and Networks portal, use the "Network view" to create the network connections between the devices in your project. First, click the "Connections" tab, and then select the connection type with the dropdown, just to the right (for example, an ISO on TCP connection).

To create a PROFINET connection, click the green (PROFINET) box on the first device, and drag a line to the PROFINET box on the second device. Release the mouse button and your PROFINET connection is joined.

Refer to "Device Configuration: Creating a network connection" (Page 618) for more information.

11.2.11 PLC-to-PLC communication

A CPU can communicate with another CPU on a network by using the TSEND_C and TRCV_C instructions.

Consider the following when setting up communications between two CPUs:

- Configuration/Setup: Hardware configuration is required.
- Supported functions: Reading/Writing data to a peer CPU
- No Ethernet switch is required for one-to-one communications; an Ethernet switch is required for more than two devices in a network.

Table 11- 52 Required steps in configuring communications between two CPUs

Step	Task
1	Establishing the hardware communications connection A PROFINET interface establishes the physical connection between two CPUs. Since Auto-Cross-Over functionality is built into the CPU, you can use either a standard or crossover Ethernet cable for the interface. An Ethernet switch is not required to connect the two CPUs. Refer to "Communication with a programming device: Establishing the hardware communications connection" (Page 714) for more information.
2	Configuring the devices You must configure two CPUs in your project. Refer to "Communication with a programming device: Configuring the devices" (Page 715) for more information.
3	Configuring the logical network connections between two CPUs Refer to "PLC-to-PLC communication: Configuring logical network connections between two devices" (Page 718) for more information.
4	Configuring an IP address in your project Use the same configuration process; however, you must configure IP addresses for two CPUs (for example, PLC_1 and PLC_2). Refer to "Device configuration: Configuring an IP address for a CPU in your project" (Page 626) for more information.
5	Configuring transmit (send) and receive parameters You must configure TSEND_C and TRCV_C instructions in both CPUs to enable communications between them. Refer to "Configuring communications between two CPUs: Configuring transmit (send) and receive parameters" (Page 719) for more information.
6	Testing the PROFINET network You must download the configuration for each CPU. Refer to "Device configuration: Testing the PROFINET network" (Page 629) for more information.

11.2.11.1 Configuring logical network connections between two devices

After you configure the rack with the CPU, you are now ready to configure your network connections.

In the Devices and Networks portal, use the "Network view" to create the network connections between the devices in your project. First, click the "Connections" tab, and then select the connection type with the dropdown, just to the right (for example, an ISO on TCP connection).

To create a PROFINET connection, click the green (PROFINET) box on the first device, and drag a line to the PROFINET box on the second device. Release the mouse button and your PROFINET connection is joined.

Refer to "Device Configuration: Creating a network connection" (Page 618) for more information.

11.2.11.2 Configuring the Local/Partner connection path between two devices

Configuring General parameters

You specify the communication parameters in the "Properties" configuration dialog of the communication instruction. This dialog appears near the bottom of the page whenever you have selected any part of the instruction.

Refer to "Device configuration: Configuring the Local/Partner connection path (Page 619)" for more information.

In the "Address Details" section of the Connection parameters dialog, you define the TSAPs or ports to be used. The TSAP or port of a connection in the CPU is entered in the "Local TSAP" field. The TSAP or port assigned for the connection in your partner CPU is entered under the "Partner TSAP" field.

11.2.11.3 Configuring transmit (send) and receive parameters

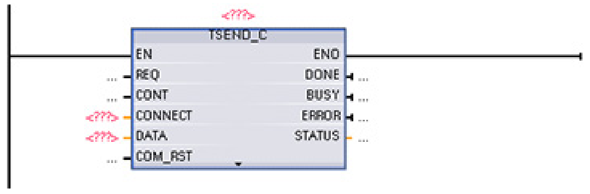
Communication blocks (for example, TSEND_C and TRCV_C) are used to establish connections between two CPUs. Before the CPUs can engage in PROFINET communications, you must configure parameters for transmitting (or sending) messages and receiving messages. These parameters dictate how communications operate when messages are being transmitted to or received from a target device.

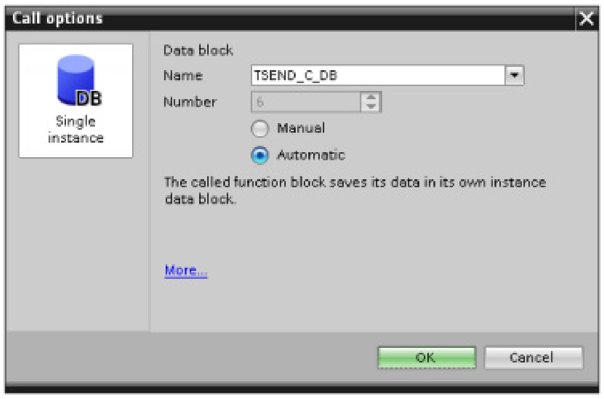
Configuring the TSEND_C instruction transmit (send) parameters

TSEND_C instruction

The TSEND_C instruction (Page 645) creates a communications connection to a partner station. The connection is set up, established, and automatically monitored until it is commanded to disconnect by the instruction. The TSEND_C instruction combines the functions of the TCON, TDISCON and TSEND instructions.

From the Device configuration in STEP 7, you can configure how a TSEND_C instruction transmits data. To begin, you insert the instruction into the program from the "Communications" folder in the "Instructions" task card. The TSEND_C instruction is displayed, along with the Call options dialog where you assign a DB for storing the parameters of the instruction.




Call options

Single instance

Data block

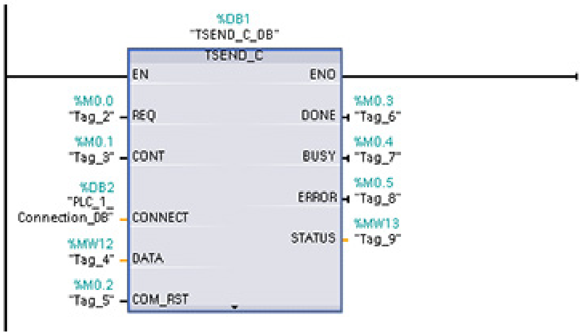
Name: TSEND_C_DB

Number: 6

Manual [✓ Automatic]

The called function block saves its data in its own instance data block.

You can assign tag memory locations to the inputs and outputs, as shown in the following figure:



Configuring General parameters

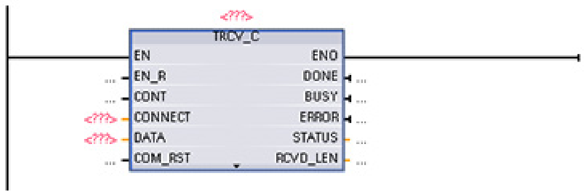
You specify the parameters in the Properties configuration dialog of the TSEND_C instruction. This dialog appears near the bottom of the page whenever you have selected any part of the TSEND_C instruction.

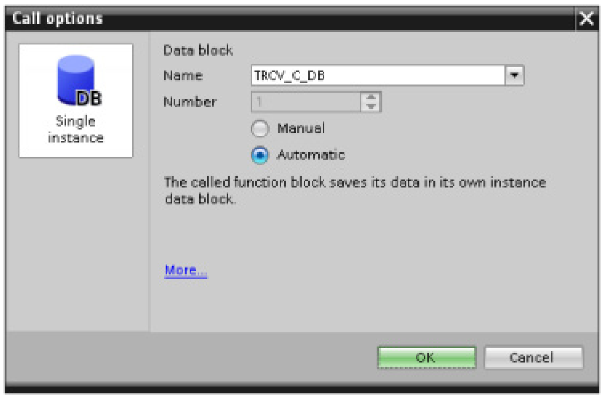
Configuring the TRCV_C instruction receive parameters

TRCV_C instruction

The TRCV_C instruction (Page 645) creates a communications connection to a partner station. The connection is set up, established, and automatically monitored until it is commanded to disconnect by the instruction. The TRCV_C instruction combines the functions of the TCON, TDISCON, and TRCV instructions.

From the CPU configuration in STEP 7, you can configure how a TRCV_C instruction receives data. To begin, insert the instruction into the program from the "Communications" folder in the "Instructions" task card. The TRCV_C instruction is displayed, along with the Call options dialog where you assign a DB for storing the parameters of the instruction.




Call options

Single instance

Data block

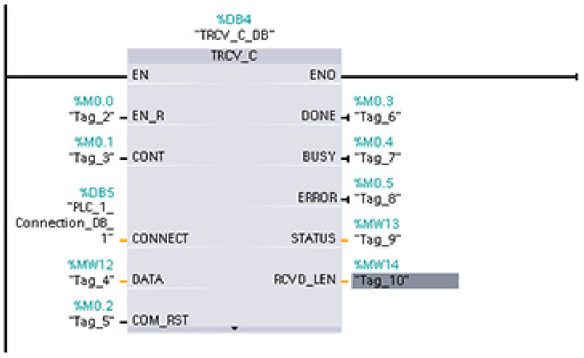
Name: TRCV_C_DB

Number: 1

Manual [✓ Automatic]

The called function block saves its data in its own instance data block.

You can assign tag memory locations to the inputs and outputs, as shown in the following figure:



Configuring the General parameters

You specify the parameters in the Properties configuration dialog of the TRCV_C instruction. This dialog appears near the bottom of the page whenever you have selected any part of the TRCV_C instruction.

11.2.12 Configuring a CPU and PROFINET IO device

11.2.12.1 Adding a PROFINET IO device

Adding a PROFINET IO device

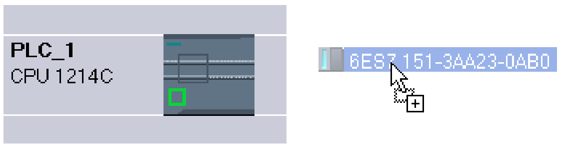
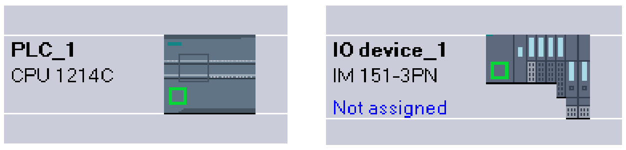
In the "Devices and networks" portal, use the hardware catalog to add PROFINET IO devices.

Note

To add a PROFINET IO device, you can use STEP 7 Professional or Basic, V11 or greater.

For example, expand the following containers in the hardware catalog to add an ET200S IO device: Distributed I/O, ET200S, Interface modules, and PROFINET. You can then select the interface module from the list of ET200S devices (sorted by part number) and add the ET200S IO device.

Table 11- 53 Adding an ET200S IO device to the device configuration

Insert the IO device	Result
	 PLC_1 CPU 1214C IO device_1 IM 151-3PN Not assigned

You can now connect the PROFINET IO device to the CPU:

1. Right-click the "Not assigned" link on the device and select "Assign new IO controller" from the context menu to display the "Select IO controller" dialog.
2. Select your S7-1200 CPU (in this example, "PLC_1") from the list of IO controllers in the project.
3. Click "OK" to create the network connection.

11.2.12.2 Configuring logical network connections between a CPU and a PROFINET IO device

Configuring logical network connections

After you configure the rack with the CPU, you are now ready to configure your network connections.

In the "Devices and networks" portal, use the "Network view" to create the network connections between the devices in your project. To create a PROFINET connection, click the green (PROFINET) box on the first device, and drag a line to the PROFINET box on the second device. Release the mouse button and your PROFINET connection is joined.

Refer to "Device Configuration: Creating a network connection" (Page 618) for more information.

11.2.12.3 Assigning CPUs and device names

Assigning CPUs and device names

Network connections between the devices also assign the PROFINET IO device to the CPU, which is required for that CPU to control the device. To change this assignment, click the PLC Name shown on the PROFINET IO device. A dialog box opens that allows the PROFINET IO device to be disconnected from the current CPU and reassigned or left unassigned, if desired.

The devices on your PROFINET network must have an assigned name before you can connect with the CPU. Use the "Network view" to assign names to your PROFINET devices if the devices have not already been assigned a name or if the name of the device is to be changed. Right-click the PROFINET IO device and select "Assign device name" to do this.

For each PROFINET IO device, you must assign the same name to that device in both the STEP 7 project and, using the "Online & diagnostics" tool, to the PROFINET IO device configuration memory (for example, an ET200 S interface module configuration memory). If a name is missing or does not match in either location, the PROFINET IO data exchange mode will not run. Refer to "Online and diagnostic tools: Assigning a name to a PROFINET device online (Page 1070)" for more information.

11.2.12.4 Assigning Internet Protocol (IP) addresses

Assigning the IP addresses

In a PROFINET network, each device must also have an Internet Protocol (IP) address. This address allows the device to deliver data on a more complex, routed network:

- If you have programming or other network devices that use an on-board adapter card connected to your plant LAN or an Ethernet-to-USB adapter card connected to an isolated network, you must assign IP addresses to them. Refer to "Assigning IP addresses to programming and network devices" (Page 622) for more information.
- You can also assign an IP address to a CPU or network device online. This is particularly useful in an initial device configuration. Refer to "Assigning an IP address to a CPU online" (Page 625) for more information.
- After you have configured your CPU or network device in your project, you can configure parameters for the PROFINET interface, to include its IP address. Refer to "Configuring an IP address for a CPU in your project" (Page 626) for more information.

11.2.12.5 Configuring the IO cycle time

Configuring the IO cycle time

A PROFINET IO device is supplied with new data from the CPU within an "IO cycle" time period. The update time can be separately configured for each device and determines the time interval in which data is transmitted from the CPU to and from the device.

STEP 7 calculates the "IO cycle" update time automatically in the default setting for each device of the PROFINET network, taking into account the volume of data to be exchanged and the number of devices assigned to this controller. If you do not want to have the update time calculated automatically, you can change this setting.

You specify the "IO cycle" parameters in the "Properties" configuration dialog of the PROFINET IO device. This dialog appears near the bottom of the page whenever you have selected any part of the instruction.

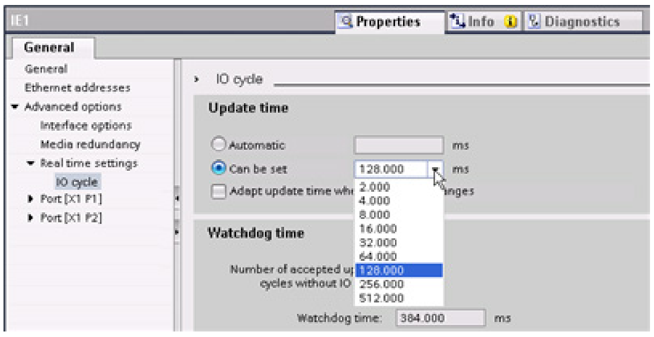
In the "Device view" of the PROFINET IO device, click the PROFINET port. In the "PROFINET Interface" dialog, access the "IO cycle" parameters with the following menu selections:

- "Advanced options"
- "Realtime settings"
- "IO cycle"

Define the IO cycle "Update time" with the following selections:

- To have a suitable update time calculated automatically, select "Automatic".
- To set the update yourself, select "Can be set" and enter the required update time in ms.
- To ensure consistency between the send clock and the update time, activate the "Adapt update time when send clock changes" option. This option ensures that the update time is not set to less than the send clock.

Table 11- 54 Configuring the ET200S PROFINET IO cycle time

ET200 S PROFINET IO device	ET200S PROFINET IO cycle dialog
	 IE1 Properties Info Diagnostics General General Ethernet addresses Advanced options Interface options Media redundancy Real time settings IO cycle Port [X1 P1] Port [X1 P2] IO cycle **Update time** Automatic: ms; [✓ Can be set]: 128.000 ms (2.000, 4.000, 8.000, 16.000, 32.000, 64.000, 128.000, 256.000, 512.000); Adapt update time wh… …nges. **Watchdog time** Number of accepted u… cycles without IO; Watchdog time: 384.000 ms

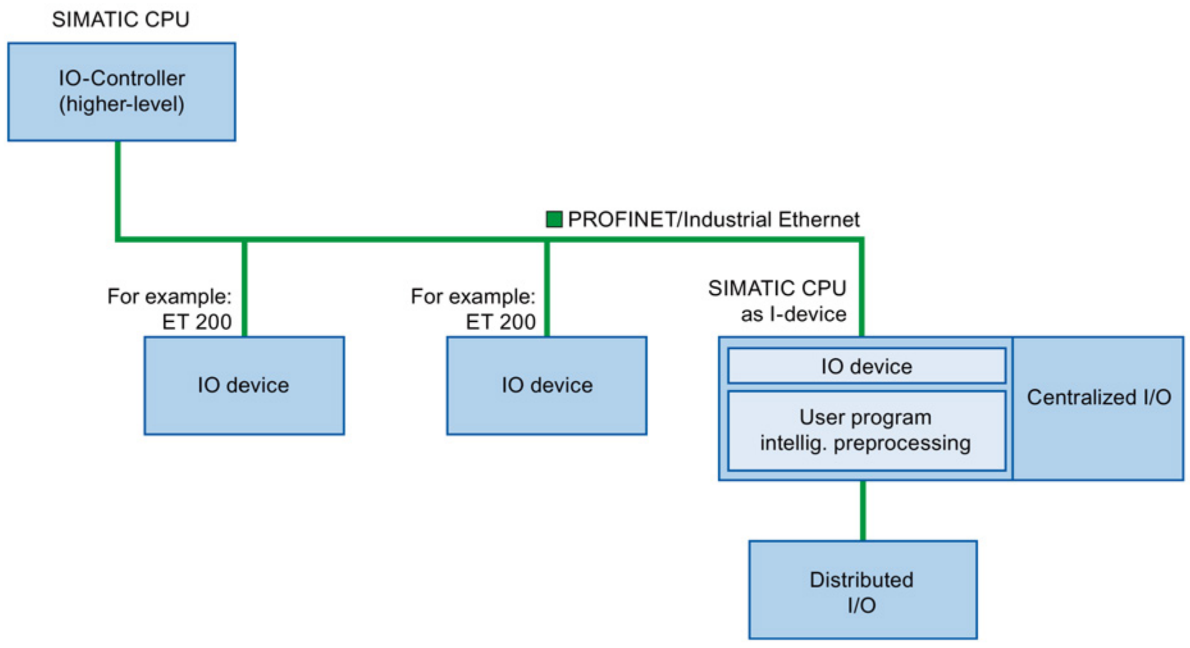
① PROFINET port

11.2.13 Configuring a CPU and PROFINET I-device

11.2.13.1 I-device functionality

The "I-device" (intelligent IO device) functionality of a CPU facilitates data exchange with an IO controller and operation of the CPU as intelligent preprocessing unit of sub processes, for example. The I-device is linked as an IO device to a "higher-level" IO controller.

The pre-processing is handled by the user program on the CPU. The process values acquired in the centralized or distributed (PROFINET IO or PROFIBUS DP) I/O are pre-processed by the user program and made available through a PROFINET IO interface to the CPU of a higher-level station.



"I-device" naming conventions

In the remainder of this description, a CPU or a CP with I-device functionality is simply called an "I-device".

11.2.13.2 Properties and advantages of the I-device

Fields of application

Fields of application of the I-device:

- Distributed processing:

  A complex automation task can be divided into smaller units/subprocesses. This results in manageable processes which lead to simplified subtasks.
- Separating subprocesses:

  Complicated, widely distributed and extensive processes can be subdivided into several subprocesses with manageable interfaces by using I-devices. These subprocesses can be stored in individual STEP 7 projects if necessary, which can later be merged to form one master project.
- Know-how protection:

  Components can only be delivered with a GSD file for the I-device interface description instead of with a STEP 7 project. The user can protect his program since it no longer has to be published.

Properties

Properties of the I-device:

- Unlinking STEP 7 projects:

  Creators and users of an I-device can have completely separated STEP 7 automation projects. The GSD file forms the interface between the STEP 7 projects. This allows a link to standard IO controllers through a standardized interface.
- Real-time communication:

  The I-device is provided with a deterministic PROFINET IO system through a PROFINET IO interface and therefore supports RT (real-time communication) and IRT (isochronous real time).

Advantages

The I-device has the following advantages:

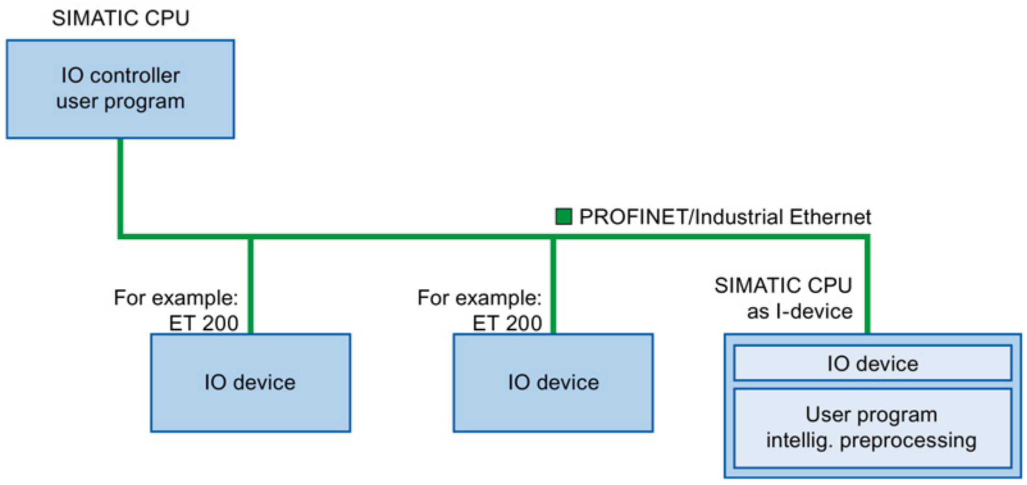
- Simple linking of IO controllers
- Real-time communication between IO controllers
- Relieving the IO controller by distributing the computing capacity to I-devices.
- Lower communications load by processing process data locally.
- Manageable, due to processing of subtasks in separate STEP 7 projects

11.2.13.3 Characteristics of an I-device

An I-device is included in an IO system like a standard IO device.

I-device without lower-level PROFINET IO system

The I-device does not have its own distributed I/O. The configuration and parameter assignment of the I-devices in the role of an IO device is the same as for a distributed I/O system (for example, ET 200).



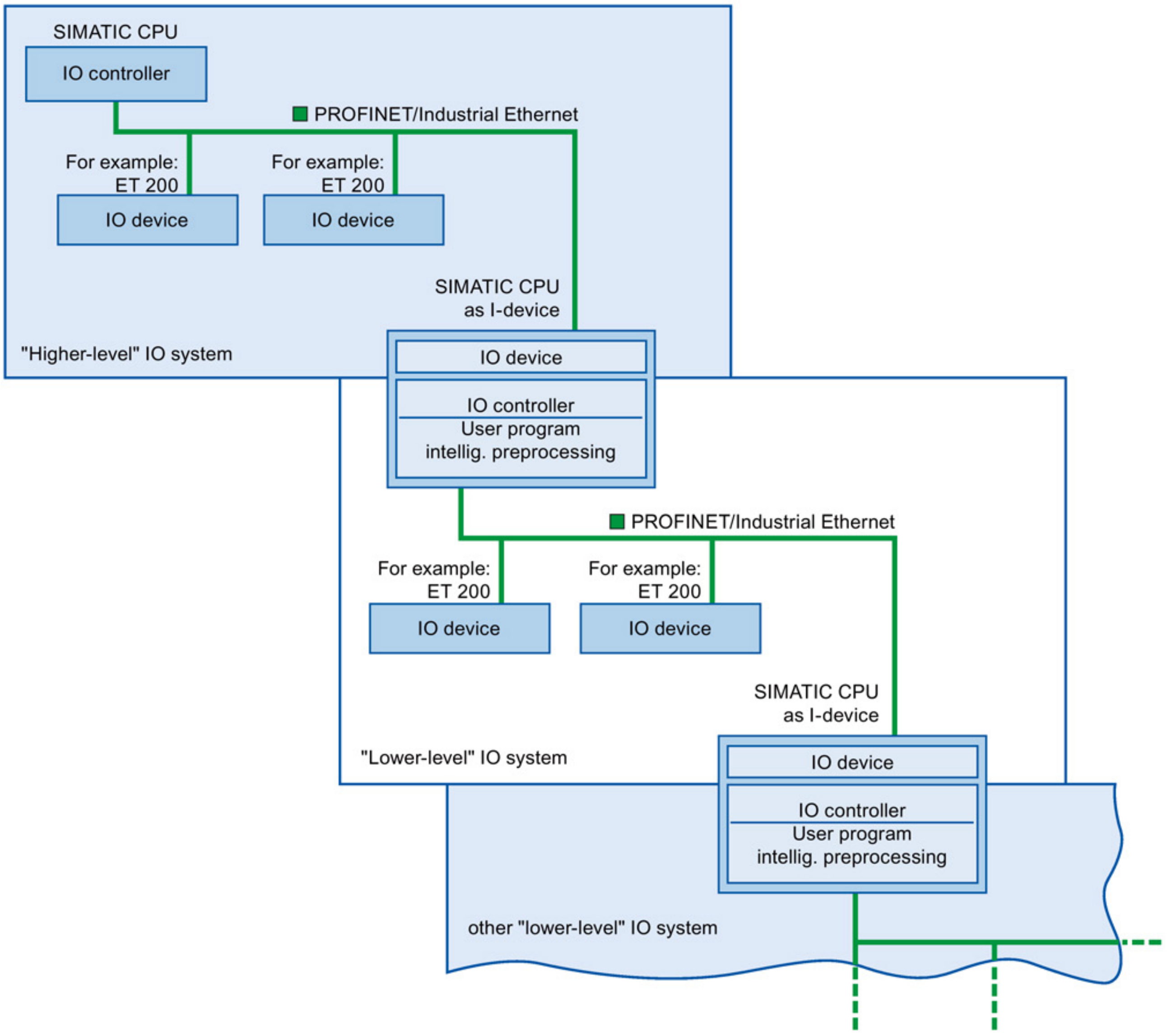
I-device with lower-level PROFINET IO system

Depending on the configuration, an I-device can also be an IO controller on a PROFINET interface in addition to having the role of an IO device.

This means that the I-device can be part of a higher-level IO system through its PROFINET interface and as an IO controller can support its own lower-level IO system.

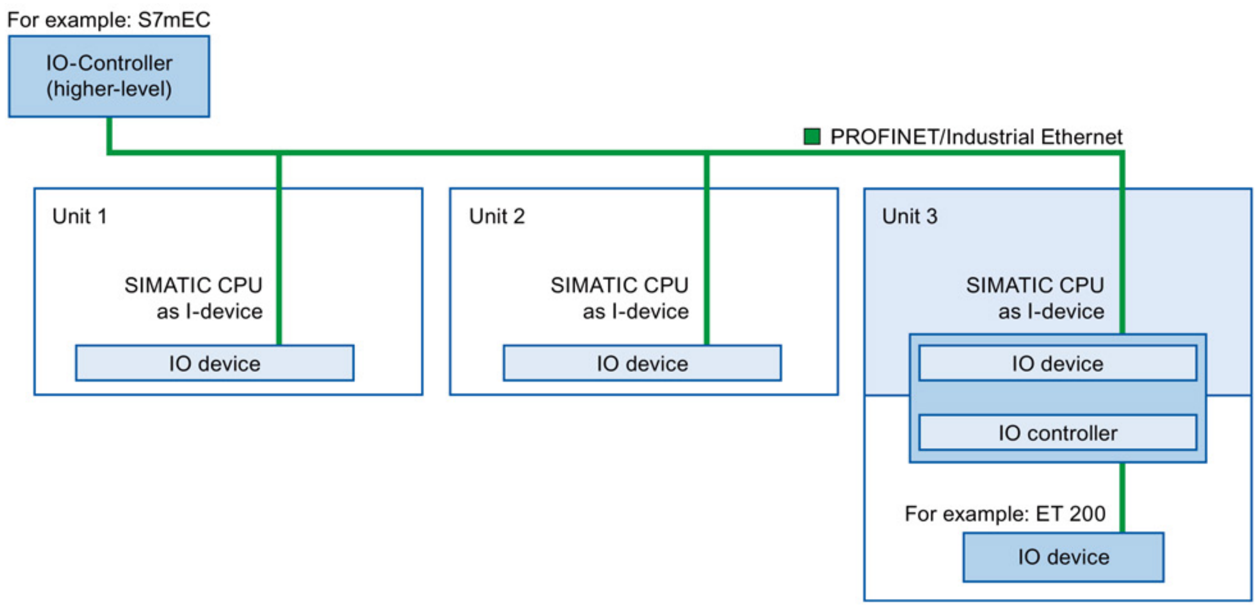
The lower-level IO system can, in turn, contain I-devices (see figure below). This makes hierarchically structured IO systems possible.

In addition to its role as IO controller, an I-device can also be used through a PROFIBUS interface as DP master for a lower-level PROFIBUS system.



Example: I-device as IO device and IO controller

The I-device as IO device and IO controller is explained based on the example of a print process. The I-device controls a unit (a subprocess). One unit is used, for example, to insert additional sheets such as flyers or brochures in a package of printed material.



Unit 1 and unit 2 each consist of an I-device with centralized I/O. The I-device along with the distributed I/O system (for example, ET 200) forms unit 3.

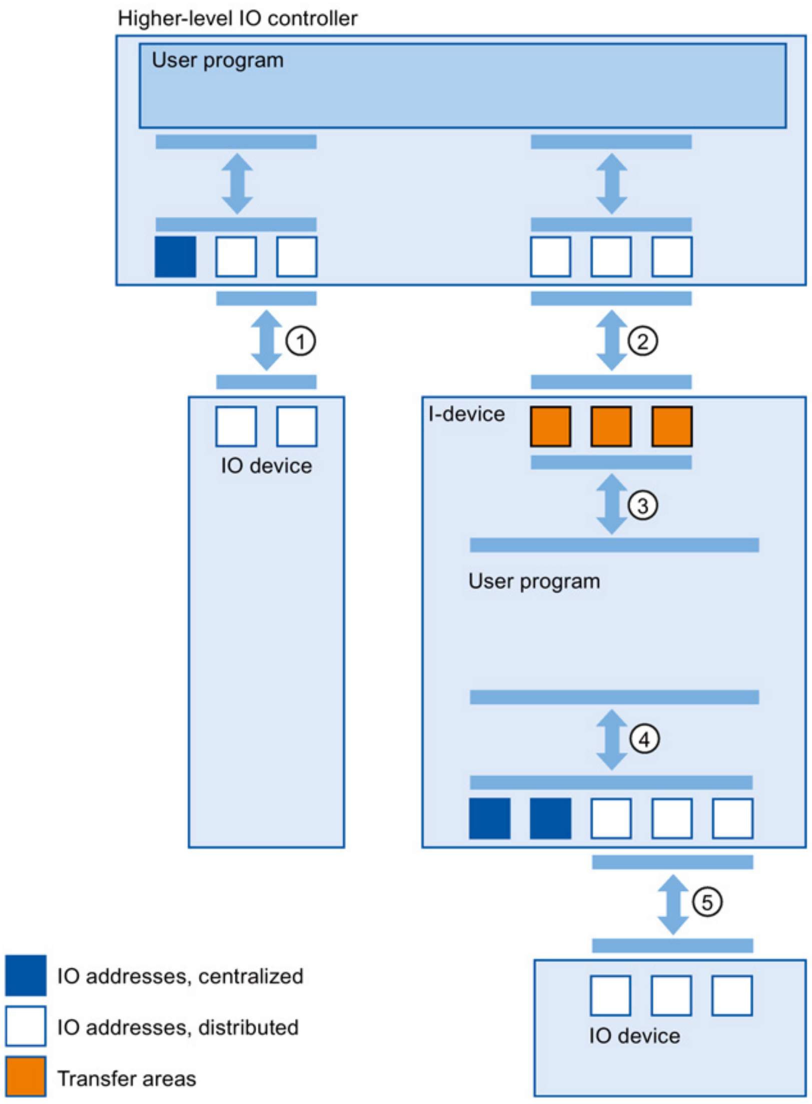
The user program on the I-device is responsible for preprocessing the process data. For this task, the user program of the I-device requires default settings (for example, control data) from the higher-level IO controller. The I-device provides the higher-level IO controller with the results (for example, status of its subtask).

11.2.13.4 Data exchange between higher- and lower-level IO system

Transfer areas are an interface to the user program of the I-device CPU. Inputs are processed in the user program and outputs are the result of the processing in the user program.

The data for communication between IO controller and I-device is made available in the transfer areas. A transfer area contains an information unit that is exchanged consistently between IO controller and I-device. You can find more information on configuration and use of transfer areas in the section "Configuring the I-device".

The next figure shows the data exchange between the higher- and lower-level IO system. The individual communication relations are explained below based upon the numbers:


Higher-level IO controller
User program
①
②
IO device
I-device
③
User program
④
⑤
IO device
IO addresses, centralized
IO addresses, distributed
Transfer areas

① **Data exchange between higher-level IO controller and normal IO-device**

In this way, the IO controller and IO devices exchange data through PROFINET.

② **Data exchange between higher-level IO controller and I-device**

In this way, the IO controller and the I-device exchange data through PROFINET.

The data exchange between a higher-level IO controller and an I-device is based upon the conventional IO controller / IO device relationship.

For the higher-level IO controller, the transfer areas of the I-devices represent submodules of a pre-configured station.

The output data of the IO controller is the input data of the I-device. Analogously, the input data of the IO controller is the output data of the I-device.

③ **Transfer relationship between the user program and the transfer area**

In this way, the user program and the transfer area exchange input and output data.

④ **Data exchange between the user program and the I/O of the I-device**

In this way, the user program and the centralized / distributed I/O exchange input and output data.

⑤ **Data exchange between the I-device and a lower-level IO device**

In this way, the I-device and its IO devices exchange data. The data transfer is through PROFINET.

11.2.13.5 Configuring the I-device

There are basically two possibilities for configuration:

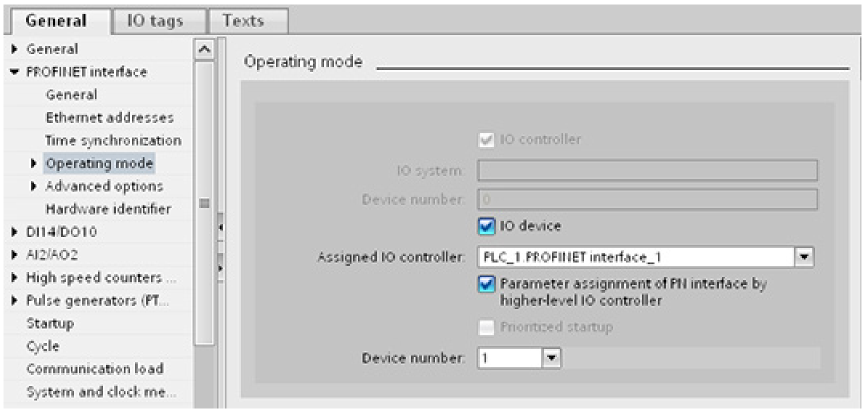
- Configuration of an I-device within a project
- Configuration of an I-device that is used in another project or in another engineering system.

STEP 7 allows you to configure an I-device for another project or for another engineering system by exporting a configured I-device to a GSD file. You import the GSD file in other projects or engineering systems as with other GSD files. The transfer areas for the data exchange, among other data, are stored in this GSD file.

Configuration of an I-device within a project

1. Drag-and-drop a PROFINET CPU from the hardware catalog into the network view.
2. Drag-and-drop a PROFINET CPU, which can also be configured as an IO device, from the hardware catalog into the network view. This device is configured as an I-device (for example, CPU 1215C).
3. Select the PROFINET interface for the I-device.
4. In the Inspector window in the area navigation choose "Operating mode" and select the check box "IO device".
5. Now you have the option of choosing the IO controller in the "Assigned IO controller" drop-down list.

   Once you have chosen the IO controller, the networking and the IO system between both devices are displayed in the network view.



Operating mode

☑ IO controller

IO system:

Device number: 0

☑ IO device

Assigned IO controller: PLC_1.PROFINET interface_1

☑ Parameter assignment of PN interface by higher-level IO controller

☐ Prioritized startup

Device number: 1

6. With the "Parameter assignment of PN interface by higher-level IO controller" check box, you specify whether the interface parameters will be assigned by the I-device itself or by a higher-level IO controller.

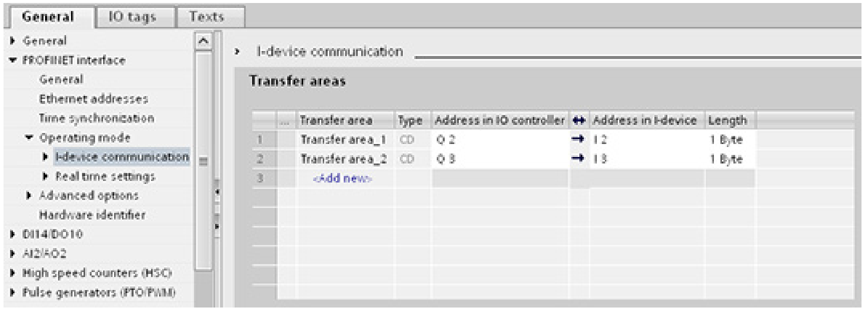
   If you operate the I-device with a lower-level IO system, then the parameters of the I-device PROFINET interface (for example, port parameter) cannot be assigned with the higher-level IO controller.

7. Configure the transfer areas. The transfer areas are found in the area navigation section "I-device communication":

   – Click in the first field of the "Transfer area" column. STEP 7 assigns a default name which you can change.

   – Select the type of communication relation: you can currently only select CD or F-CD.

   – Addresses are automatically preset; you can correct addresses if necessary, and determine the length of the transfer area which is to be consistently transferred.



Transfer areas

	...	Transfer area	Type	Address in IO controller	↔	Address in I-device	Length
1		Transfer area_1	CD	Q 2	→	I 2	1 Byte
2		Transfer area_2	CD	Q 3	→	I 3	1 Byte
3		<Add new>					

8. A separate entry is created in the area navigation for each transfer area. If you select one of these entries, you can adjust the details of the transfer area, or correct them and comment on them.

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.

However, 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.

The "Export" button is found in the "I-device communication" section of the Inspector window.

The hardware configuration is compiled and the export dialog opened.

Assign a name for the I-device proxy as well as a description in the fields provided. Click the "Export" button to complete your process.

Finally, import the GSD file, for example, in another project.

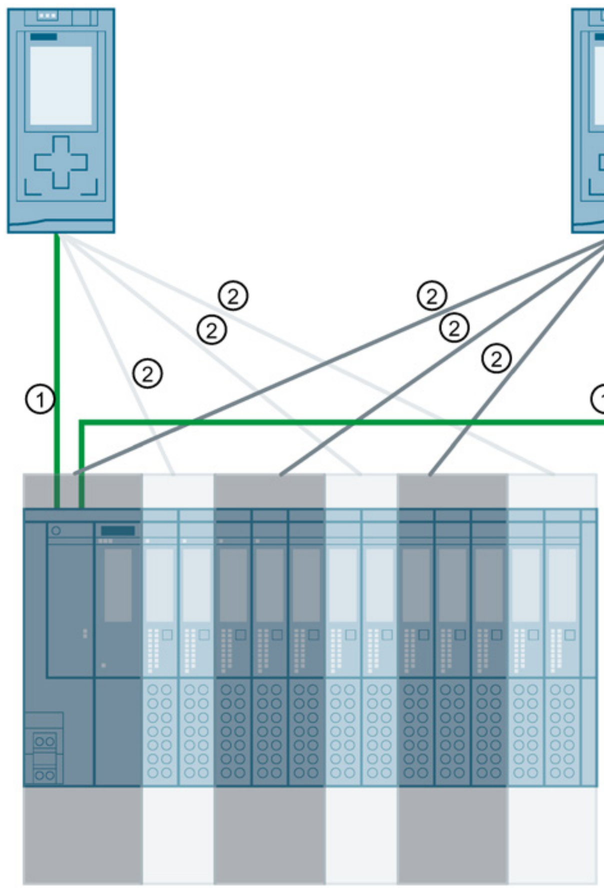
11.2.14 Shared devices

11.2.14.1 Shared device functionality

Numerous IO controllers are often used in larger or widely distributed systems.

Without the "Shared Device" function, each I/O module of an IO device is assigned to the same IO controller. If sensors that are physically close to each other must provide data to different IO controllers, several IO devices are required.

The "Shared Device" function allows the modules or submodules of an IO device to be divided up among different IO controllers. This allows flexible automation concepts. You have, for example, the possibility of combining I/O modules lying near each other into an IO device.



① PROFINET

② Logical assignment

Principle

Access to the submodules of the shared device is then divided up among the individual IO controllers. Each submodule of the shared device is assigned exclusively to one IO controller.

Requirement (GSD configuration)

- STEP 7 V12 Service Pack 1 or higher
- CPU as of FW 1.1 as IO controller
- IO device supports the shared device function, e.g. interface module IM 155-5 PN ST
- GSD file for configuring the IO device is installed
- An S7-1200 CPU configured as an I-device supports the Shared Device functionality. You must export the PROFINET GSD file for the I-device from STEP 7 (as of V5.5) and then import it into STEP 7 (TIA Portal).

Configuring the access

The IO device must be present in several projects so that the modules or submodules of an IO device can be assigned to different IO controllers. A separate project is required for each IO controller.

You use the "Shared device" parameter of the interface module to determine the modules or submodules to which the IO controller has access:

- If the local IO controller has access to the configured module, select the name of theIO controller from the list.
- If the IO controller from a different project and not the local IO controller is to have access to the configured module, select the entry "---".

The configuration is consistent regarding access if each module or submodule in exactly one project is assigned to an IO controller.

Module or submodule is assigned to another IO controller

The paragraph below describes the consequences of the "---" setting of the "Shared device" parameter from the point of view of the local IO controller.

In this case, the local IO controller does not have access to the module configured in this way. Specifically, this means:

- No data exchange with the module or submodule
- No reception of alarms or diagnostics, which means no display of the diagnostics status in the online view
- No parameter assignment of the module or submodule

Setting of the real-time properties

STEP 7 calculates the communication load and thus the resulting update times. You must enter the number of project-external IO controllers in the project in which thePROFINET interface of the shared device is assigned to the IO controller so that a calculation is possible with shared device configurations.

The maximum possible number of IO controllers for the shared device depends on the device. This number is stored in the GSD file of the shared device.

You can set a very short send clock with a CPU as IO controller. The send clock can be shorter than the shortest send clock supported by the shared device. In this case, the shared device is operated by the IO controller with a send clock that it supports (send clock adaptation).

Example: A CPU supports send clocks starting from 0.25 ms. A configured IO device also supports send clocks starting at 0.25 ms; another IO device supports send clocks starting at 1 ms. In this case, you have the option of setting the short send clock of 0.25 ms for the CPU. The CPU operates the "slow" IO device with the send clock of 1 ms, for example.

Rules for the configuration

- IO controllers that use the shared device are created in different projects. In each project, care must be taken that the shared device is configured identically in each station. Only one IO controller may ever have full access to a submodule. Inconsistencies in the configuration result in a failure of the shared device.
- I/O addresses of a module or submodule can only be edited if a module or submodule is assigned to the IO controller in the same project.
- The shared device must have the same IP parameters and the same device name in each project.
- The send clock must be identical for all IO controllers that have access to the shared device.
- The S7 subnet ID of the subnet to which the shared device is connected must be identical in all projects.
- The following functions are only available if the PROFINET interface of the shared device is assigned to the local IO controller:
  – IRT operation
  – Prioritized startup
  – Parameter assignment of the port properties

Boundary conditions

The following boundary conditions result because a shared device configuration is distributed across several projects:

- The addresses of modules or submodules that are not assigned to this IO controller are missing in the address overview of each IO controller that has access to a shared device.
- The modules or submodules that are not assigned are not taken into consideration in the configuration limit calculation for the shared device during the consistency check. For this reason, you must verify for yourself that the maximum number of submodules or the maximum amount of cyclic IO data for the shared device will not be exceeded. For information on the maximum quantities, refer to the documentation for the devices you are using.
- Configuration errors such as the assignment of a module or submodule to several IO controllers are not detected in STEP 7.
- CPUs that are loaded with a shared device configuration do not have any information on whether the IO device is a shared device. Modules or submodules that are assigned to other IO controllers and therefore other CPUs are missing in the loaded configuration. These modules or submodules are therefore displayed neither in the CPU web server nor in the CPU display.

11.2.14.2 Example: Configuring a shared device (GSD configuration)

This example describes how to configure a distributed I/O system as a shared device with STEP 7 V13 SP1 or higher.

A "distributed" configuration with different engineering tools for different IO controller families is possible. The procedure described below is based on STEP 7 as of V13 SP1 and is limited to configuration with two IO controllers of the S7-1200 series that share one shared device.

The example creates two projects with one IO controller each:

- Controller1
- Controller2

You must create the shared device in both projects, even though it is physically one and the same IO device.

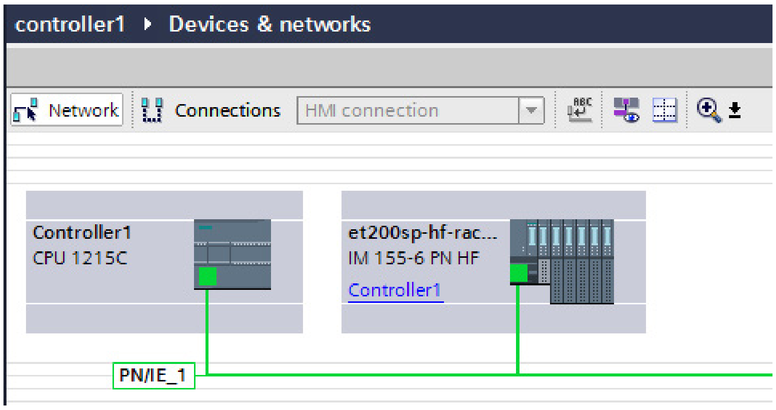
Requirements

- STEP 7 V13 SP1 or higher
- IO device supports shared device functionality (for example, ET 200SP IM 155-6 PN HF V3.1).
- GSD file for configuring the IO device as a shared device is installed.

Procedure: Creating project 1

To create the first project with a shared device, follow these steps:

1. Start STEP 7.
2. Create a new project with the name "Controller1".
3. Insert a CPU 1215C from the hardware catalog in the network view. Name it "Controller1".
4. Insert an IO device with the "Shared Device" function (for example, an ET 200SP) from the hardware catalog (hardware catalog: Other field devices > PROFINET IO > I/O).
5. Assign the IO controller "Controller1" to the IO device.



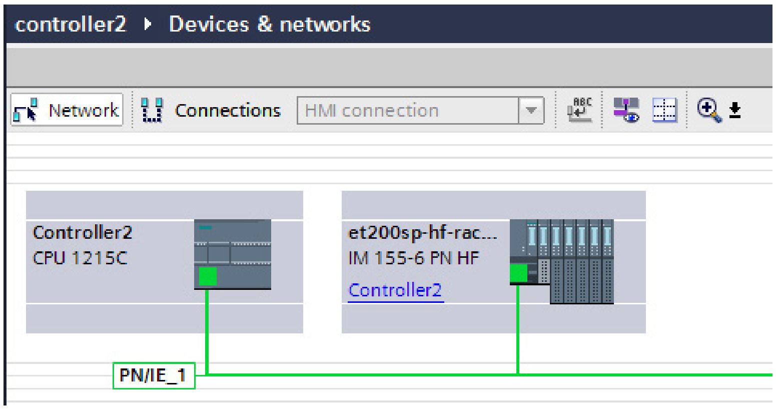
6. Double-click the IO device and insert all required modules and submodules from the hardware catalog in the device overview table.
7. Assign the module parameters.
8. Save the project.

Procedure: Creating project 2

To create the second project with a shared device, follow these steps:

1. Start STEP 7 once again.

   A new instance of STEP 7 opens.
2. In the new instance, create a new project with the name "Controller2".
3. Insert a CPU 1215C in the network view. Name it "Controller2".
4. Copy the IO device from the project "Controller1" and insert it in the network view of project "Controller2".
5. Assign the IO controller "Controller2" to the IO device.



6. Save the project.

Both projects now have an identically structured IO device that must be configured in the next step for the different types of IO controller access.

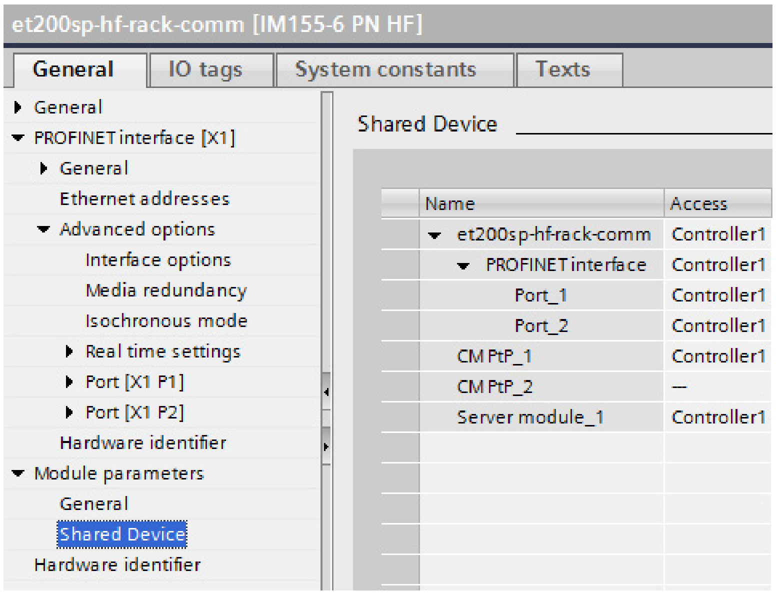
Procedure: Configuring access to the shared device

The modules and submodules you insert in the shared device are automatically assigned to the local CPU. To change the assignment, follow these steps:

1. Select the interface module in the network or device view of project "Controller1".
2. Select the "Shared Device" area in the Inspector window.

   A table shows which CPU has access to the respective module or submodule for all configured modules. The default setting is that the local CPU has access to all modules and submodules.
3. Keep the "Controller1" setting for all modules and submodules that are to remain in the address range of the local CPU.

   Select the setting "---" for all modules and submodules that are to be located in the address range of the CPU from the "Controller2" project (Controller2). This means that an IO controller outside the project is to have access to the module or submodule.

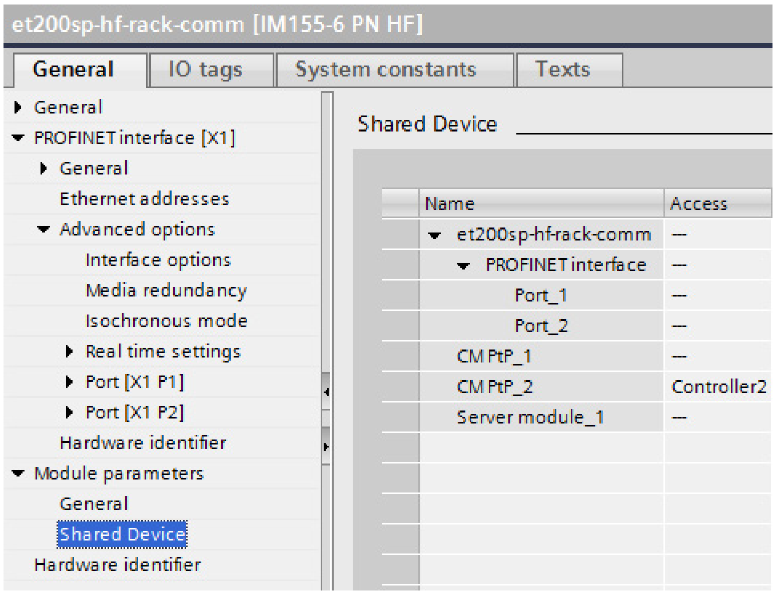


Name	Access
et200sp-hf-rack-comm	Controller1
PROFINET interface	Controller1
Port_1	Controller1
Port_2	Controller1
CM PtP_1	Controller1
CM PtP_2	---
Server module_1	Controller1

4. Select the interface module in the network or device view of project "Controller2".
5. Select the "Shared Device" area in the Inspector window.

   A table shows which CPU has access to the respective module or submodule for all configured modules.

6. Select the setting "---" for all modules and submodules that are to be located in the address range of the CPU from the "Controller1" project (Controller1).



Name	Access
et200sp-hf-rack-comm	—
PROFINET interface	—
Port_1	—
Port_2	—
CM PtP_1	—
CM PtP_2	Controller2
Server module_1	—

7. Finally, check whether the settings for access are "complementary" for each module or submodule in both projects. This means that if the local CPU has access in one project, the option "---" must be set in the other project and vice versa.

   Note: The option "---" for the PROFINET interface and therefore for the ports makes the associated parameters read-only and not changeable. Parameters of the PROFINET interface and port parameters can only be edited in the project in which the PROFINET interface is assigned to the local CPU. The ports can be interconnected in both projects regardless of this.

8. Check whether the same IP address parameters and device name are set for the shared device in all projects.

   Check whether the same S7 subnet ID is set in all projects for the subnet to which the shared device is connected (subnet properties, "General" area in the Inspector window).

Note

If you make changes to the shared device: Make the same changes in each project on the shared device. Make sure that only one IO controller has access to a module or submodule.

Procedure: Adjusting the real-time settings

To ensure that all IO controllers and shared devices are operated with the appropriate send clock and that the update times are calculated correctly based on the communication load, you must adjust and check the following settings:

1. Select the project whose IO controllers have access to the PROFINET interface and the ports of the shared device.
2. Select the interface module of the shared device in the network view.
3. In the Inspector window, navigate to the "PROFINET interface > Advanced options > Real time settings > IO cycle" area.
4. In the "Shared device" area, set the number of project-external IO controllers. The maximum number depends on the IO device (specification in GSD file).
5. You must set the same send clock for each IO controller that has access to modules and submodules of the shared device:

- If you configure the IO controller with STEP 7 (TIA Portal):
  - Open the corresponding project.
  - Select the PROFINET interface of the IO controller.
  - Select the "Advanced options > Real time settings > IO communication" area in the Inspector window and set the shared send clock.
- If you configure the IO controller with a different engineering tool:
  - Select the PROFINET interface of the shared device in STEP 7 (TIA Portal) and read out the send clock on the shared device ("Advanced options > Real time settings" area).
  - Enter the read send clock in the engineering tool.

Note

If you configure all IO controllers that have access to the shared device in STEP 7 (TIA Portal), you can set shorter send clocks on the IO controller than supported by the shared device (send clock adaptation).

Compiling and loading

You must compile the configurations for the different IO controllers and load them to the CPUs one after the other.

Due to the distributed configuration with separate projects, STEP 7 does not output consistency errors in the case of incorrect access parameter assignment. These are examples of incorrect access parameter assignment:

- Several IO controllers have access to the same module
- IP address parameters or send clocks are not identical

These errors do not show up until controller operation and are output as configuration errors.

11.2.14.3 Example: Configuring an I-device as a shared device

This example describes how to configure an S7-1200 as an I-device with STEP 7 Version V13 SP1 or higher and then use it in two projects as a shared device.

A "distributed" configuration with different engineering tools for different IO controller families is possible. The procedure described below is based on STEP 7 V13 SP1 and is limited to a configuration with two IO controllers of the S7-1200 family that share the transfer areas of an I-device as a shared device. The I-device itself is an CPU 1215C.

The example creates three projects with one IO controller each:

- S7-1200-I-Device
- Controller1
- Controller2

You use the S7-1200-I-Device project to configure the I-device. You use the PROFINET GSD variant of S7-1200-I-Device in the Controller1 and Controller2 projects in order to assign the transfer areas in the respective higher-level IO controller.

Shared I-device concept

The shared I-device concept requires a minimum of three separate projects:

- I-device project: You configure and program an I-device to perform a particular automation task. You define transfer areas as the I/O interface for the higher level controllers and assign these transfer areas to different IO controllers. For the connection to higher-level IO controllers, you provide a PROFINET GSD file and use the transfer areas to access the I-device.
- Controllers that share the I-device (two projects): You use the I-device as a PROFINET GSD variant during configuration of the PROFINET IO system and, in this process, specify the I/O addresses under which the IO controllers access the transfer areas.

I-device

You assign the following parameters for an S7-1200 CPU as an I-device:

- Centralized and distributed I/O
- Desired transfer areas
- Number of IO controllers having access to this I-device (always greater than 1 for a shared device)

Note

You configure the I-device without a higher-level IO controller. As a result, you can only use the local I/O addresses of the transfer area (= "Address in the I-device") to create the user program for editing the addresses from the transfer area. You download the I-device, completely configured except for the connection to the higher-level IO controller, to the S7-1200 CPU.

You export a PROFINET GSD file from the I-device configuration.

Controllers that share the I-device

You must install the PROFINET GSD file created from the I-device configuration in all engineering systems that you use in configuring a PROFINET IO system with this shared I-device. If you configure all uses of this I-device with STEP 7 V13 SP1, it is sufficient to install the GSD file in STEP 7.

You configure the I-device as a GSD variant on the PROFINET IO system in the projects involved. In STEP 7 V13 SP1, you find this I-device under "Other field devices > PROFINET IO > PLCs & CPs" following installation.

In each of the projects involved, you assign transfer areas exclusively to the higher-level IO controllers (default setting: all). You set the other transfer areas to "---" (not assigned). When you do so, the local IO controller cannot access this transfer area, and you can assign the transfer area to another IO controller in another project.

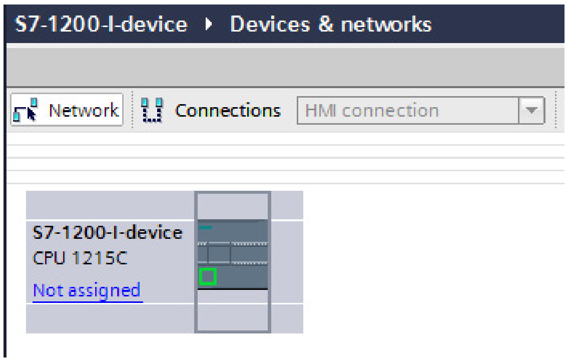
Requirements

- STEP 7 V13 SP1 or higher
- IO device supports shared device functionality (for example, ET 200SP IM 155-6 PN HF V3.1).
- GSD file for configuring the IO device as a shared device is installed.

Procedure: Creating the S7-1200-I-device project

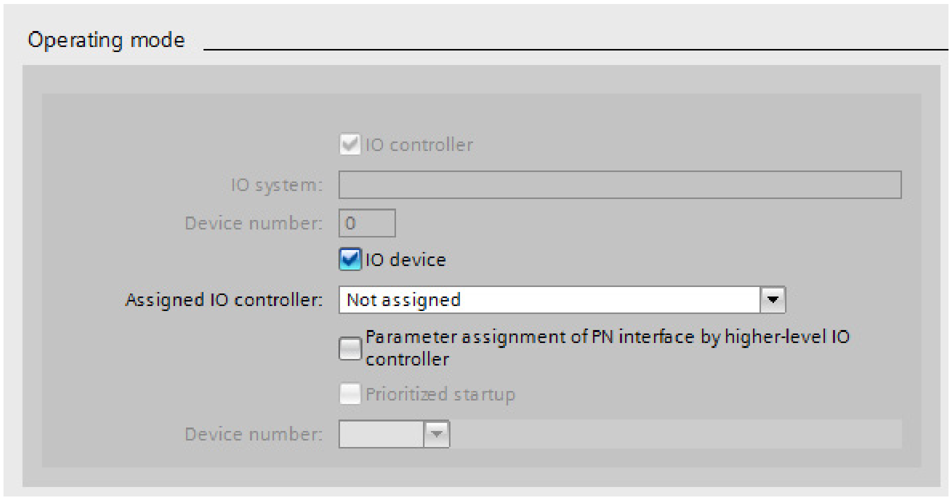
To create the project with a shared I-device, follow these steps:

1. Start STEP 7.
2. Create a new project with the name "S7-1200-I-device".
3. Insert a CPU 1215C from the hardware catalog in the network view. Assign the name "S7-1200-I-device".



4. Double-click the IO device and configure all required modules and submodules.

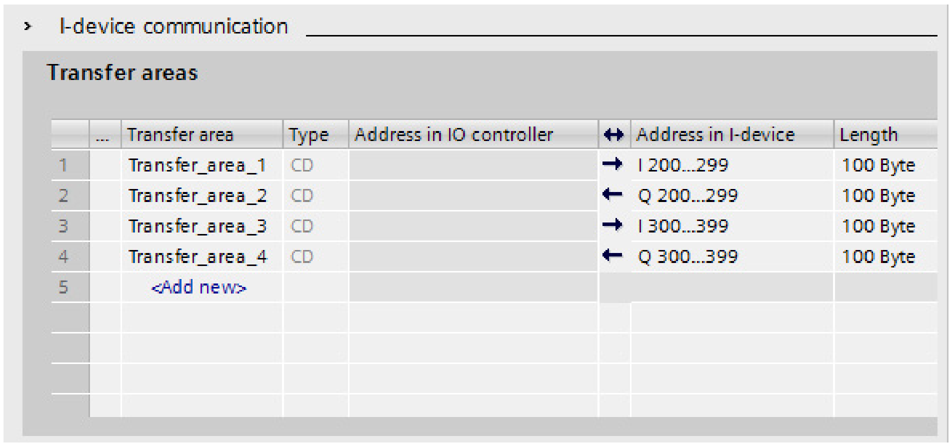
5. Assign the module parameters. In particular, you must configure the following settings for the CPU in the area of the PROFINET interface [X1]:
   – Enable the "IO device" option in the "Operating mode" area.



Operating mode

	✓ IO controller
IO system:	
Device number:	0
	✓ IO device
Assigned IO controller:	Not assigned
	☐ Parameter assignment of PN interface by higher-level IO controller
	☐ Prioritized startup
Device number:	

   – Configure the transfer areas in the "Operating mode" > "I-device configuration" area. The "Address in IO controller" column remains empty because no IO controller is assigned.



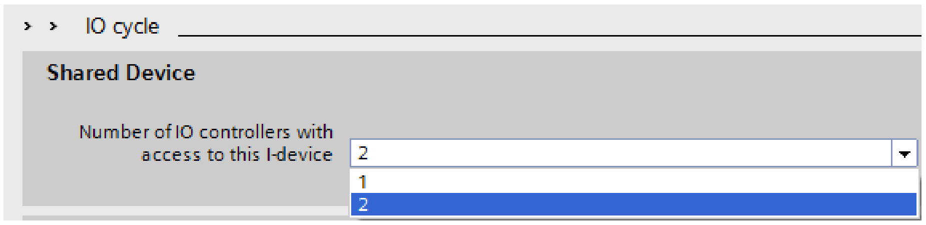
› I-device communication

Transfer areas

	...	Transfer area	Type	Address in IO controller	↔	Address in I-device	Length
1		Transfer_area_1	CD		→	I 200...299	100 Byte
2		Transfer_area_2	CD		←	Q 200...299	100 Byte
3		Transfer_area_3	CD		→	I 300...399	100 Byte
4		Transfer_area_4	CD		←	Q 300...399	100 Byte
5		<Add new>					

   Note: To change an input area to an output area, and vice versa, you must navigate to the area of the corresponding transfer area.

   – Select the number of IO controllers (at least two) that will access the shared I-device during operation ("Operating mode" > "Real time settings" area, "Shared Device" area).

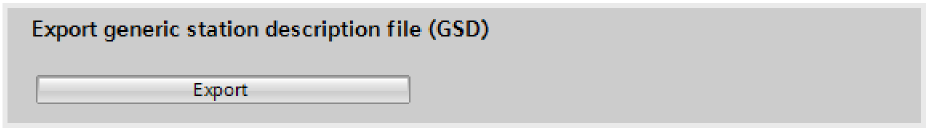


› › IO cycle

Shared Device

Number of IO controllers with access to this I-device: 2

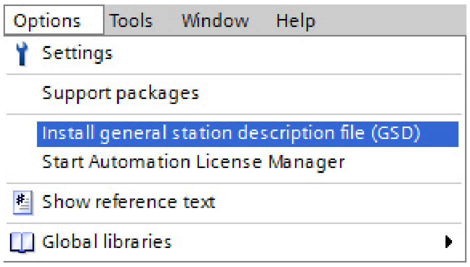
6. Save the project.
7. Click the "Export" button ("Mode" > "I-device configuration" area, "Export general station description file (GSD)" section). If you do not change the name in the Export dialog, the GSD file uses an assigned format name (for example, "GSDML-V2.31-#Siemens-PreConf_S7-1200-I-Device-20130925-123456").



Procedure: Creating the Controller1 project

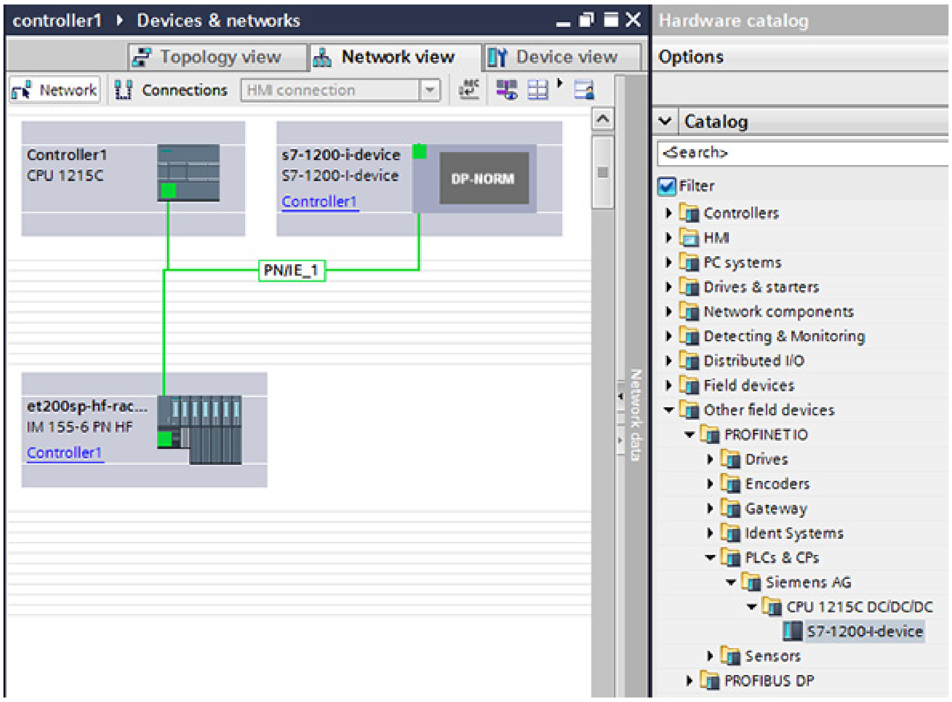
To create the first project with a shared I-device, follow these steps:

1. Start STEP 7.
2. Install the PROFINET GSD file from the export of the I-device CPU (S7-1200-I-Device).

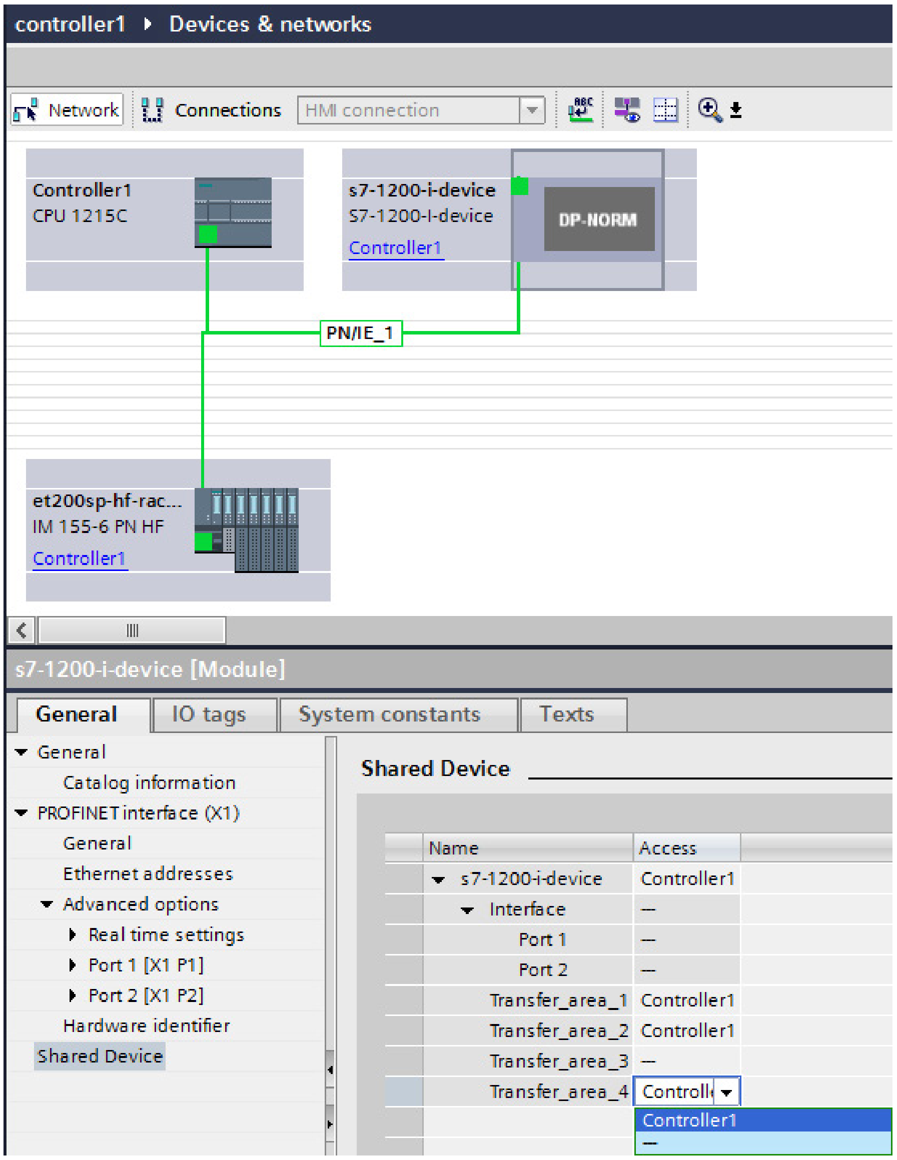


3. Create a new project with the name "Controller1".
4. Insert the CPU 1215C in the network view. The name of the CPU should be "Controller1".
5. Insert the I-device from the hardware catalog (Hardware catalog: Other field devices > PROFINET IO > PLCs & CPs).

6. Assign the IO controller "Controller1" to the I-device.

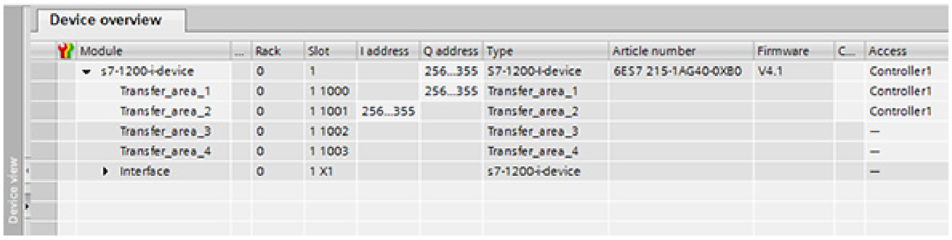


7. Select the "Shared device" area in the properties of the I-device:
   - In the table, all transfer areas and the PROFINET interface are assigned to the local IO controller (Controller1).
   - Define the transfer areas to which the Controller1 CPU should **not** have access. Select the "---" entry for these areas. These transfer areas are provided for Controller2.



Name	Access
s7-1200-i-device	Controller1
Interface	—
Port 1	—
Port 2	—
Transfer_area_1	Controller1
Transfer_area_2	Controller1
Transfer_area_3	—
Transfer_area_4	Controll
	Controller1
	—

8. You can adapt the addresses from the Device view of the IO controller in the Device overview. To open the device overview, double-click the I-device.



Device overview

Module	...	Rack	Slot	I address	Q address	Type	Article number	Firmware	C...	Access
s7-1200-i-device		0	1		256...355	S7-1200-I-device	6ES7 215-1AG40-0XB0	V4.1		Controller1
Transfer_area_1		0	1 1000		256...355	Transfer_area_1				Controller1
Transfer_area_2		0	1 1001	256...355		Transfer_area_2				Controller1
Transfer_area_3		0	1 1002			Transfer_area_3				---
Transfer_area_4		0	1 1003			Transfer_area_4				---
Interface		0	1 X1			s7-1200-i-device				---

9. Save the project.

Procedure - Creating the Controller2 project

To create the second project with a shared device, follow these steps:

1. Start STEP 7 once again.

   A new instance of STEP 7 opens.

2. In the new instance, create a new project with the name "Controller2".
3. Insert the CPU 1215C in the network view. Assign the name "Controller2".
4. Insert the I-device from the hardware catalog (Hardware catalog: Other field devices > PROFINET IO > PLCs & CPs).
5. Assign the IO controller "Controller2" to the I-device.
6. Adapt the access to the transfer areas as in the Controller1 project. Ensure that no duplicate assignments result.
7. Adapt the parameters of the subnet and PROFINET interface. Because the shared I-device involves the same device in different projects, these data must match.
8. Save the project.

Both projects now have an identically configured shared I-device. The IO controller access and the parameters of the PROFINET interface should still be checked in the different projects during the next step.

Summary - Assigning parameters for access to the shared device

The transfer areas are automatically assigned to the local IO controller. To change the assignment, follow these steps:

1. Click the "S7-1200-I-Device" device in the network view of the "Controller1" project, and select the "Shared device" area.
2. A table shows which CPU has access to each of the configured transfer areas. The default setting is that the local CPU has access to all modules and submodules.
3. Keep the setting "Controller1" for all transfer areas that are to remain in the address range of the local CPU.

   Select the setting "---" for all transfer areas that are to be located in the address range of the "Controller2" CPU from the "Controller2" project. This means that an IO controller outside the project is to have access to the transfer area.

4. Follow the same procedure for the remaining projects.
5. Finally, check whether the settings for access are "complementary" for each module or submodule in both projects. This means that if the local CPU has access in one project, the option "---" must be set in the other project and vice versa.

   Note: The option "---" for the PROFINET interface and therefore for the ports makes the associated parameters read-only and not changeable. Parameters of the PROFINET interface and port parameters can only be edited in the project in which the PROFINET interface is assigned to the local CPU. The ports can be interconnected in both projects regardless of this.
6. Check whether the same IP address parameters and device name are set for the shared device in all projects.

   Check whether the same S7 subnet ID is set in all projects for the subnet to which the shared device is connected (subnet properties, "General" area in the Inspector window).

Note

If you make changes to the I-device (for example, change the number or length of the transfer areas), export the I-device as a GSD file again. Re-install the GSD file in each project that uses the I-device as a shared device. Make sure that only one IO controller has access to a transfer area.

Procedure - Adjusting the real-time settings

To ensure that all IO controllers and shared devices are operated with the appropriate send clock and that the update times are calculated correctly based on the communication load, you must adjust and check the following settings:

1. You must set the same send clock for each IO controller that has access to modules and submodules of the shared device:

- If you configure the IO controller with STEP 7 (TIA Portal), perform these steps:
  – Open the corresponding project.
  – Select the PROFINET interface of the IO controller.
  – Select the "Advanced options > Real time settings > IO communication" area in the Inspector window and set the shared send clock.
- If you configure the IO controller with a different engineering tool, perform these steps:
  – Select the PROFINET interface of the shared device in STEP 7 (TIA Portal) and read out the send clock on the shared device ("Advanced options > Real time settings" area)
  – Enter the read send clock in the engineering tool.

Note

If you configure **all** IO controllers that have access to the shared I-device in STEP 7 (TIA Portal), you can set shorter send clocks on the IO controller than supported by the shared device (send clock adaptation).

Compiling and downloading

You must compile the configurations for the different IO controllers and download them to the CPUs one after the other.

Due to the distributed configuration with separate projects, STEP 7 does not output consistency errors in the case of incorrect access parameter assignment. These are examples of incorrect access parameter assignment:

- Several IO controllers have access to the same module.
- IP address parameters or send clocks are not identical.

These errors do not show up until controller operation and are output as configuration errors.

11.2.15 Diagnostics

Refer to "Organization blocks (OBs)" (Page 88) for information on how to use organization blocks (OBs) for diagnostics with these communication networks.

11.2.16 Distributed I/O instructions

Refer to "Distributed I/O (PROFINET, PROFIBUS, or AS-i)" (Page 350) for information on how to use the distributed I/O instructions with these communication networks.

11.2.17 Diagnostic instructions

Refer to the "Diagnostics (PROFINET or PROFIBUS)": "Diagnostics instructions" (Page 384) for information on how to use these instructions with these communication networks.

11.2.18 Diagnostic events for distributed I/O

Refer to the "Diagnostics (PROFINET or PROFIBUS)": "Diagnostics events for distributed I/O" (Page 384) for information on how to use this diagnostic information with these communication networks.

11.3 PROFIBUS

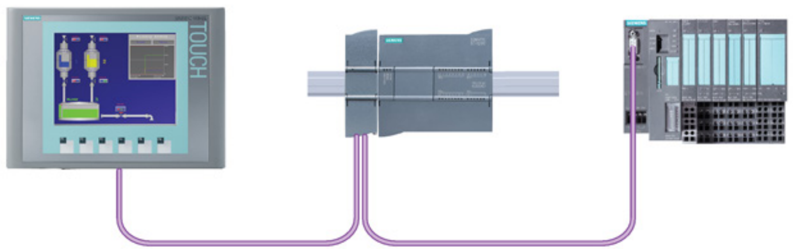
A PROFIBUS system uses a bus master to poll slave devices distributed in a multi-drop fashion on an RS485 serial bus. A PROFIBUS slave is any peripheral device (I/O transducer, valve, motor drive, or other measuring device) which processes information and sends its output to the master. The slave forms a passive station on the network since it does not have bus access rights, and can only acknowledge received messages, or send response messages to the master upon request. All PROFIBUS slaves have the same priority, and all network communication originates from the master.

A PROFIBUS master forms an "active station" on the network. PROFIBUS DP defines two classes of masters. A class 1 master (normally a central programmable controller (PLC) or a PC running special software) handles the normal communication or exchange of data with the slaves assigned to it. A class 2 master (usually a configuration device, such as a laptop or programming console used for commissioning, maintenance, or diagnostics purposes) is a special device primarily used for commissioning slaves and for diagnostic purposes.

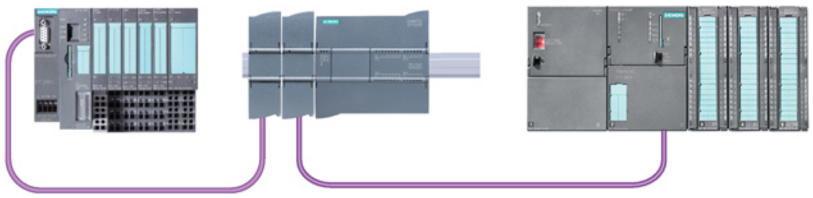
The S7-1200 is connected to a PROFIBUS network as a DP slave with the CM 1242-5 communication module. The CM 1242-5 (DP slave) module can be the communications partner of DP V0/V1 masters. If you want to configure the module in a third-party system, there is a GSD file available for the CM 1242-5 (DP slave) on the CD that ships with the module and on Siemens Automation Customer Support (http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objid=6GK72425DX300XE0&caller=view) pages on the Internet.

In the figure below, the S7-1200 is a DP slave to an S7-300 controller:

The S7-1200 is connected to a PROFIBUS network as a DP master with the CM 1243-5 communication module. The CM 1243-5 (DP master) module can be the communications partner of DP V0/V1 slaves. In the figure below, the S7-1200 is a master controlling an ET200S DP slave:



If a CM 1242-5 and a CM 1243-5 are installed together, an S7-1200 can perform as both a slave of a higher-level DP master system and a master of a lower-level DP slave system, simultaneously:



For V4.0, you can configure a maximum of three PROFIBUS CMs per station, in which there can be any combination of DP master or DP slave CMs. DP masters in a V3.0 or greater CPU firmware implementation can each control a maximum of 32 slaves.

The configuration data of the PROFIBUS CMs is stored on the local CPU. This allows simple replacement of these communications modules when necessary.

To use PROFIBUS with S7-1200 V4.0 CPUs, you must upgrade the PROFIBUS Master CM firmware to V1.3.

You can make this upgrade using a Secure Digital (SD) card.

Note

It is recommended that you always update the PROFIBUS CM firmware to the latest version available (http://support.automation.siemens.com/WW/view/en/42131407) at the Siemens Service and Support web site.

11.3.1 Communications services of the PROFIBUS CMs

The PROFIBUS CMs use the PROFIBUS DP-V1 protocol.

Types of communication with DP-V1

The following types of communication are available with DP-V1:

- Cyclic communication (CM 1242-5 and CM 1243-5)

  Both PROFIBUS modules support cyclic communication for the transfer of process data between DP slave and DP master.

  Cyclic communication is handled by the operating system of the CPU. No software blocks are required for this. The I/O data is read or written directly from/to the process image of the CPU.

- Acyclic communication (CM 1243-5 only)

  The DP master module also supports acyclic communication using software blocks:

  – The "RALRM" instruction is available for interrupt handling.

  – The "RDREC" and "WRREC" instructions are available for transferring configuration and diagnostics data.

Functions not supported by the CM 1243-5: SYNC/FREEZE and Get_Master_Diag

Other communications services of the CM 1243-5

The CM 1243-5 DP master module supports the following additional communications services:

- S7 communication

  – PUT/GET services

    The DP master functions as a client and server for queries from other S7 controllers or PCs via PROFIBUS.

  – PG/OP communication

    The PG functions allow the downloading of configuration data and user programs from a PG and the transfer of diagnostics data to a PG.

    Possible communications partners for OP communication are HMI panels, SIMATIC panel PCs with WinCC flexible or SCADA systems that support S7 communication.

11.3.2 Reference to the PROFIBUS CM user manuals

Further information

You can find detailed information on the PROFIBUS CMs in the manuals for the devices. You can find these on the Internet in the pages of Siemens Industrial Automation Customer Support under the following entry IDs:

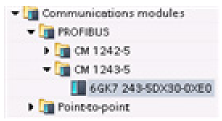
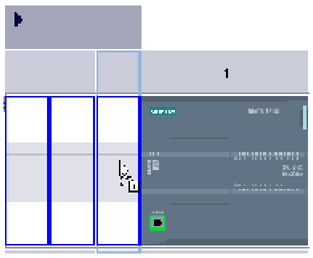
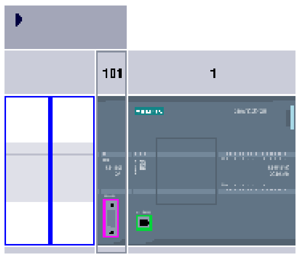
- CM 1242-5 (http://support.automation.siemens.com/WW/view/en/44632650)
- CM 1243-5 (http://support.automation.siemens.com/WW/view/en/44632657)

11.3.3 Configuring a DP master and slave device

11.3.3.1 Adding the CM 1243-5 (DP master) module and a DP slave

In the "Devices and networks" portal, use the hardware catalog to add PROFIBUS modules to the CPU. These modules are connected to the left side of the CPU. To insert a module into the hardware configuration, select the module in the hardware catalog and either double-click or drag the module to the highlighted slot.

Table 11- 55 Adding a PROFIBUS CM 1243-5 (DP master) module to the device configuration

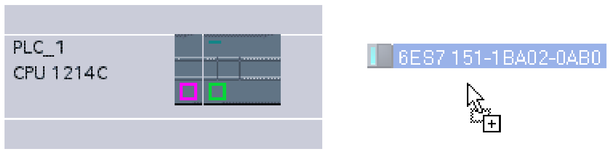
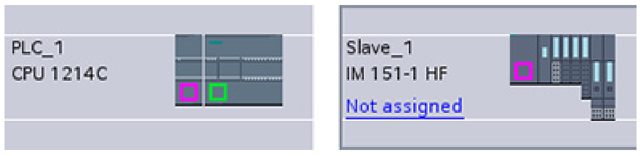
Module	Select the module	Insert the module	Result
CM 1243-5 (DP master)	 Communications modules / PROFIBUS / CM 1242-5 / CM 1243-5 / 6GK7 243-5DX30-0XE0 / Point-to-point		

Use the hardware catalog to add DP slaves as well. For example, to add an ET200 S DP slave, in the Hardware Catalog, expand the following containers:

- Distributed I/O
- ET200 S
- Interface modules
- PROFIBUS

Next, select "6ES7 151-1BA02-0AB0" (IM151-1 HF) from the list of part numbers, and add the ET200 S DP slave as shown in the figure below.

Table 11- 56 Adding an ET200 S DP slave to the device configuration

Insert the DP slave	Result
 PLC_1 CPU 1214C 6ES7 151-1BA02-0AB0	 PLC_1 CPU 1214C Slave_1 IM 151-1 HF Not assigned

11.3.3.2 Configuring logical network connections between two PROFIBUS devices

After you configure the CM 1243-5 (DP master) module, you are now ready to configure your network connections.

In the Devices and Networks portal, use the "Network view" to create the network connections between the devices in your project. To create the PROFIBUS connection, select the purple (PROFIBUS) box on the first device. Drag a line to the PROFIBUS box on the second device. Release the mouse button and your PROFIBUS connection is joined.

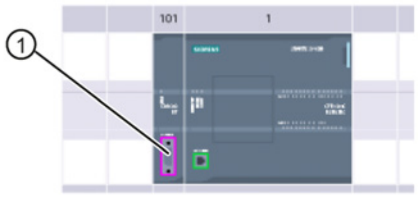
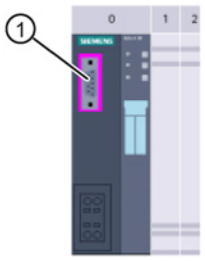
Refer to "Device Configuration: Creating a network connection" (Page 618) for more information.

11.3.3.3 Assigning PROFIBUS addresses to the CM 1243-5 module and DP slave

Configuring the PROFIBUS interface

After you configure logical network connections between two PROFIBUS devices, you can configure parameters for the PROFIBUS interfaces. To do so, click the purple PROFIBUS box on the CM 1243-5 module, and the "Properties" tab in the inspector window displays the PROFIBUS interface. The DP slave PROFIBUS interface is configured in the same manner.

Table 11- 57 Configuring the CM 1243-5 (DP master) module and ET200 S DP slave PROFIBUS interfaces

CM 1243-5 (DP master) module	ET200 S DP slave
	

① PROFIBUS port

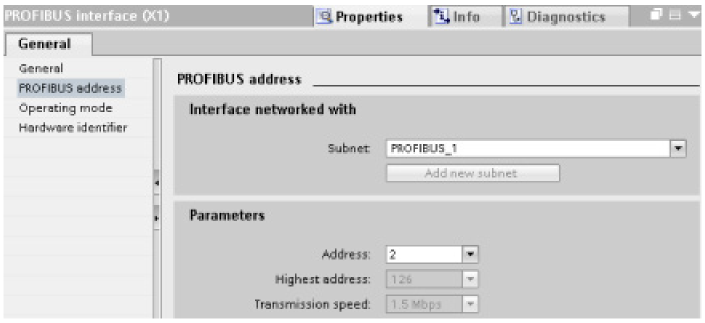
Assigning the PROFIBUS address

In a PROFIBUS network, each device is assigned a PROFIBUS address. This address can range from 0 through 127, with the following exceptions:

- Address 0: Reserved for network configuration and/or programming tools attached to the bus
- Address 1: Reserved by Siemens for the first master
- Address 126: Reserved for devices from the factory that do not have a switch setting and must be re-addressed through the network
- Address 127: Reserved for broadcast messages to all devices on the network and may not be assigned to operational devices

Thus, the addresses that may be used for PROFIBUS operational devices are 2 through 125.

In the Properties window, select the "PROFIBUS address" configuration entry. STEP 7 displays the PROFIBUS address configuration dialog, which is used to assign the PROFIBUS address of the device.



PROFIBUS address

Interface networked with

Subnet: PROFIBUS_1

Parameters

Address: 2
Highest address: 126
Transmission speed: 1.5 Mbps

Table 11- 58 Parameters for the PROFIBUS address

Parameter		Description
Subnet	Name of the Subnet to which the device is connected. Click the "Add new subnet" button to create a new subnet. "Not connected" is the default. Two connection types are possible: • The "Not connected" default provides a local connection. • A subnet is required when your network has two or more devices.	
Parameters	Address	Assigned PROFIBUS address for the device
	Highest address	The highest PROFIBUS address is based on the active stations on the PROFIBUS (for example, DP master). Passive DP slaves independently have PROFIBUS addresses from 1 to 125 even if the highest PROFIBUS address is set to 15, for example. The highest PROFIBUS address is relevant for token forwarding (forwarding of the send rights), and the token is only forwarded to active stations. Specifying the highest PROFIBUS address optimizes the bus.

Parameter		Description
	Transmission rate	Transmission rate of the configured PROFIBUS network: The PROFIBUS transmission rates range from 9.6 Kbits/sec to 12 Mbits/sec. The transmission rate setting depends on the properties of the PROFIBUS nodes being used. The transmission rate should not be greater than the rate supported by the slowest node. The transmission rate is normally set for the master on the PROFIBUS network, with all DP slaves automatically using that same transmission rate (auto-baud).

11.3.4 Distributed I/O instructions

Refer to "Distributed I/O (PROFINET, PROFIBUS, or AS-i)" (Page 350) for information on how to use the distributed I/O instructions with these communication networks.

11.3.5 Diagnostic instructions

Refer to the "Diagnostics (PROFINET or PROFIBUS)": "Diagnostics instructions" (Page 384) for information on how to use these instructions with these communication networks.

11.3.6 Diagnostic events for distributed

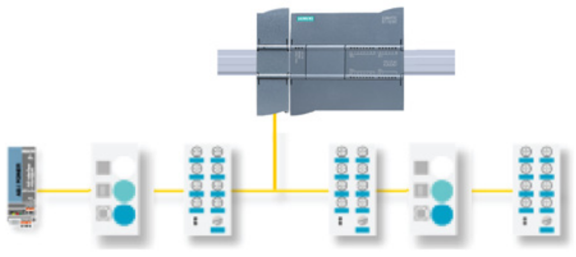
Refer to the "Diagnostics (PROFINET or PROFIBUS)": "Diagnostics events for distributed I/O" (Page 384) for information on how to use this diagnostic information with these communication networks.

11.4 AS-i

The S7-1200 AS-i master CM 1243-2 allows the attachment of an AS-i network to an S7-1200 CPU.

The actuator/sensor interface, or AS-i, is a single master network connection system for the lowest level in automation systems. The CM 1243-2 serves as the AS-i master for the network. Using a single AS-i cable, sensors and actuators (AS-i slave devices) can be connected to the CPU through the CM 1243-2. The CM 1243-2 handles all AS-i network coordination and relays data and status information from the actuators and sensors to the CPU through the I/O addresses assigned to the CM 1243-2. You can access binary or analog values depending on the slave type. The AS-i slaves are the input and output channels of the AS-i system and are only active when called by the CM 1243-2.

In the figure below, the S7-1200 is an AS-i master controlling AS-i I/O module digital/analog slave devices.



To use AS-i with S7-1200 V4.0 CPUs, you must upgrade the AS-i Master CM firmware to V1.1.

You can make this upgrade using the webserver or a SIMATIC Memory card.

Note

For V4.0 S7-1200 CPUs, if using the web server or a SIMATIC memory card to upgrade from V1.0 to V1.1 AS-i firmware, you must update the AS-i firmware in the AS-i Master CM 1243-2 according to the following procedure:

1. Download the firmware upgrade to the AS-i Master CM 1243-2.
2. When the download is complete, power cycle the S7-1200 CPU to complete the firmware upgrade process in the AS-i Master CM 1243-2.
3. Repeat steps 1 and 2 for each additional AS-i Master CM 1243-2. The S7-1200 PLC allows a maximum of three AS-i Master CM 1243-2.

Note

It is recommended that you always update the AS-i CM firmware to the latest version available (http://support.automation.siemens.com/WW/view/en/43416171)at the Siemens Service and Support web site.

11.4.1 Configuring an AS-i master and slave device

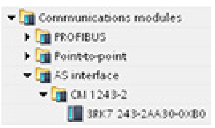
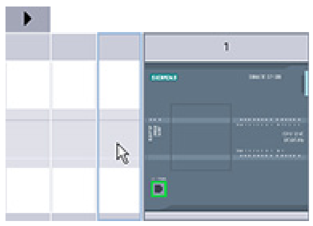
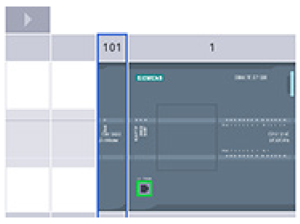
The AS-i master CM 1243-2 is integrated into the S7-1200 automation system as a communication module.

You can find detailed information on the AS-i master CM 1243-2 in the "AS-i master CM 1243-2 and AS-i data decoupling unit DCM 1271 for SIMATIC S7-1200" Manual (http://support.automation.siemens.com/WW/view/en/50414115/133300).

11.4.1.1 Adding the AS-i master CM 1243-2 and AS-i slave

Use the hardware catalog to add AS-i master CM1243-2 modules to the CPU. These modules are connected to the left side of the CPU, and a maximum of three AS-i master CM1243-2 modules can be used. To insert a module into the hardware configuration, select the module in the hardware catalog and either double-click or drag the module to the highlighted slot.

Table 11- 59 Adding an AS-i master CM1243-2 module to the device configuration

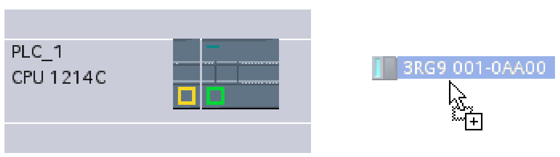
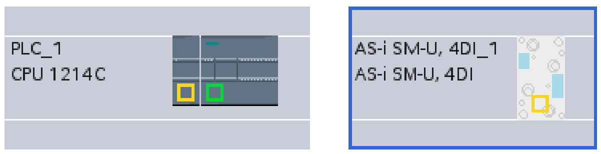
Module	Select the module	Insert the module	Result
CM 1243-2 AS-i Master	 Communications modules PROFIBUS Point-to-point AS interface CM 1243-2 3RK7 243-2AA30-0XB0		

Use the hardware catalog to add AS-i slaves as well. For example, to add an "I/O module, compact, digital, input" slave, in the Hardware Catalog, expand the following containers:

- Field devices
- AS-Interface slaves

Next, select "3RG9 001-0AA00" (AS-i SM-U, 4DI) from the list of part numbers, and add the "I/O module, compact, digital, input" slave as shown in the figure below.

Table 11- 60 Adding an AS-i slave to the device configuration

Insert the AS-i slave	Result
 PLC_1 CPU 1214C 3RG9 001-0AA00	 PLC_1 CPU 1214C AS-i SM-U, 4DI_1 AS-i SM-U, 4DI

11.4.1.2 Configuring logical network connections between two AS-i devices

After you configure the AS-i master CM1243-2, you are now ready to configure your network connections.

In the Devices and Networks portal, use the "Network view" to create the network connections between the devices in your project. To create the AS-i connection, select the yellow (AS-i) box on the first device. Drag a line to the AS-i box on the second device. Release the mouse button and your AS-i connection is joined.

Refer to "Device Configuration: Creating a network connection" (Page 618) for more information.

11.4.1.3 Configuring the properties of the AS-i master CM1243-2

To configure parameters for the AS-i interface, click the yellow AS-i box on the AS-i master CM1243-2 module, and the "Properties" tab in the inspector window displays the AS-i interface.

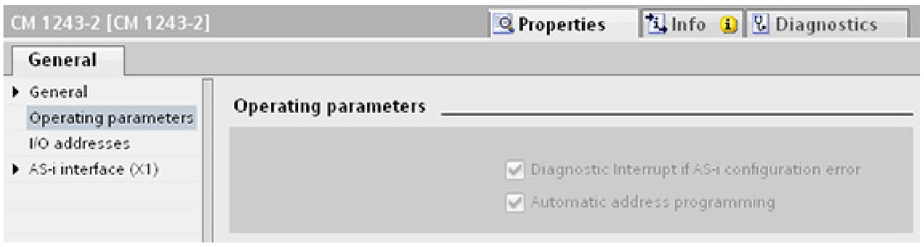
In the STEP 7 inspector window, you can view, configure, and change general information, addresses and operating parameters:

Table 11- 61 AS-i master CM1243-2 module properties

Property	Description
General	Name of the AS-i master CM 1243-2
Operating parameters	Parameters for the response of the AS-i master
I/O addresses	Address area for the slave I/O addresses
AS-i interface (X1)	Assigned AS-i network

Note

"Diagnostic interrupt for faults in the AS-i configuration" and "Automatic address programming" are always active and are therefore shown in gray.

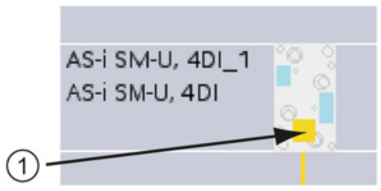


☑ Diagnostic Interrupt if AS-i configuration error
☑ Automatic address programming

11.4.1.4 Assigning an AS-i address to an AS-i slave

Configuring the AS-i slave interface

To configure parameters for the AS-i interface, click the yellow AS-i box on the AS-i slave, and the "Properties" tab in the inspector window displays the AS-i interface.



① AS-i port

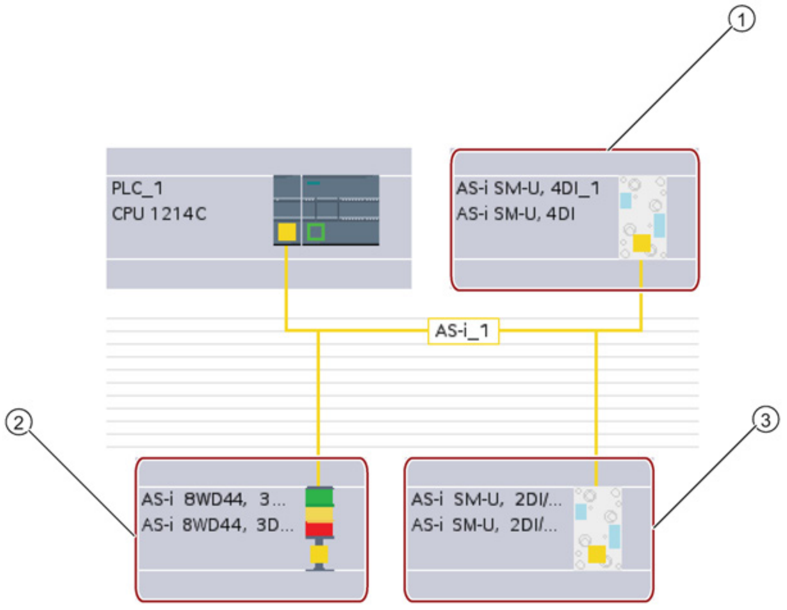
Assigning the AS-i slave address

In an AS-i network, each device is assigned an AS-i slave address. This address can range from 0 through 31; however, address 0 is reserved only for new slave devices. The slave addresses are 1(A or B) to 31(A or B) for a total of up to 62 slave devices.

"Standard" AS-i devices use the entire address, having a number address without the A or B designation. "A/B node" AS-i devices use the A or B portion of each address, enabling each of the 31 addresses to be used twice. The address space range is 1A to 31A plus 1B to 31B.

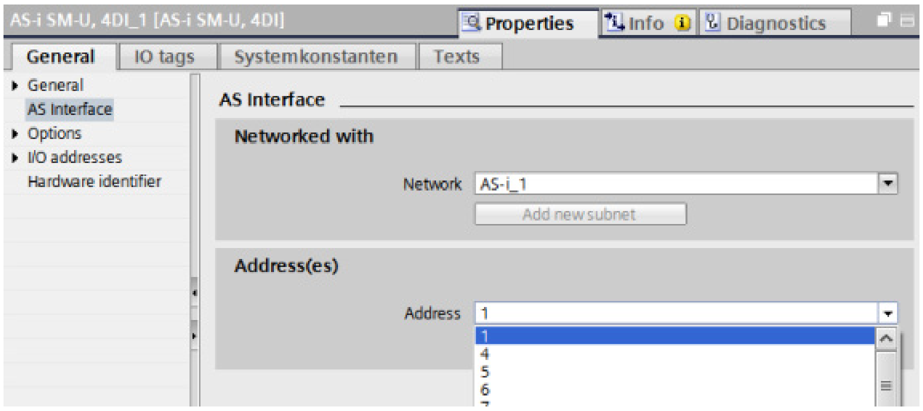
Any address in the range of 1 - 31 can be assigned to an AS-i slave device; in other words, it does not matter whether the slaves begin with address 21 or whether the first slave is actually given the address 1.

In the example below, three AS-i devices have been addressed as "1" (a standard type device), "2A" (an A/B node type device), and "3" (a standard type device):



① AS-i slave address 1; Device: AS-i SM-U, 4DI; article number: 3RG9 001-0AA00
② AS-i slave address 2A; Device: AS-i 8WD44, 3DO, A/B; article number: 8WD4 428-0BD
③ AS-i slave address 3; Device: AS-i SM-U, 2DI/2DO; article number: 3RG9 001-0AC00

Enter the AS-i slave address here:



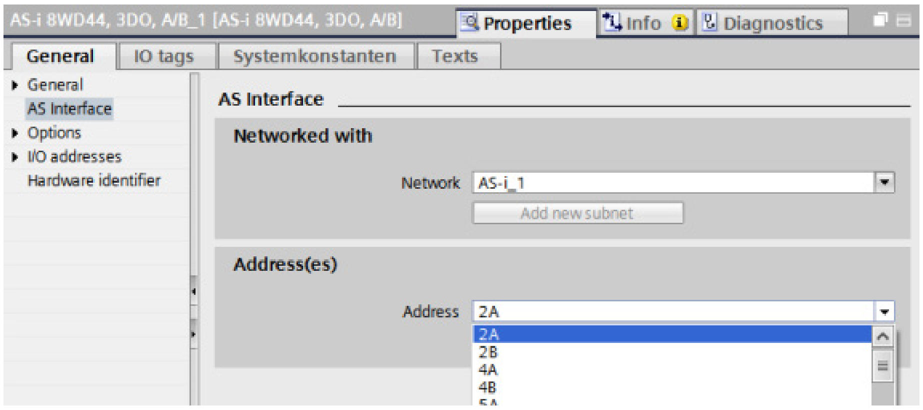
AS Interface

Networked with

Network AS-i_1

Address(es)

Address 1



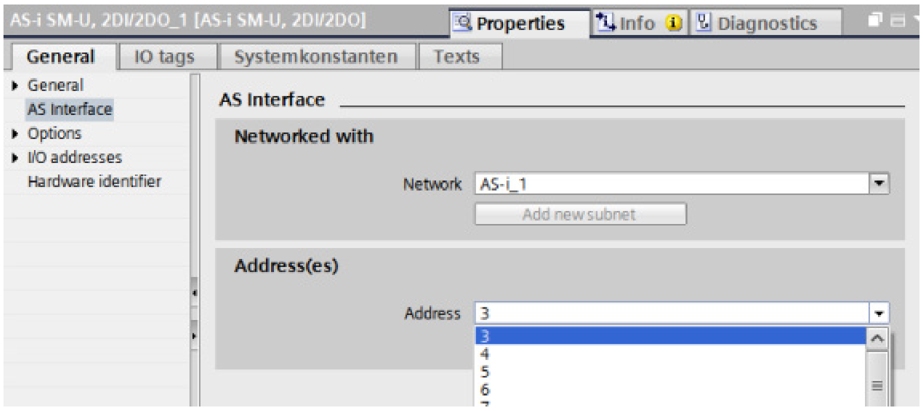
AS Interface

Networked with

Network AS-i_1

Address(es)

Address 2A



AS Interface

Networked with

Network AS-i_1

Address(es)

Address 3

Table 11- 62 Parameters for the AS-i interface

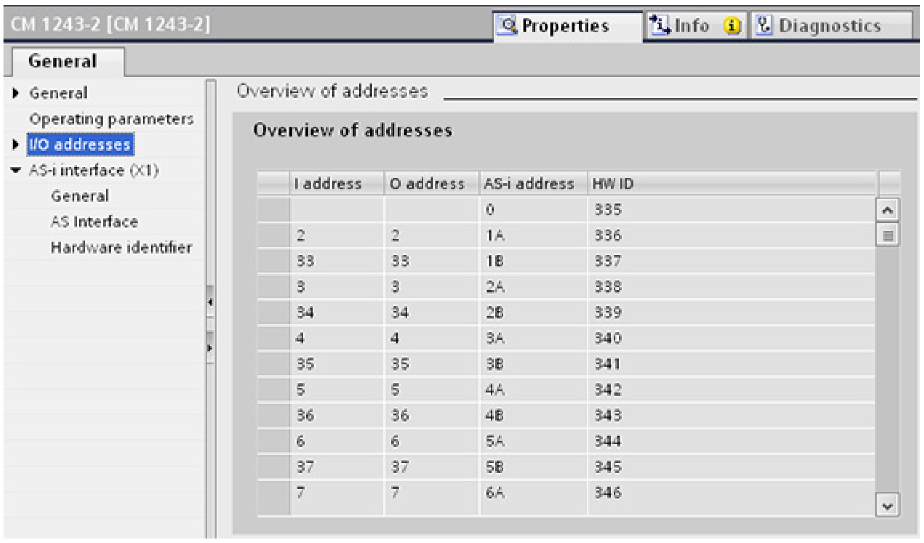
Parameter	Description
Network	Name of the network to which the device is connected
Address(es)	Assigned AS-i address for the slave device in range of 1(A or B) to 31(A or B) for a total of up to 62 slave devices

11.4.2 Exchanging data between the user program and AS-i slaves

11.4.2.1 STEP 7 basic configuration

The AS-i master reserves a 62-byte data area in the I/O area of the CPU. Access to the digital data is performed here in bytes; for each slave, there is one byte of input and one byte of output data.

The assignment of the AS-i connections of the AS-i digital slaves to the data bits of the assigned byte is indicated in the inspection window of the AS-i master CM 1243-2.



Overview of addresses

I address	O address	AS-i address	HW ID
		0	335
2	2	1A	336
33	33	1B	337
3	3	2A	338
34	34	2B	339
4	4	3A	340
35	35	3B	341
5	5	4A	342
36	36	4B	343
6	6	5A	344
37	37	5B	345
7	7	6A	346

You can access the data of the AS-i slaves in the user program by using the displayed I/O addresses with the appropriate bit logic operations (for example, "AND") or bit assignments.

Note

"System assignment" is automatically activated if you do not configure the AS-i slaves with STEP 7.

If you do not configure any slaves, you must inform the AS-i master CM1243-2 about the actual bus configuration using the online function "ACTUAL > EXPECTED".

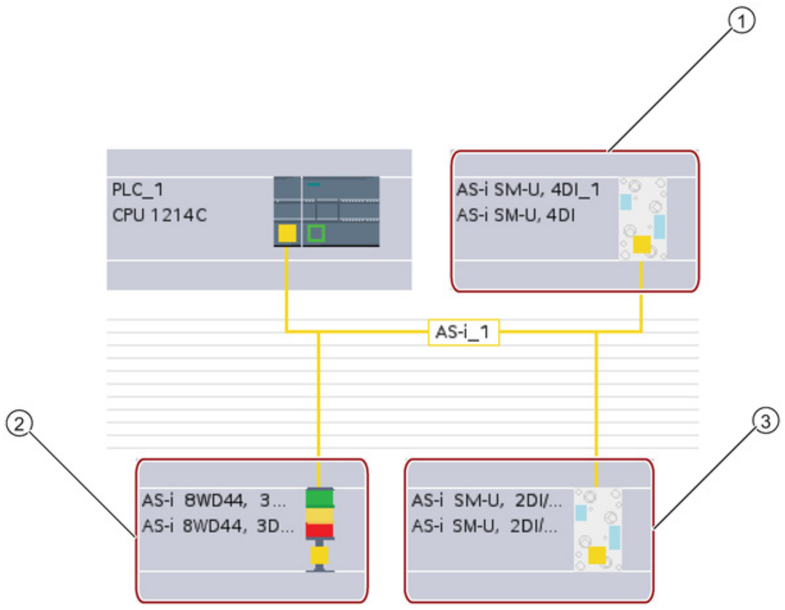
Further information

You can find detailed information on the AS-i master CM 1243-2 in the "AS-i master CM 1243-2 and AS-i data decoupling unit DCM 1271 for SIMATIC S7-1200" Manual (http://support.automation.siemens.com/WW/view/en/50414115/133300).

11.4.2.2 Configuring slaves with STEP 7

Transferring AS-i digital values

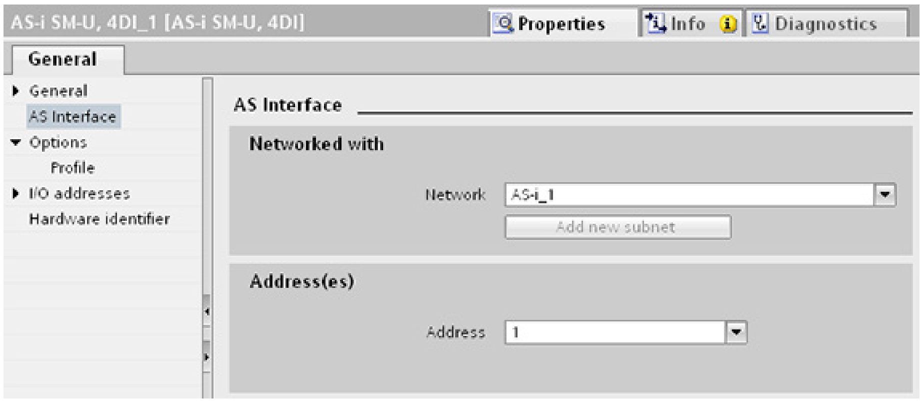
The CPU accesses the digital inputs and outputs of the AS-i slaves through the AS-i master CM1243-2 in cyclic operation. The data is accessed through I/O addresses or by means of a data record transfer.



① AS-i slave address 1
② AS-i slave address 2A
③ AS-i slave address 3

Access to the digital data is performed here in bytes (in other words, one byte is assigned to each AS-i digital slave). When you configure the AS-i slaves in STEP 7, the I/O address for accessing the data from the user program is displayed in the inspection window for the respective AS-i slave.

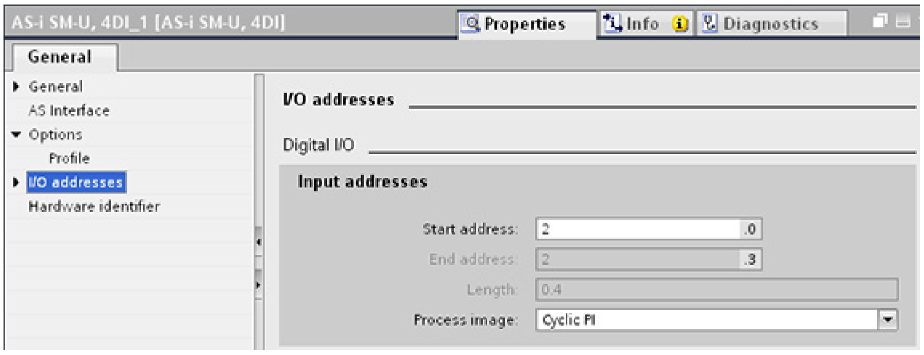
The digital input module (AS-i SM-U, 4DI) in the AS-i network above has been assigned slave address 1. By clicking on the digital input module, the "AS interface" tab in the device "Properties" displays the slave address, as shown below:



Network: AS-i_1

Address: 1

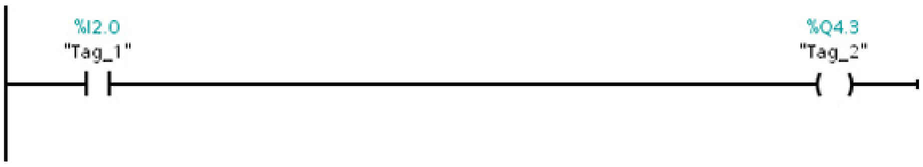
The digital input module (AS-i SM-U, 4DI) in the AS-i network above has been assigned I/O address 2. By clicking on the digital input module, the "I/O addresses" tab in the device "Properties" displays the I/O address, as shown below:



Start address: 2 .0
End address: 2 .3
Length: 0.4
Process image: Cyclic PI

You can access the data of the AS-i slaves in the user program by using their I/O addresses with the appropriate bit logic operations (for example, "AND") or bit assignments. The following simple program illustrates how the assignment works:

Input 2.0 is polled in this program. In the AS-i system, this input belongs to slave1 (Input byte 2, bit 0). Output 4.3, which is then set, corresponds to AS-i slave 3 (Output byte 4, bit 3)



Transferring AS-i analog values

You can access analog data of an AS-i slave through the process image of the CPU if you have configured this AS-i slave in STEP 7 as an analog slave.

If you did not configure the analog slave in STEP 7, you can only access the data of the AS-i slave through the acyclic functions (data record interface). In the user program of the CPU, AS-i calls are read and written using the RDREC (read data record) and WRREC (write data record) distributed I/O instructions.

Note

A configuration of the AS-i slaves specified through STEP 7 and downloaded into the S7 station is transferred by the CPU on the AS-i master CM1243-2 during S7 station start-up. Any existing configuration that was determined through the "System assignment" online function (Page 765) ("ACTUAL -> EXPECTED") will be overwritten.

Further information

You can find detailed information on the AS-i master CM 1243-2 in the "AS-i master CM 1243-2 and AS-i data decoupling unit DCM 1271 for SIMATIC S7-1200" Manual (http://support.automation.siemens.com/WW/view/en/50414115/133300).

11.4.3 Distributed I/O instructions

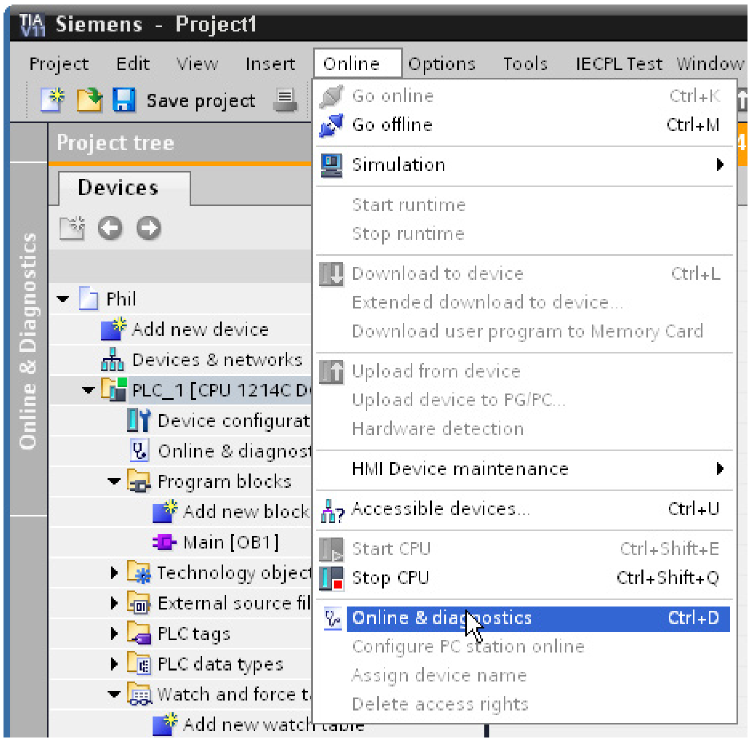
Refer to "Distributed I/O (PROFINET, PROFIBUS, or AS-i)" (Page 350) for information on how to use the distributed I/O instructions with these communication networks.

11.4.4 Working with AS-i online tools

Changing AS-i operational modes online

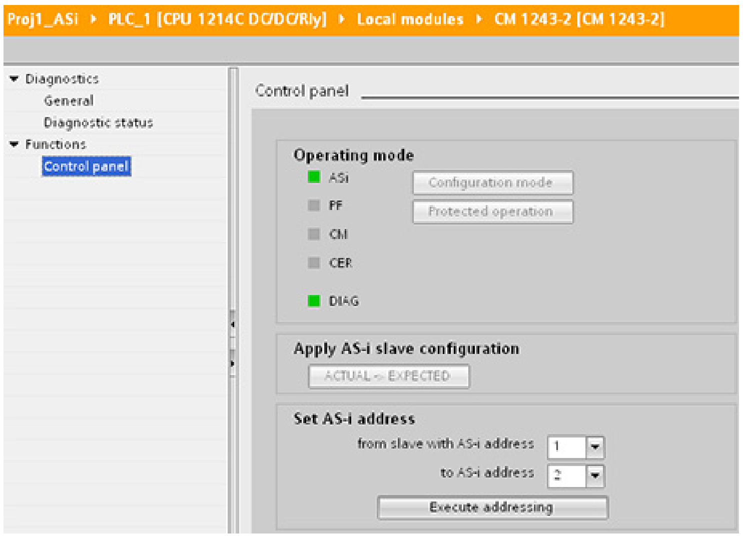
You must go online to view and change the AS-i operational modes.

In order to go online, your must first be in "Device configuration" with the AS-i master CM1243-2 module selected, and then click the "Go online" button in the toolbar. Next, select the "Online and diagnostics" command from the "Online" menu.



There are two AS-i operational modes:

- Protection mode:
  - You cannot change AS-i slave device and CPU I/O addresses.
  - The green "CM" LED is OFF.
- Configuration mode:
  - You can make required changes in your AS-i slave device and CPU I/O addresses.
  - The green "CM" LED is ON.



In the "Set AS-i address" field, you can change the AS-i slave address. A new slave that has not been assigned an address always has address 0. It is detected by the master as a new slave without an address assignment and is not included in normal communication until assigned an address.

Configuration error

When the yellow "CER" LED is ON, there is an error in the AS-i slave device configuration. Select the "ACTUAL > EXPECTED" button to overwrite the AS-i master CM1243-2 module slave device configuration with the AS-i field network slave device configuration.

11.5 S7 communication

11.5.1 GET and PUT (Read and write from a remote CPU) instructions

You can use the GET and PUT instructions to communicate with S7 CPUs through PROFINET and PROFIBUS connections. This is only possible if the "Permit access with PUT/GET communication" function is activated for the partner CPU in the "Protection" property of the local CPU properties:

- Accessing data in a remote CPU: An S7-1200 CPU can only use absolute addresses in the ADDR_x input field to address variables of remote CPUs (S7-200/300/400/1200).
- Accessing data in a standard DB: An S7-1200 CPU can only use absolute addresses in the ADDR_x input field to address DB variables in a standard DB of a remote S7 CPU.
- Accessing data in an optimized DB: An S7-1200 CPU cannot access DB variables in an optimized DB of a remote S7-1200 CPU.
- Accessing data in a local CPU: An S7-1200 CPU can use either absolute or symbolic addresses as inputs to the RD_x or SD_x input fields of the GET or PUT instruction, respectively.

Note

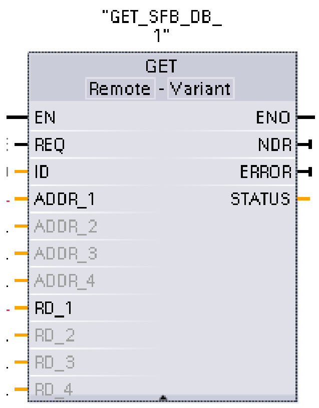
V4.0 CPU program GET/PUT operation is not automatically enabled

A V3.0 CPU program GET/PUT operation is automatically enabled in a V4.0 CPU.

However, a V4.0 CPU program GET/PUT operation in a V4.0 CPU is not automatically enabled. You must go the CPU "Device configuration", inspector window "Properties"tab, "Protection" property to enable GET/PUT access (Page 197).

Table 11- 63 GET and PUT instructions

LAD / FBD	SCL	Description

 "GET_SFB_DB_1"
GET
Remote - Variant
EN
REQ
ID
ADDR_1
ADDR_2
ADDR_3
ADDR_4
RD_1
RD_2
RD_3
RD_4
ENO
NDR
ERROR
STATUS
	`"GET_DB"(`
	`    req:=_bool_in_,`
	`    ID:=_word_in_,`
	`    ndr=>_bool_out_,`
	`    error=>_bool_out_,`
	`    status=>_word_out_,`
	`    addr_1:=_remote_inout_,`
	`   [...addr_4:=_remote_inout_,]`
	`    rd_1:=_variant_inout_`
	`   [,...rd_4:=_variant_inout_]);`
		Use the GET instruction to read data from a remote S7 CPU. The remote CPU can be in either RUN or STOP mode.
		STEP 7 automatically creates the DB when you insert the instruction.

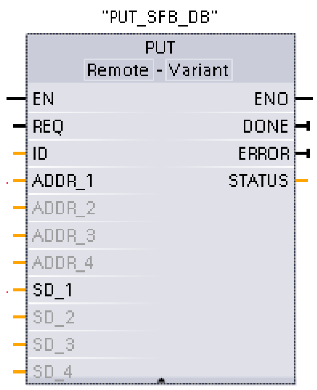
 "PUT_SFB_DB"
PUT
Remote - Variant
EN
REQ
ID
ADDR_1
ADDR_2
ADDR_3
ADDR_4
SD_1
SD_2
SD_3
SD_4
ENO
DONE
ERROR
STATUS
	`"PUT_DB"(`
	`    req:=_bool_in_,`
	`    ID:=_word_in_,`
	`    done=>_bool_out_,`
	`    error=>_bool_out_,`
	`    status=>_word_out_,`
	`    addr_1:=_remote_inout_,`
	`   [...addr_4:=_remote_inout_,]`
	`    sd_1:=_variant_inout_,`
	`   [....sd_4:=_variant_inout_]);`
		Use the PUT instruction to write data to a remote S7 CPU. The remote CPU can be in either RUN or STOP mode.
		STEP 7 automatically creates the DB when you insert the instruction.

Table 11- 64 Data types for the parameters

Parameter and type		Data type	Description
REQ	Input	Bool	A low to high (positive edge) signal starts the operation.
ID	Input	CONN_PRG (Word)	S7 connection ID (Hex)
NDR (GET)	Output	Bool	New Data Ready: • 0: request has not yet started or is still running • 1: task was completed successfully
DONE (PUT)	Output	Bool	DONE: • 0: request has not yet started or is still running • 1: task was completed successfully

Parameter and type		Data type	Description
ERROR STATUS	Output Output	Bool Word	• ERROR=0 STATUS value: – 0000H: neither warning nor error – <> 0000H: Warning, STATUS supplies detailed information • ERROR=1 There is an error. STATUS supplies detailed information about the nature of the error.
ADDR_1	InOut	Remote	Pointer to the memory areas in the remote CPU that stores the data to be read (GET) or that is sent (PUT).
ADDR_2	InOut	Remote	
ADDR_3	InOut	Remote	
ADDR_4	InOut	Remote	
RD_1 (GET) SD_1 (PUT)	InOut	Variant	Pointer to the memory areas in the local CPU that stores the data to be read (GET) or sent (PUT). Data types allowed: Bool (only a single bit allowed), Byte, Char, Word, Int, DWord, DInt, or Real. Note: If the pointer accesses a DB, you must specify the absolute address, such as: P# DB10.DBX5.0 Byte 10 In this case, 10 represents the number of bytes to GET or PUT.
RD_2 (GET) SD_2 (PUT)	InOut	Variant	
RD_3 (GET) SD_3 (PUT)	InOut	Variant	
RD_4 (GET) SD_4 (PUT)	InOut	Variant	

You must ensure that the length (number of bytes) and data types for the ADDR_x (remote CPU) and RD_x or SD_x (local CPU) parameters match. The number after the identifier "Byte" is the number of bytes referenced by the ADDR_x, RD_x, or SD_x parameter.

Note

The total number of bytes received on a GET instruction or the total number of bytes sent on a PUT instruction is limited. The limitations are based on how many of the four possible address and memory areas you use:

- If you use only ADDR_1 and RD_1/SD_1, a GET instruction can get 222 bytes and a PUT instruction can send 212 bytes.
- If you use ADDR_1, RD_1/SD_1, ADDR_2, and RD_2/SD_2, a GET instruction can get a total of 218 bytes and a PUT instruction can send a total of 196 bytes.
- If you use ADDR_1, RD_1/SD_1, ADDR_2, RD_2/SD_2, ADDR_3, and RD_3/SD_3 a GET instruction can get a total of 214 bytes and a PUT instruction can send a total of 180 bytes.
- If you use ADDR_1, RD_1/SD_1, ADDR_2, RD_2/SD_2, ADDR_3, RD_3/SD_3, ADDR_4, RD_4/SD_4 a GET instruction can get a total of 210 bytes and a PUT instruction can send a total of 164 bytes.

The sum of the number of bytes of each of your address and memory area parameters must be less than or equal to the defined limits. If you exceed these limits, the GET or PUT instruction returns an error.

On the rising edge of the REQ parameter, the read operation (GET) or write operation (PUT) loads the ID, ADDR_1, and RD_1 (GET) or SD_1 (PUT) parameters.

- For GET: The remote CPU returns the requested data to the receive areas (RD_x), starting with the next scan. When the read operation has completed without error, the NDR parameter is set to 1. A new operation can only be started only after the previous operation has completed.
- For PUT: The local CPU starts sending the data (SD_x) to the memory location (ADDR_x) in the remote CPU. When the write operation has completed without error, the remote CPU returns an execution acknowledgement. The DONE parameter of the PUT instruction is then set to 1. A new write operation can only be started after the previous operation has completed.

Note

To ensure data consistency, always evaluate when the operation has been completed (NDR = 1 for GET, or DONE = 1 for PUT) before accessing the data or initiating another read or write operation.

The ERROR and STATUS parameters provide information about the status of the read (GET) or write (PUT) operation.

Table 11- 65 Error information

ERROR	STATUS (decimal)	Description
0	11	• New job cannot take effect since previous job is not yet completed. • The job is now being processed in a priority class having lower priority.
0	25	Communication has started. The job is being processed.
1	1	Communications problems, such as: • Connection description not loaded (local or remote) • Connection interrupted (for example: cable, CPU is turned off, or CM/CB/CP is in STOP mode) • Connection to partner not yet established
1	2	Negative acknowledgement from the partner device. The task cannot be executed.
1	4	Errors in the send area pointers (RD_x for GET, or SD_x for PUT) involving the data length or the data type.
1	8	Access error on the partner CPU
1	10	Access to the local user memory not possible (for example, attempting to access a deleted DB)
1	12	When the SFB was called: • An instance DB was specified that does not belong to GET or PUT • No instance DB was specified, but rather a shared DB • No instance DB found (loading a new instance DB)

ERROR	STATUS (decimal)	Description
1	20	• Exceeded the maximum number of parallel jobs/instances • The instances were overloaded at CPU-RUN This status is possible for first execution of the GET or PUT instruction
1	27	There is no corresponding GET or PUT instruction in the CPU.

11.5.2 Creating an S7 connection

Connection mechanisms

To access remote connection partners with PUT/GET instructions, the user must also have permission.

By default, the "Permit access with PUT/GET communication" option is not enabled. In this case, read and write access to CPU data is only possible for communication connections that require configuration or programming both for the local CPU and for the communication partner. Access through BSEND/BRCV instructions is possible, for example.

Connections for which the local CPU is only a server (meaning that no configuration/programming of the communication with the communication partner exists at the local CPU), are therefore not possible during operation of the CPU, for example:

- PUT/GET, FETCH/WRITE or FTP access through communication modules
- PUT/GET access from other S7 CPUs
- HMI access through PUT/GET communication

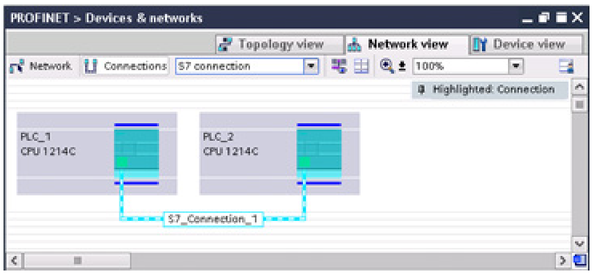
If you want to allow access to CPU data from the client side, that is, you do not want to restrict the communication services of the CPU, you can configure the access protection for the S7-1200 CPU (Page 197) for this level of security.

Connection types

The connection type that you select creates a communication connection to a partner station. The connection is set up, established, and automatically monitored.

In the Devices and Networks portal, use the "Network view" to create the network connections between the devices in your project. First, click the "Connections" tab, and then select the connection type with the dropdown, just to the right (for example, an S7 connection). Click the green (PROFINET) box on the first device, and drag a line to the PROFINET box on the second device. Release the mouse button and your PROFINET connection is joined.

Refer to "Creating a network connection" (Page 618) for more information.



Click the "Highlighted: Connection" button to access the "Properties" configuration dialog of the communication instruction.

11.5.3 Configuring the Local/Partner connection path between two devices

Configuring General parameters

You specify the communication parameters in the "Properties" configuration dialog of the communication instruction. This dialog appears near the bottom of the page whenever you have selected any part of the instruction.

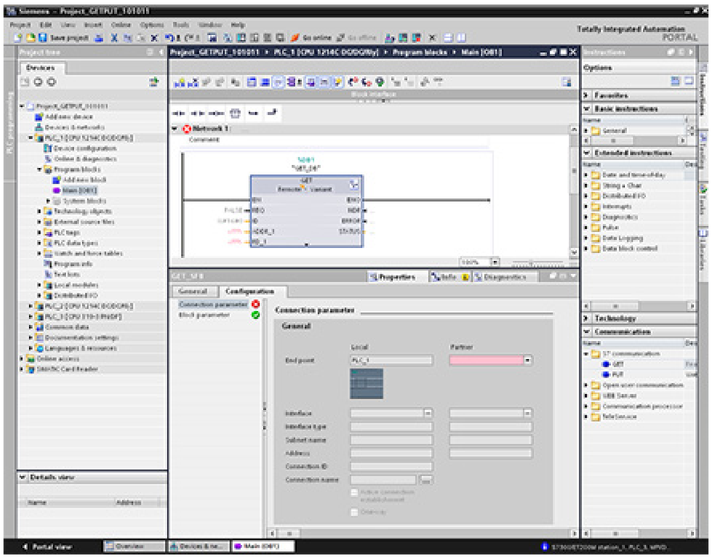
Refer to "Device configuration: Configuring the Local/Partner connection path (Page 619)" for more information.

In the "Address Details" section of the Connection parameters dialog, you define the TSAPs or ports to be used. The TSAP or port of a connection in the CPU is entered in the "Local TSAP" field. The TSAP or port assigned for the connection in your partner CPU is entered under the "Partner TSAP" field.

11.5.4 GET/PUT connection parameter assignment

The GET/PUT instructions connection parameter assignment is a user aid for configuring CPU-to-CPU S7 communication connections.

After inserting a GET or PUT block, STEP 7 displays the connection parameter assignment dialog for the GET/PUT instructions:



The inspector window displays the properties of the connection whenever you have selected any part of the instruction. You can configure the communication parameters in the "Configuration" tab of the "Properties" for the communication instruction.

Note

V4.1 and later CPU program GET/PUT operation is not automatically enabled

A V3.0 CPU program GET/PUT operation is automatically enabled in a V4.1 and later CPU.

However, a V4.1 and later CPU program GET/PUT operation in a V4.1 and later CPU is not automatically enabled. You must go the CPU "Device configuration", inspector window "Properties"tab, "Protection" property to enable GET/PUT access (Page 197).

11.5.4.1 Connection parameters

The "Connection parameters" page allows you to configure the necessary S7 connection and to configure the parameter "Connection ID" that is referenced by the GET/PUT block parameter "ID". The page's content has information about the local endpoint and allows you to define the local interface. You can also define the partner end point.

The "Block parameters" page allows you to configure the additional block parameters.

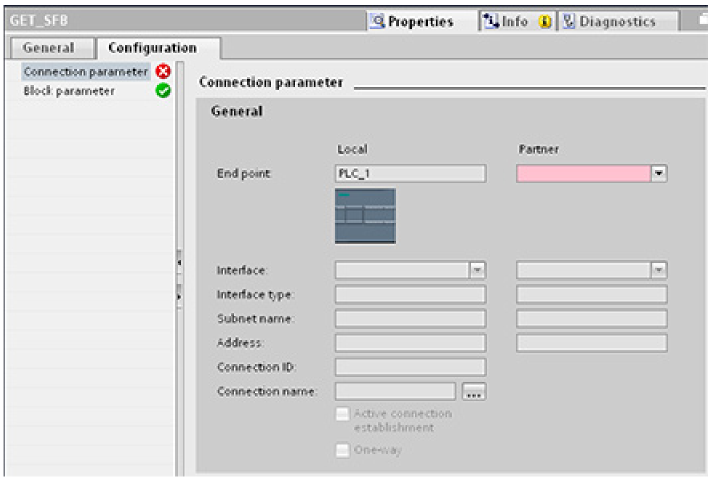


Table 11- 66 Connection parameter: General definitions

Parameter		Definition
Connection parameter: General	End point	"Local End point": Name assigned to the Local CPU "Partner End point": Name assigned to the Partner (remote) CPU Note: In the "Partner End point" dropdown list, the system displays all potential S7 connection partners of the current project as well as the option "unspecified". An unspecified partner represents a communication partner which is not currently in the STEP 7 project (for example, a third party device communication partner).
	Interface	Name assigned to the interfaces Note: You can change the connection by changing the Local and Partner interfaces
	Interface type	Type of interface
	Subnet name	Name assigned to the subnets
	Address	Assigned IP addresses Note: You can specify the remote address of a third party device for an "unspecified" communication partner.
	Connection ID	ID number: Automatically generated by the GET/PUT connection parameter assignment
	Connection name	Local and Partner CPU data storage location: Automatically generated by the GET/PUT connection parameter assignment
	Active connection establishment	Checkbox to select Local CPU as the active connection

Parameter		Definition
	One-way	Checkbox to specify a one-way or two-way connection; read-only Note: In a PROFINET GET/PUT connection, both the local and partner devices can act as a server or a client. This allows a two-way connection, and the "One-way" checkbox is unchecked. In a PROFIBUS GET/PUT connection, in some cases, the Partner device can only act as a server (for example, an S7-300), and the "One-way" checkbox is checked.

Connection ID parameter

There are three ways to change the system-defined connection IDs:

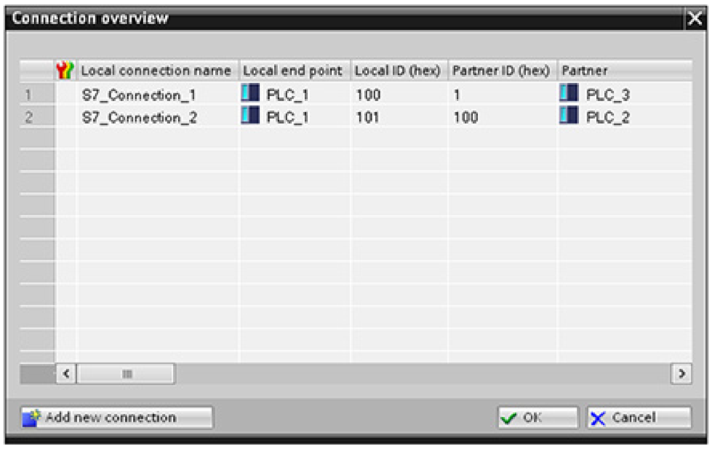
1. The user can change the current ID directly on the GET/PUT block. If the new ID belongs to an already existing connection, the connection is changed.
2. The user can change the current ID directly on the GET/PUT block, but the new ID does not already exist. A new S7 connection is created by the system.
3. The user can change the current ID through the "Connection overview" dialog: The user-input is synchronized with the ID-parameter on the corresponding GET/PUT block.

Note

The parameter "ID" of the GET/PUT block is not a connection name, but a numerical expression which is written like the following example: W#16#1

Connection name parameter

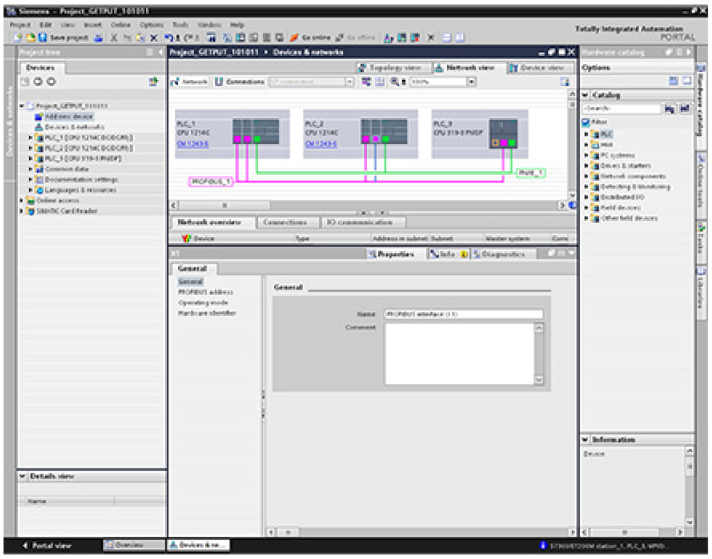
The connection name is editable through a special user control, the "Connection overview" dialog. This dialog offers all the available S7 connections which could be selected as an alternative for the current GET/PUT communication. The user can create a completely new connection in this table. Click the button to the right of the "Connection name" field to start the "Connection overview" dialog.



	Local connection name	Local end point	Local ID (hex)	Partner ID (hex)	Partner
1	S7_Connection_1	PLC_1	100	1	PLC_3
2	S7_Connection_2	PLC_1	101	100	PLC_2

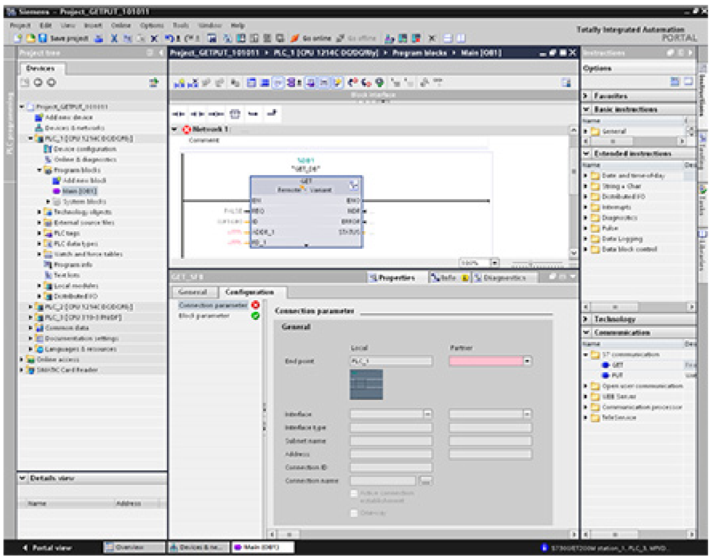
11.5.4.2 Configuring a CPU-to-CPU S7 connection

Given the configuration of PLC_1, PLC_2, and PLC_3 as shown in the figure below, insert GET or PUT blocks for "PLC_1".



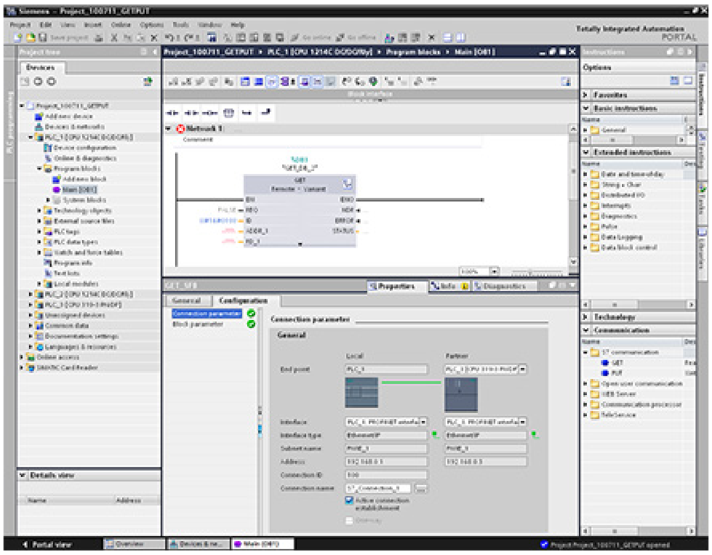
For the GET or PUT instruction, the "Properties" tab is automatically displayed in the inspector window with the following menu selections:

- "Configuration"
- "Connection parameters"



Configuring a PROFINET S7 connection

For the "Partner End point", select "PLC_3".



The system reacts with the following changes:

Table 11- 67 Connection parameter: General values

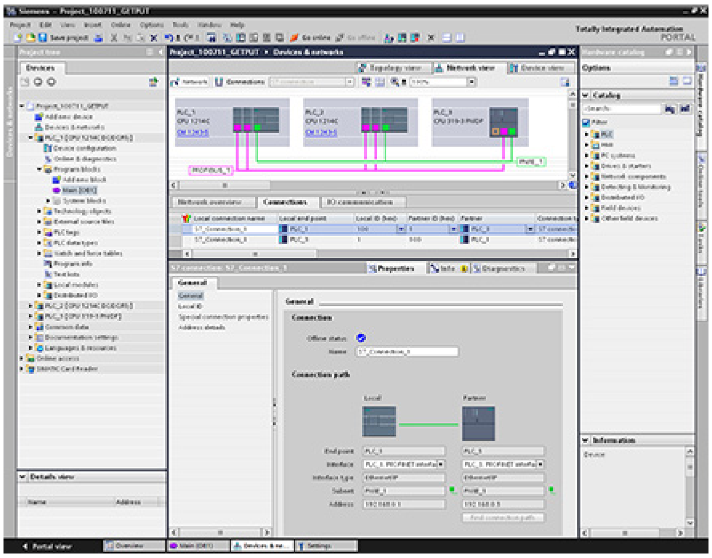
Parameter		Definition
Connection parameter: General	End point	"Local End point" contains "PLC_1" as read-only. "Partner End point" field contains "PLC_3[CPU319-3PN/DP]": • The color switches from red to white • The "Partner" device image is shown. • A connection line appears between the PLC_1- and PLC_3 device images (green Ethernet line).
	Interface	"Local Interface" contains "CPU1214C DC/DC/DC, PROFINET interface (R0/S1)". "Partner Interface" contains: "CPU319-3PN/DP, PROFINET interface (R0/S2)".
	Interface type	"Local Interface type" contains "Ethernet/IP"; control is read-only. "Partner Interface type" contains "Ethernet/IP"; control is read-only. Interface type images are shown at the right beside the Local and Partner "Interface type" (green Ethernet icon).
	Subnet name	"Local Subnet name" contains "PN/IE_1"; control is read only. "Partner Subnet name" contains "PN/IE_1"; control is read only.
	Address	"Local Address" contains the Local IP address; control is read only. "Partner Address" contains the Partner IP address; control is read only.
	Connection ID	"Connection ID" contains "100". In the Program editor, in the Main [OB1], the GET/PUT block "Connection ID" value also contains "100".

Parameter		Definition
	Connection name	"Connection name" contains the default connection name (for example, "S7_Connection_1"); control is enabled.
	Active connection estab-lishment	Checked and enabled to select the Local CPU as the active connection.
	One-way	Read-only and unchecked. Note: "PLC_1" (an S7-1200 CPU 1214CDC/DC/Rly) and "PLC_3" (an S7-300 CPU 319-3PN/DP) can both act as a server and a client in a PROFINET GET/PUT connection, allowing a two-way connection.

The GET/PUT icon in the Property View tree also changes from red to green.

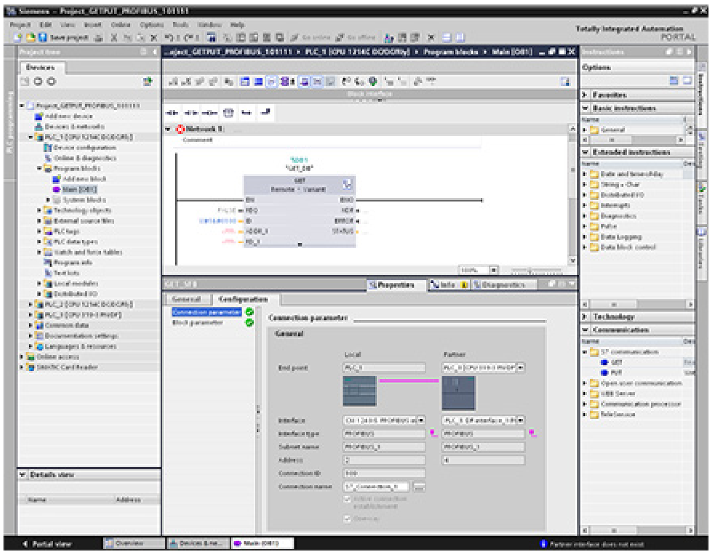
Completed PROFINET S7 connection

In the "Network view", a two-way S7 connection is shown in the "Connections" table between "PLC_1" and "PLC_3".



Configuring a PROFIBUS S7 connection

For the "Partner End point", select "PLC_3".



The system reacts with the following changes:

Table 11- 68 Connection parameter: General values

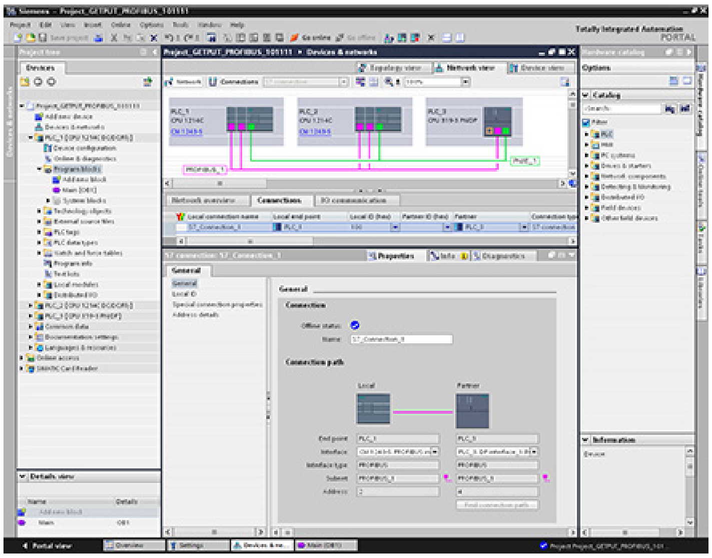
Parameter		Definition
Connection parameter: General	End point	"Local End point" contains "PLC_1" as read-only. "Partner End point" field contains "PLC_3[CPU319-3PN/DP]": • The color switches from red to white • The "Partner" device image is shown. • A connection line appears between the PLC_1- and PLC_3 device images (purple PROFIBUS line).
	Interface	"Local Interface" contains "CPU1214C DC/DC/DC, PROFIBUS interface (R0/S1)". "Partner Interface" contains: "CPU319-3PN/DP, PROFIBUS interface (R0/S2)".
	Interface type	"Local Interface type" contains "PROFIBUS"; control is read-only. "Partner Interface type" contains " PROFIBUS "; control is read-only. Interface type images are shown at the right beside the Local and Partner "Interface type" (purple PROFIBUS icon).
	Subnet name	"Local Subnet name" contains " PROFIBUS _1"; control is read only. "Partner Subnet name" contains " PROFIBUS _1"; control is read only.
	Address	"Local Address" contains the Local IP address; control is read only. "Partner Address" contains the Partner IP address; control is read only.
	Connection ID	"Connection ID" contains "100". In the Program editor, in the Main [OB1], the GET/PUT block "Connection ID" value also contains "100".

Parameter		Definition
	Connection name	"Connection name" contains the default connection name (for example, "S7_Connection_1"); control is enabled.
	Active connection establishment	Read-only, checked, and enabled to select the Local CPU as the active connection.
	One-way	Read-only and checked. Note: "PLC_3" (an S7-300 CPU319-3PN/DP) can act only as a server (cannot also be a client) in a PROFIBUS GET/PUT connection, allowing only a one-way connection.

The GET/PUT icon in the Property View tree also changes from red to green.

Completed PROFIBUS S7 connection

In the "Network view", a one-way S7 connection is shown in the "Connections" table between "PLC_1" and "PLC_3".



Local connection name	Local end point	Local ID (hex)	Partner ID (hex)	Partner	Connection type
S7_Connection_1	PLC_1	100		PLC_3	S7 connection

General

Connection

Offline status: ✓
Name: S7_Connection_1

Connection path

	Local	Partner
End point	PLC_1	PLC_3
Interface	CM 1243-5, PROFIBUS in	PLC_3, DP interface_1
Interface type	PROFIBUS	PROFIBUS
Subnet	PROFIBUS_1	PROFIBUS_1
Address	2	4

Web server 12

The Web server for the S7-1200 provides Web page access to data about your CPU and process data.

You can access the S7-1200 Web pages from a PC or from a mobile device. The Web server displays the pages in a format and size compatible with the device you use to access the Web pages. The Web server supports a minimum resolution of 240 x 240 pixels.

You use a Web browser to access the IP address of the S7-1200 CPU or the IP address of a Web server-enabled CP (communications processor) module (Page 793) in the local rack with the CPU to establish the connection. The S7-1200 supports multiple concurrent connections.

Standard Web pages

The S7-1200 includes standard Web pages that you can access from the Web browser of your PC (Page 791) or from a mobile device (Page 792):

- Introduction (Page 800) - entry point to the standard Web pages
- Start Page (Page 801) - general information about the CPU
- Identification (Page 802) - detailed information about the CPU including serial, order, and version numbers
- Module Information (Page 804) - information about the modules in the local rack and the ability to update firmware
- Communication (Page 807) - information about the network addresses, physical properties of the communication interfaces, and communication statistics
- Diagnostic Buffer (Page 803) - the diagnostic buffer
- Variable Status (Page 808) - CPU variables and I/O, accessible by address or PLC tag name
- File Browser (Page 810) - browser for files stored internally in the CPU or on a memory card, for example, data logs and recipes
- Login (Page 796) - log in as a different user, or log out.

These pages are built in to the S7-1200 CPU, and are available in English, German, French, Spanish, Italian, and Simplified Chinese. Some pages require additional user privileges (Page 789) that you configure in STEP 7 in order to view the page. For details about the standard Web pages, and how to access them, refer to the Standard Web pages (Page 794) section.

Note

S7-1200 CPUs do not include a separate firmware update standard Web page. The firmware update feature is included in the module information page.

User-defined Web pages

The S7-1200 also provides support for you to create user-defined Web pages that can access CPU data. You can develop these pages with the HTML authoring software of your choice, and include pre-defined "AWP" (Automation Web Programming) commands in your HTML code to access CPU data. Refer to the User-defined Web pages (Page 813) chapter for specific information on the development of user-defined Web pages, and the associated configuration and programming in STEP 7.

You can access the user-defined pages from either a PC or mobile device from the standard Web pages.

Web browser requirement

The Web server supports the following PC Web browsers:

- Internet Explorer 8.0
- Internet Explorer 9.0
- Mozilla Firefox 17.0.1
- Google Chrome 23.0
- Apple Safari 5.1.7 (Windows)
- Apple Safari 6.0.2 (Mac)

The Web server supports the following mobile device Web browsers:

- Internet Explorer 6.0 and earlier, for HMI panels
- Mobile Safari 7534.48.3 (iOS 5.0.1)
- Mobile Android Browser 2.3.4
- Mobile Google Chrome 23.0

For browser-related restrictions that can interfere with the display of standard or user-defined Web pages, see the Constraints (Page 855) section.

12.1 Enabling the Web server

You enable the Web server in STEP 7 from Device Configuration for the CPU to which you intend to connect.

To enable the Web server, follow these steps:

1. Select the CPU in the Device Configuration view.
2. In the inspector window, select "Web server" from the CPU properties.
3. Select the check box for "Activate web server on all modules of this device".
4. For increased security, select "Permit access only with HTTPS" to require secure access to the Web server.

WARNING

Unauthorized access to the CPU through the Web server

Unauthorized access to the CPU or changing PLC variables to invalid values could disrupt process operation and could result in death, severe personal injury and/or property damage.

Because enabling the Web server allows authorized users to perform operating mode changes, writes to PLC data, and firmware updates, Siemens recommends that you observe the following security practices:

- Enable access to the Web server only with the HTTPS protocol.
- Password-protect Web server user IDs (Page 789) with a strong password. Strong passwords are at least ten characters in length, mix letters, numbers, and special characters, are not words that can be found in a dictionary, and are not names or identifiers that can be derived from personal information. Keep the password secret and change it frequently.
- Do not extend the default minimum privileges of the "Everybody" user.
- Perform error-checking and range-checking on your variables in your program logic because Web page users can change PLC variables to invalid values.
- Use a secure Virtual Private Network (VPN) to connect to the S7-1200 PLC Web server from a location outside your protected network.

After you download the device configuration, you can use the standard Web pages to access the CPU. If you select "Enable" for "Automatic update", standard Web pages refresh every ten seconds.

If you created and enabled user-defined Web pages (Page 813), you can access them from the standard Web page menu.

Note

Device exchange: Replacing a V3.0 CPU with a V4.1 CPU

If you replace an existing V3.0 CPU with a V4.1 CPU (Page 1287) and convert your V3.0 project to a V4.1 project, note that STEP 7 and the V4.1 CPU retain the Web server settings for

- "Activate web server on this module"
- "Permit access only with HTTPS"

Note

If a "Download in RUN" (Page 1087) is in progress, standard and user-defined Web pages do not update data values or permit you to write data values until the download is complete. The Web server discards any attempts to write data values while a download is in progress.

Enabling other languages for the Web server

You can also optionally select other languages for the display of the standard Web pages. Select "User interface languages" from the Properties tab of your device configuration, and you can then assign a STEP 7 project language to one of the six languages that the Web server supports. After you download the device configuration, the standard Web pages provide a selector for the user interface language. If you do not select any languages, the default is English.

12.2 Configuring Web server users

You can configure users with various privilege levels for accessing the CPU through the Web server.

To configure Web server users and their associated privileges, follow these steps:

1. Select the CPU in the Device Configuration view.
2. In the inspector window, select "Web server" from the CPU properties and enable the Web server (Page 787).
3. Select "User management" in the Web server properties.
4. Enter user names, access levels, and passwords for the user logins that you want to provide.

After you download the configuration to the CPU, only authorized users can access Web server functions for which they have privileges.

Web server access levels

STEP 7 provides a default user named "Everybody" with no password. By default, this user has no additional privileges and can only view the Start (Page 801) and Introduction (Page 800) standard Web pages. You can, however, grant additional privileges to the "Everybody" user as well as other users that you configure:

- Query diagnostics
- Read tags
- Write tags
- Read tag status
- Write tag status
- Open user-defined pages
- Write in user-defined pages
- Read files
- Write/delete files
- Change operating mode
- Flash LEDs
- Perform firmware update
- Change system parameter
- Change application parameter

⚠ WARNING

Access to Web server

Granting privileges to the "Everybody" user makes it possible to log in to the Web server with no password. Unauthorized access to the CPU or changing PLC variables to invalid values could disrupt process operation and could result in death, severe personal injury and/or property damage.

Because the "Everybody" user when granted sufficient privileges can perform operating mode changes, writes to PLC data, and firmware updates with no password, Siemens recommends that you observe the following security practices

- Enable access to the Web server only with the HTTPS protocol.
- Password-protect Web server user IDs with a strong password. Strong passwords are at least ten characters in length, mix letters, numbers, and special characters, are not words that can be found in a dictionary, and are not names or identifiers that can be derived from personal information. Keep the password secret and change it frequently.
- Do not extend the default minimum privileges of the "Everybody" user.
- Perform error-checking and range-checking on your variables in your program logic because Web page users can change PLC variables to invalid values.
- Use a secure Virtual Private Network (VPN) to connect to the S7-1200 PLC Web server from a location outside your protected network.

12.3 Accessing the Web pages from a PC

You can access the S7-1200 standard Web pages from a PC or from a mobile device through the IP address of the S7-1200 CPU or the IP address of any Web server-enabled CP (Page 793) in the local rack.

To access the S7-1200 standard Web pages from a PC, follow these steps:

1. Ensure that the S7-1200 and the PC are on a common Ethernet network or are connected directly to each other with a standard Ethernet cable.
2. Open a Web browser and enter the URL "https://ww.xx.yy.zz", where "ww.xx.yy.zz" corresponds to the IP address of the S7-1200 CPU or the IP address of a CP in the local rack.

The Web browser opens the Introduction page.

Note

Use a secure Virtual Private Network (VPN) to connect to the S7-1200 PLC Web server from a location outside your protected network. Be aware also of any constraints (Page 855) that your Web environment or operating system might impose.

Alternatively, you can address your Web browser to a specific standard Web page. To do so, enter the URL in the form "https://ww.xx.yy.zz/<page>.html", where <page> corresponds to one of the standard Web pages:

- start (Page 801) - general information about the CPU
- identification (Page 802) - detailed information about the CPU including serial, order, and version numbers
- module (Page 804) - information about the modules in the local rack and the ability to update firmware
- communication (Page 807) - information about the network addresses, physical properties of the communication interfaces, and communication statistics
- diagnostic (Page 803) - the diagnostic buffer
- variable (Page 808) - CPU variables and I/O, accessible by address or PLC tag name
- filebrowser (Page 810) - browser for accessing data log files or recipe files stored internally in the CPU or on a memory card
- index (Page 800) - introduction page to enter the standard Web pages
- login (Page 796) - page to log in as a different user or log out. (Note that a login window is available from every page of the PC standard Web pages, but that the login page is necessary to log it from a mobile device.)

For example, if you enter "https://ww.xx.yy.zz/communication.html", the browser displays the communication page.

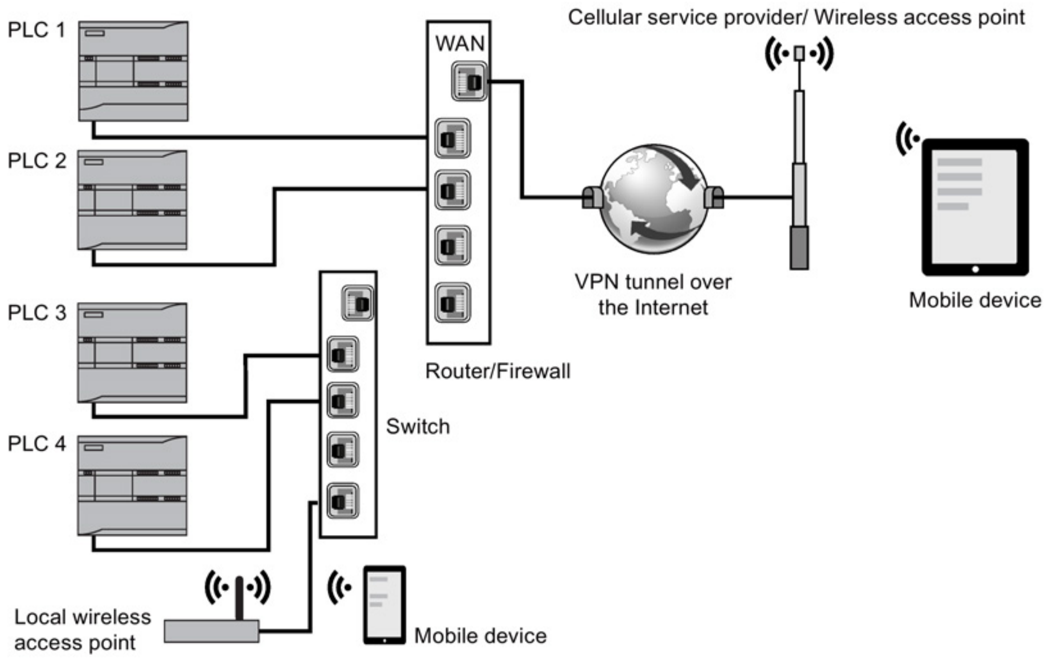
Secure access

Use a secure Virtual Private Network (VPN) to connect to the S7-1200 PLC Web server from a location outside your protected network. Require and use https:// instead of http:// for secure access (Page 787) to the standard Web pages. When you connect to the S7-1200 with https://, the Web site encrypts the session with a digital certificate. The Web server transmits the data securely and it is not accessible for anyone to view. You typically get a security warning that you can confirm with "Yes" to proceed to the standard Web pages. To avoid the security warning with each secure access, you can import the Siemens software certificate to your Web browser (Page 857).

12.4 Accessing the Web pages from a mobile device

To access an S7-1200 from a mobile device, you must connect your PLC to a network that connects to the Internet or to a local wireless access point. Use a secure Virtual Private Network (VPN) to connect a mobile device to the S7-1200 PLC Web server. You can use port forwarding in the wireless router to map the IP address of the PLC to an address by which a mobile device can access it from the Internet. To configure port forwarding, follow the instructions for the software configuration of your router. You can connect as many PLCs and switching devices as your router supports.

Without port forwarding, you can connect to a PLC, but only locally within range of the wireless signal.


PLC 1
PLC 2
PLC 3
PLC 4
WAN
Cellular service provider/ Wireless access point
VPN tunnel over the Internet
Mobile device
Router/Firewall
Switch
Local wireless access point
Mobile device

In this example, a mobile device that is within range of the local wireless access point can connect to PLC 3 and PLC 4 by their IP addresses. From the Internet outside the local wireless range, a mobile device can connect to PLC 1 and PLC 2 using the port forwarded address for each PLC.

To access the standard Web pages, you must have access to a cellular service or wireless access point. To access a PLC from the Internet, enter the port forwarded address in the Web browser of your mobile device to access the PLC, for example http://ww.xx.yy.zz:pppp or https:/ww.xx.yy.zz:pppp, where ww.xx.yy.zz is the address of the router and pppp is the port assignment for a specific PLC.

For local access through a local wireless access point, enter the IP address of the S7-1200 CPU or a Web server-enabled CP (Page 793) in the local rack: http://ww.xx.yy.zz or https::/ww.xx.yy.zz. You can also navigate to a specific Web page by name as described in Accessing the Web pages from a PC (Page 791).

For increased security, configure the Web server to be accessible only by secure access (HTTPS) (Page 787).

12.5 Using a CP module to access Web pages

Regardless of whether you access the Web server from a PC or a mobile device, you can connect to standard Web pages through one of the following CP modules when you have configured it in STEP 7 and installed it in the local rack with the S7-1200 CPU:

- CP 1242-7 GPRS V2
- CP 1243-7 LTE-EU

You use the Start standard Web page (Page 801) to access the Web pages through these CP modules. The Start page displays all configured and installed CP modules that you have in your local rack, but you can only access Web pages from the ones listed above.

Note

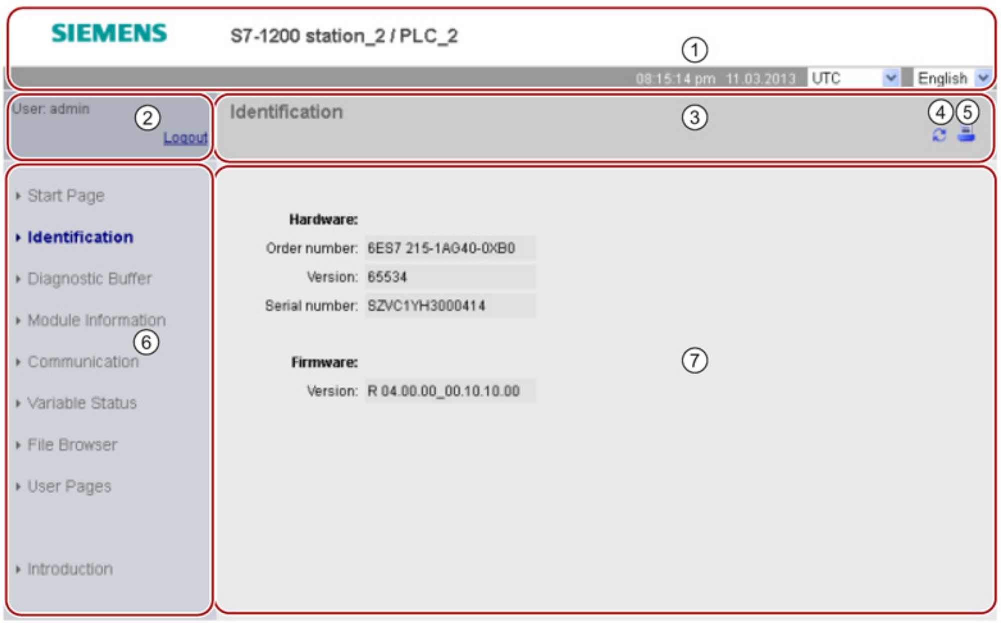
Access to standard Web pages when Web server-enabled CPs are in the local rack

You might observe delays up to one or two minutes when connecting to the S7-1200 standard Web pages when Web server-enabled CPs are in the local rack. If the pages do not appear to be available, or you get errors, just wait one or two minutes and refresh the page.

12.6 Standard Web pages

12.6.1 Layout of the standard Web pages

Each of the S7-1200 standard Web pages has a common layout with navigational links and page controls. Regardless of whether you are viewing the page on a PC or on a mobile device, each page has the same content area, but the layout and navigation controls vary based on the screen size and resolution of the device. On a standard PC or large mobile device the layout of a standard Web page appears as follows:



Hardware:
Order number: 6ES7 215-1AG40-0XB0
Version: 65534
Serial number: SZVC1YH3000414

Firmware:
Version: R 04.00.00_00.10.10.00

① Web server header with selector to display PLC Local time or UTC time, and a selector for the display language (Page 162)

② Log in or log out

③ Standard Web page header with name of the page that you are viewing. This example is the CPU Identification page. Some of the standard Web pages, such as module information, also display a navigation path here if multiple screens of that type can be accessed.

④ Refresh icon: for pages with automatic refresh, enables or disables the automatic refresh function; for pages without automatic refresh, causes the page to update with current data

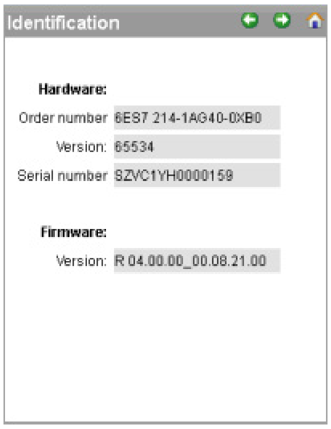
⑤ Print icon: prepares and displays a printable version of the information available from the displayed page

⑥ Navigation area to switch to another page

⑦ Content area for specific standard Web page that you are viewing. This example is the CPU Identification page.

Mobile device layout

On a device where the width is less than 768 pixels, the Web server displays a mobile version of each page. The page omits the navigation area, login area, and the header area, and includes buttons for advancing backward and forward through the Web pages, and a Home page button that takes you to a Navigation page. You can also use the navigation controls provided with your mobile device for navigation. For example, on a mobile device with a screen width less than 768 pixels the Identification page appears as follows in the vertical orientation:


Identification

Hardware:

Order number	6ES7 214-1AG40-0XB0
Version:	65534
Serial number	SZVC1YH0000159

Firmware:

Version:	R 04.00.00_00.08.21.00

Note that the standard Web page illustrations in this chapter represent the standard PC Web page appearance. Each standard Web page has an equivalent mobile page appearance.

Note

CP module standard Web pages

Certain CP modules (Page 793) provide standard Web pages that are similar in appearance and functionality to the S7-1200 CPU standard Web pages. Refer to your CP documentation for descriptions of the CP standard Web pages.

12.6.2 Logging in and user privileges

Each of the PC standard Web pages provides a login window above the navigation pane. Due to space considerations, the mobile Web pages provide a separate Login page. The S7-1200 supports multiple user logins with various access levels (privileges):

- Query diagnostics
- Read tags
- Write tags
- Read tag status
- Write tag status
- Open user-defined pages
- Write in user-defined pages
- Read files
- Write/delete files
- Change operating mode
- Flash LEDs
- Perform firmware update
- Change system parameter
- Change application parameter

You configure user roles, associated access levels (privileges), and passwords (Page 789) in the Web server user management properties of the STEP 7 device configuration of the CPU.

Logging in

STEP 7 provides a default user named "Everybody" with no password. By default, this user has no additional privileges and can only view the Start (Page 801) and Introduction (Page 800) standard Web pages. You can, however, grant additional privileges to the "Everybody" user as well as other users that you configure:

⚠ WARNING

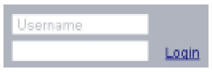
Access to Web server

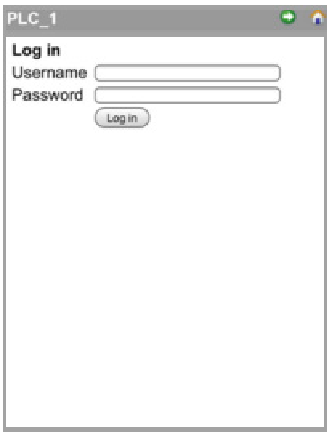
Granting privileges to the "Everybody" user makes it possible to log in to the Web server with no password. Unauthorized access to the CPU or changing PLC variables to invalid values could disrupt process operation and could result in death, severe personal injury and/or property damage.

Because the "Everybody" user when granted sufficient privileges can perform operating mode changes, writes to PLC data, and firmware updates with no password, Siemens recommends that you observe the following security practices

- Enable access to the Web server only with the HTTPS protocol.
- Password-protect Web server user IDs (Page 789) with a strong password. Strong passwords are at least ten characters in length, mix letters, numbers, and special characters, are not words that can be found in a dictionary, and are not names or identifiers that can be derived from personal information. Keep the password secret and change it frequently.
- Do not extend the default minimum privileges of the "Everybody" user.
- Perform error-checking and range-checking on your variables in your program logic because Web page users can change PLC variables to invalid values.
- Use a secure Virtual Private Network (VPN) to connect to the S7-1200 PLC Web server from a location outside your protected network.

To perform certain actions such as changing the operating mode of the controller, writing values to memory, and updating the CPU firmware you must have the required privileges. Note that if you have set the protection level of the CPU (Page 197) to "Complete protection (no access)", then the "Everybody" user cannot access the Web server.





The log in frame is near the upper left corner on each standard Web page when displayed from a PC or a wide mobile device.

The Log In page is a separate page on small mobile devices, and is selectable from the Home page.

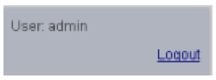
To log in, follow these steps:

1. Enter the user name for the Username field.
2. Enter the user password in the Password field.

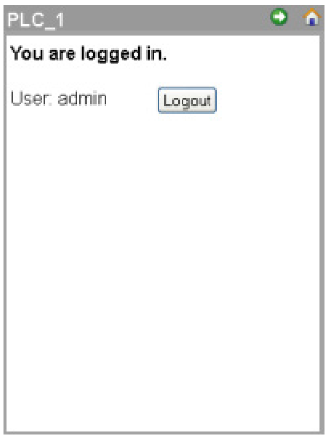
Your login times out after thirty minutes of inactivity. If the currently-loaded page is continually refreshing, the login does not time out.

If you encounter any errors logging in, return to the Introduction page (Page 800) and download the Siemens security certificate (Page 857). You can then log in with no errors.

Logging out



To log out, simply click the "Logout" link from any page when viewing from a PC or wide mobile device.



From a small mobile device, navigate to the Log-in/Logout page from the Home page and tap the "Log-out" button.

After you log out, you can only access and view standard Web pages according to the privileges of the "Everybody" user. Each of the standard Web page descriptions defines the required privileges for that page.

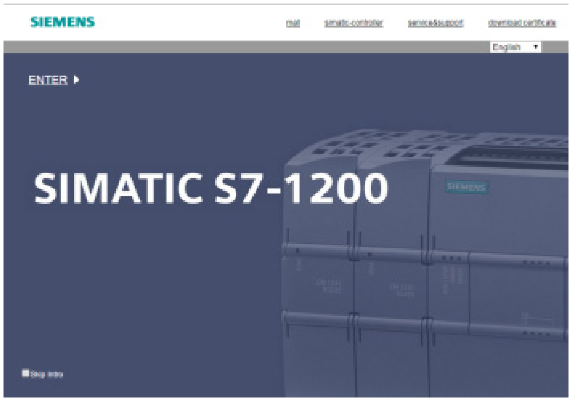
Note

Log off prior to closing Web server

If you have logged in to the Web server, be sure to log off prior to closing your Web browser. The Web server supports a maximum of seven concurrent logins.

12.6.3 Introduction

The Introduction page is the welcome screen for entry into the S7-1200 standard Web pages.



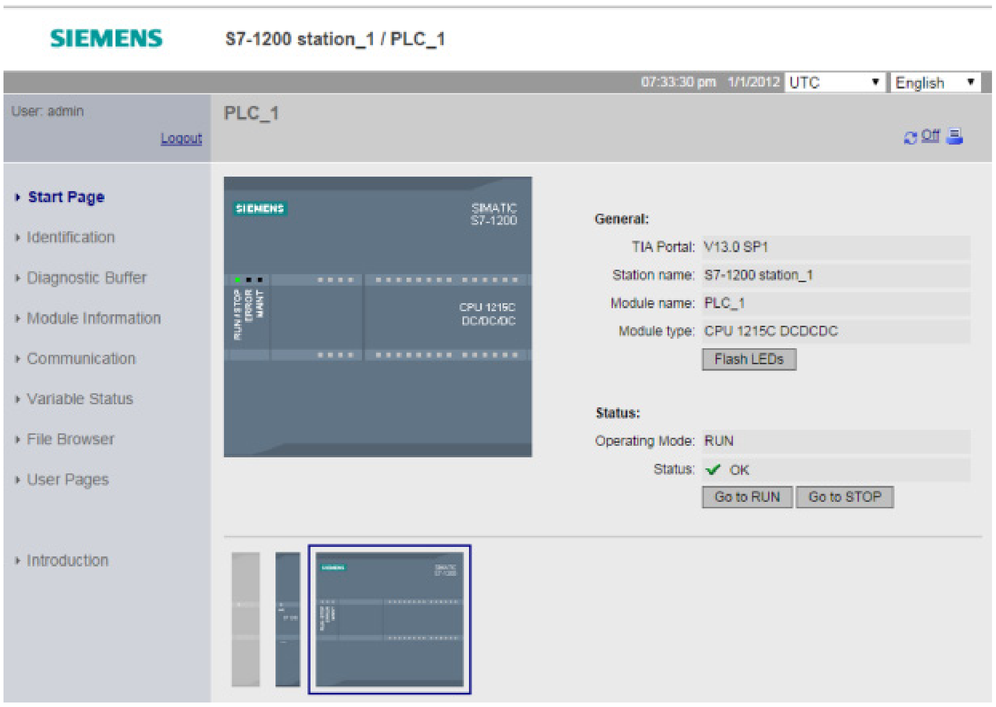
From this page, you click "Enter" to access the S7-1200 standard Web pages. At the top of the screen are links to useful Siemens Web sites, as well as a link to download the Siemens security certificate (Page 857). You can also choose to skip the introduction page on future accesses to the Web server.

12.6.4 Start

The Start page displays a representation of the CPU or CP to which you are connected and lists general information about the device. For the CPU, you can use the buttons to change the operating mode and flash the LEDs, if you have logged in (Page 796) with those specific privileges.

The bottom portion of the screen is visible if you have configured and installed Web server-enabled CP modules (Page 793) in the local rack with the S7-1200 CPU. You can hover over and click a Web server-enabled CP module to access the standard Web pages. Refer to the documentation for your CP module for information about the CP module Web pages. You see the name of CP module when you hover over it.

The Web server also displays any other CM and CP modules in the local rack, but you cannot click them as they do not contain Web pages. The module appearance for these CMs and CPs are light gray (desensitized) to indicate that they are display-only and not clickable modules.



General:

TIA Portal:	V13.0 SP1
Station name:	S7-1200 station_1
Module name:	PLC_1
Module type:	CPU 1215C DCDCDC

Status:

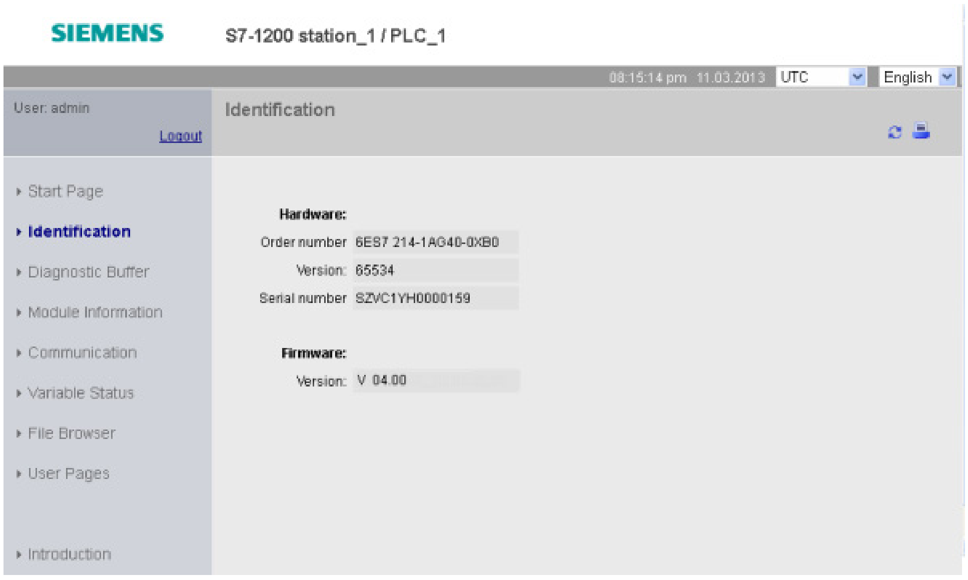
Operating Mode:	RUN
Status:	✓ OK

Note that the S7-1200 fail-safe CPUs display additional data on this page related to functional safety.

12.6.5 Identification

The Identification page displays identifying characteristics of the CPU:

- Serial number
- Article number
- Version information



Hardware:

Order number	6ES7 214-1AG40-0XB0
Version:	65534
Serial number	SZVC1YH0000159

Firmware:

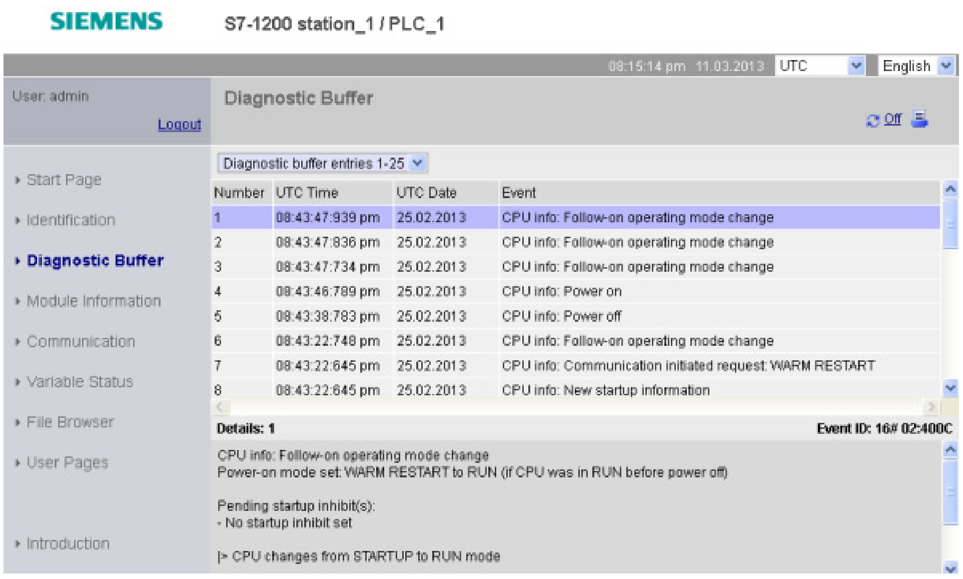
Version:	V 04.00

Viewing the Identification page requires the "query diagnostics" privilege (Page 789).

12.6.6 Diagnostic Buffer

The diagnostic buffer page displays diagnostic events. From the selector on the left, you can choose what range of diagnostic buffer entries to display, either 1 to 25 or 26 to 50. From the selector on the right, you can choose whether to display the times in UTC times or PLC local times. The top part of the page displays the diagnostic entries with the time and date of when the event occurred.

From the top part of the page, you can select any individual entry to show detailed information about that entry in the bottom part of the page.



Diagnostic Buffer

Diagnostic buffer entries 1-25

Number	UTC Time	UTC Date	Event
1	08:43:47:939 pm	25.02.2013	CPU info: Follow-on operating mode change
2	08:43:47:836 pm	25.02.2013	CPU info: Follow-on operating mode change
3	08:43:47:734 pm	25.02.2013	CPU info: Follow-on operating mode change
4	08:43:46:789 pm	25.02.2013	CPU info: Power on
5	08:43:38:783 pm	25.02.2013	CPU info: Power off
6	08:43:22:748 pm	25.02.2013	CPU info: Follow-on operating mode change
7	08:43:22:645 pm	25.02.2013	CPU info: Communication initiated request: WARM RESTART
8	08:43:22:645 pm	25.02.2013	CPU info: New startup information

Details: 1 **Event ID: 16# 02:400C**

CPU info: Follow-on operating mode change
Power-on mode set: WARM RESTART to RUN (if CPU was in RUN before power off)

Pending startup inhibit(s):
- No startup inhibit set

|> CPU changes from STARTUP to RUN mode

Viewing the Diagnostic Buffer page requires the "query diagnostics" privilege (Page 789).

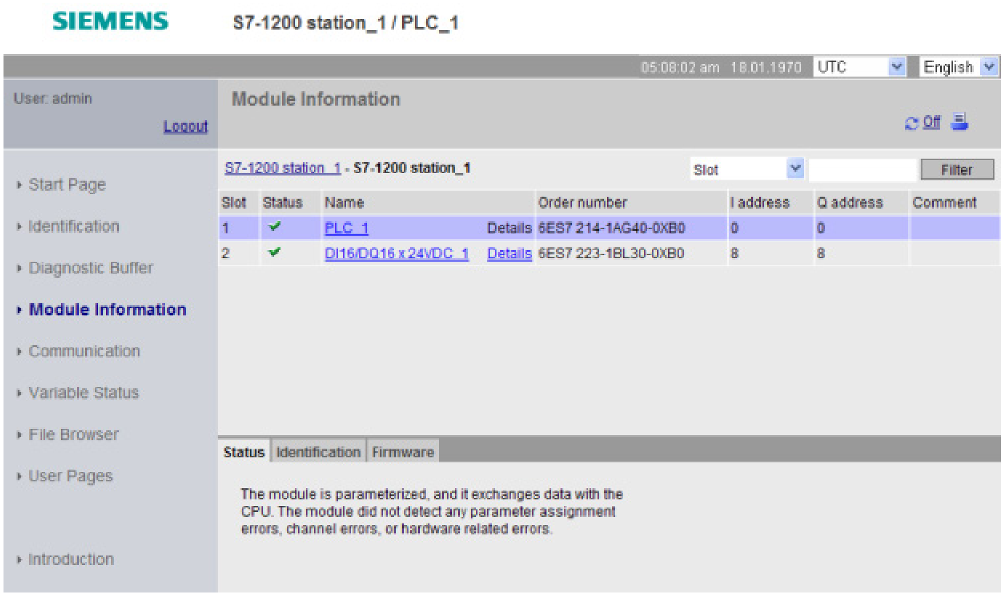
12.6.7 Module Information

The module information page provides information about all the modules in the local rack. The top section of the screen shows a summary of the modules, and the bottom section shows status, identification and firmware information of the selected module. The module information page also provides the capability to perform a firmware update.

Viewing the Module Information page requires the "query diagnostics" privilege (Page 789).

Module information: Status tab

The status tab in the bottom section of the module information page displays a description of the current status of the module that is selected in the top section.



Slot	Status	Name		Order number	I address	Q address	Comment
1	✓	PLC_1	Details	6ES7 214-1AG40-0XB0	0	0	
2	✓	DI16/DQ16 x 24VDC_1	Details	6ES7 223-1BL30-0XB0	8	8	

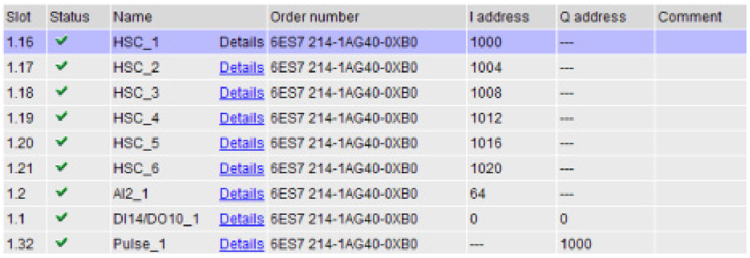
The module is parameterized, and it exchanges data with the CPU. The module did not detect any parameter assignment errors, channel errors, or hardware related errors.

Note

The mobile device module information page displays the "I address", "Q address", and "Comment" information on the Identification tab rather than as columns in the main module information table.

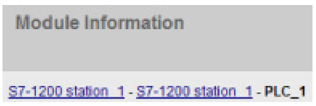
Drilling down

You can select a link in the top section to drill down to the module information for that particular module. Modules with submodules have links for each submodule. The type of information that is displayed varies with the module selected. For example, the module information dialog initially displays the name of the SIMATIC 1200 station, a status indicator, and a comment. If you drill down to the CPU, the module information displays the name of the digital and analog inputs and outputs that the CPU model provides (for example, "DI14/DO10", "AI2"), addressing information for the I/O, status indicators, slot numbers, and comments.



Slot	Status	Name		Order number	I address	Q address	Comment
1.16	✓	HSC_1	Details	6ES7 214-1AG40-0XB0	1000	---	
1.17	✓	HSC_2	Details	6ES7 214-1AG40-0XB0	1004	---	
1.18	✓	HSC_3	Details	6ES7 214-1AG40-0XB0	1008	---	
1.19	✓	HSC_4	Details	6ES7 214-1AG40-0XB0	1012	---	
1.20	✓	HSC_5	Details	6ES7 214-1AG40-0XB0	1016	---	
1.21	✓	HSC_6	Details	6ES7 214-1AG40-0XB0	1020	---	
1.2	✓	AI2_1	Details	6ES7 214-1AG40-0XB0	64	---	
1.1	✓	DI14/DO10_1	Details	6ES7 214-1AG40-0XB0	0	0	
1.32	✓	Pulse_1	Details	6ES7 214-1AG40-0XB0	---	1000	

As you drill down, the module information page shows the path you have followed. You can click any link in this path to return to a higher level.



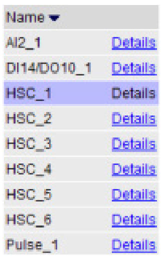
Module Information

S7-1200 station_1 - S7-1200 station_1 - PLC_1

Sorting fields

When the list displays multiple modules, you can click the column header of a field to sort it either up or down by that field.

Note: This feature is not yet available for the Chinese Module Information page.



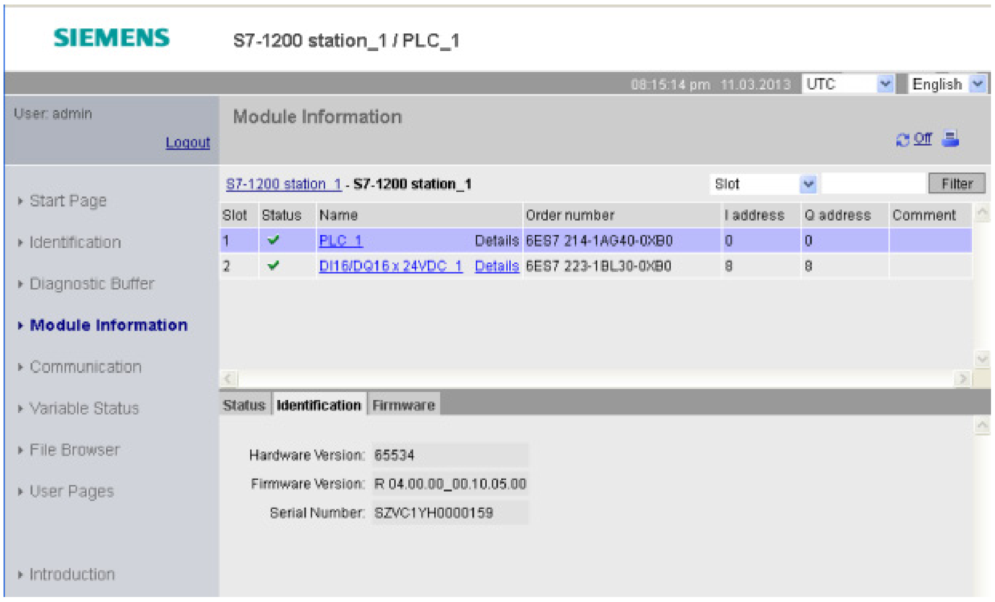
Name ▼	
AI2_1	Details
DI14/DO10_1	Details
HSC_1	Details
HSC_2	Details
HSC_3	Details
HSC_4	Details
HSC_5	Details
HSC_6	Details
Pulse_1	Details

Filtering the module information

You can filter any field in the module information list. From the drop-down list, select the field name for which you want to filter the data. Enter text in the associated text box and click the Filter link. The list updates to show you modules that correspond to your filtering criteria.

Module information: Identification tab

The identification tab displays the serial number and version numbers of the selected module.



S7-1200 station_1 - S7-1200 station_1

Slot	Status	Name		Order number	I address	Q address	Comment
1	✓	PLC_1	Details	6ES7 214-1AG40-0XB0	0	0	
2	✓	DI16/DQ16 x 24VDC_1	Details	6ES7 223-1BL30-0XB0	8	8	

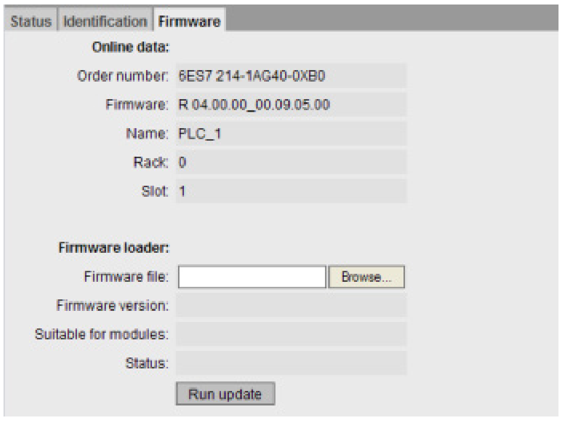
Hardware Version:	65534
Firmware Version:	R 04.00.00_00.10.05.00
Serial Number:	SZVC1YH0000159

Module information: Firmware tab

The firmware tab of the module information page displays information about the firmware of the selected module. If you have the "perform firmware update" privilege (Page 789), you can also perform a firmware update of the CPU or other modules in the local rack that support firmware update.

Note

You can only update S7-1200 CPUs of version 3.0 and higher with the Update Firmware feature.



Online data:

Order number:	6ES7 214-1AG40-0XB0
Firmware:	R 04.00.00_00.09.05.00
Name:	PLC_1
Rack:	0
Slot:	1

Firmware loader:

Firmware file:	
Firmware version:	
Suitable for modules:	
Status:	

The CPU must be in STOP mode to perform a firmware update. When the CPU is in STOP mode, click the Browse button to navigate to and select a firmware file. Firmware updates are available on the customer support Web site (http://support.automation.siemens.com).

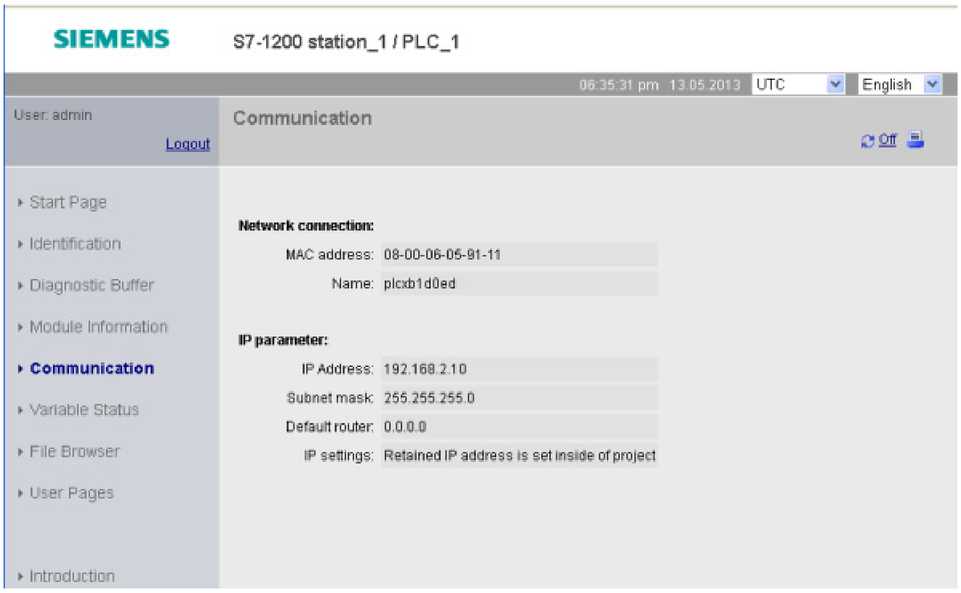
During the update, the page displays a message showing that the update is in progress. After the update completes, the page displays the article number and version number of the updated firmware. If you updated the firmware for the CPU or a signal board, the Web server restarts the CPU.

Note

You can also perform a firmware update from STEP 7 (Page 1074), or by using a memory card (Page 141).

12.6.8 Communication

The communication page displays the parameters of the connected CPU, including the MAC address, the IP address, and IP settings of the CPU.



Network connection:

MAC address:	08-00-06-05-91-11
Name:	plcxb1d0ed

IP parameter:

IP Address:	192.168.2.10
Subnet mask:	255.255.255.0
Default router:	0.0.0.0
IP settings:	Retained IP address is set inside of project

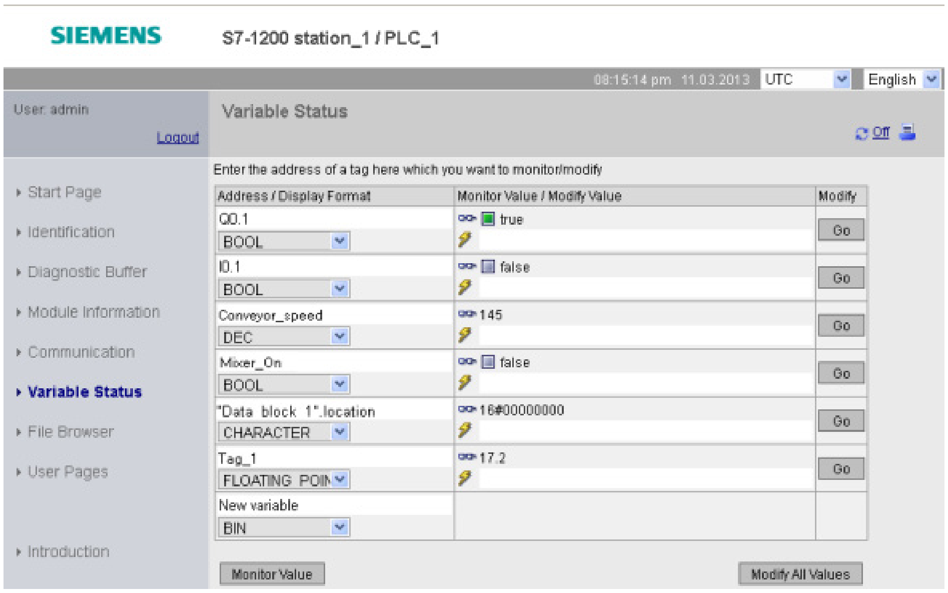
Viewing the Communication page requires the "query diagnostics" privilege.

12.6.9 Variable Status

The Variable Status page allows you to view any of the I/O or memory data in your CPU. You can enter a direct address (such as I0.0), a PLC tag name, or a tag from a specific data block. For data block tags, you enclose the data block name in double quotation marks. For each monitor value you can select a display format for the data. You can continue entering and specifying values until you have as many as you want within the limitations for the page. The monitor values show up automatically and refresh by default, unless you click the "Off" icon in the upper right area of the page. When refresh is disabled, you can click "On" to re-enable automatic refresh.

Viewing the Variable Status page requires the "read variable status" privilege. For a tag to be viewable on the Variable Status page, you must configure it as "Accessible from HIM" in STEP 7.

If you login as a user with the "write variable status" privilege (Page 796), you can also modify data values. Enter any values that you wish to set in the appropriate "Modify Value" field. Click the "Go" button beside a value to write that value to the CPU. You can also enter multiple values and click "Modify All Values" to write all of the values to the CPU. The buttons and column labels for modifying only appear if you have the "write variable status" privilege.



Enter the address of a tag here which you want to monitor/modify

Address / Display Format	Monitor Value / Modify Value	Modify
Q0.1 / BOOL	🟩 true	Go
I0.1 / BOOL	▢ false	Go
Conveyor_speed / DEC	145	Go
Mixer_On / BOOL	▢ false	Go
"Data_block_1".location / CHARACTER	16#00000000	Go
Tag_1 / FLOATING POIN	17.2	Go
New variable / BIN		

If you leave the Variable Status page and return, the Variable Status page does not retain your entries. You can bookmark the page and return to the bookmark to see the same entries. If you do not bookmark the page, you must re-enter the variables.

Note

Be aware of the following issues when using the standard Variable Status page:

- Enclose all string modifications in single quotes.
- The Variable Status page cannot monitor or modify tags that contain any of the following characters: &, <, (, +, ,(comma), ., [,], $, or %. For example, you cannot monitor the tag "Clock_2.5Hz".
- The Variable Status page does not allow you to modify a string longer than 198 characters.
- When using exponential notation to enter a value for a Real or LReal data type in the Variable Status page:
  - To enter a real-number value (Real or LReal) with a positive exponent (such as +3.402823e+25), enter the value in either of the following formats:

    +3.402823e25

    +3.402823e+25
  - To enter real-number value (Real or LReal) with a negative exponent (such as +3.402823e-25), enter the value as follows:

    +3.402823e-25
  - Be sure that the mantissa portion of the real value in exponential notation includes a decimal point. Failure to include a decimal point results in the modification of the value to an unexpected integer value. For example, enter -1.0e8 rather than -1e8.
- The Variable Status page supports only 15 digits for an LReal value (regardless of the location of the decimal point). Entering more than 15 digits creates a rounding error.

Limitations on the Variable Status page:

- The maximum number of variable entries per page is 50.
- The maximum number of characters for the URL corresponding to the Variable Status page is 2083. You can see the URL that represents your current variable page in the address bar of your browser.
- For the character display format, the page displays hexadecimal values if the actual CPU values are not valid ASCII characters as interpreted by the browser.

  Note

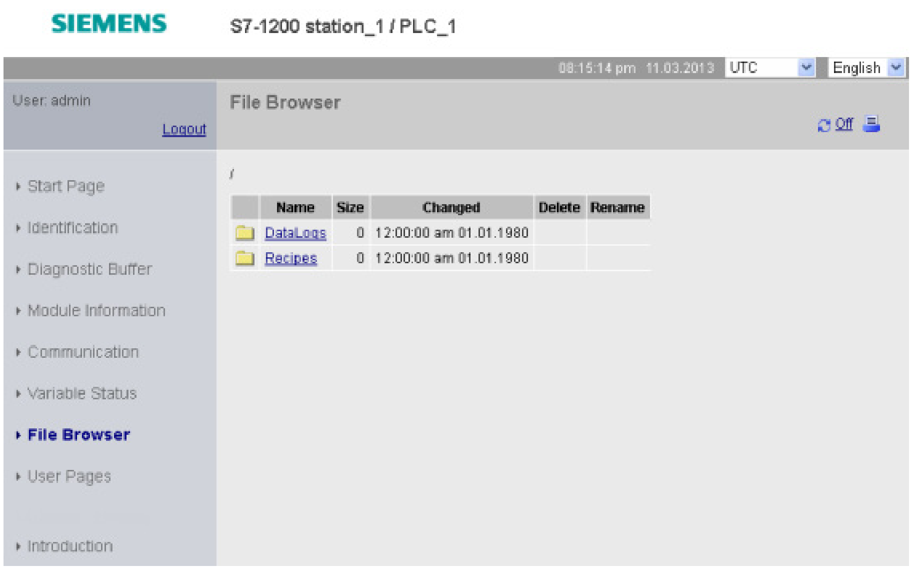
  If a tag name displays special characters such that it is rejected as an entry on the Variable Status page, you can enclose the tag name in double quotation marks. In most cases, the Variable Status page will then recognize the tag name.

12.6.10 File Browser

The File Browser page provides access to files in the internal load memory of the CPU or on the memory card (external load memory). The file browser page initially displays the root folder of the load memory, which contains the "DataLogs" and "Recipes" folders, but also displays any other folders that you might have created, if using a memory card.

The type of file access you have to the files and folders depends on your user privileges (Page 789). Any user with "read files" privileges can view the files and folders with the file browser. You cannot delete the DataLogs folder or Recipes folder regardless of your login privileges, but if you had made custom folders on the memory card, you can delete those folders if you have logged in as a user with "write/delete files" privileges.

Click a folder to access the individual files in the folder.



/

Name	Size	Changed	Delete	Rename
DataLogs	0	12:00:00 am 01.01.1980		
Recipes	0	12:00:00 am 01.01.1980		

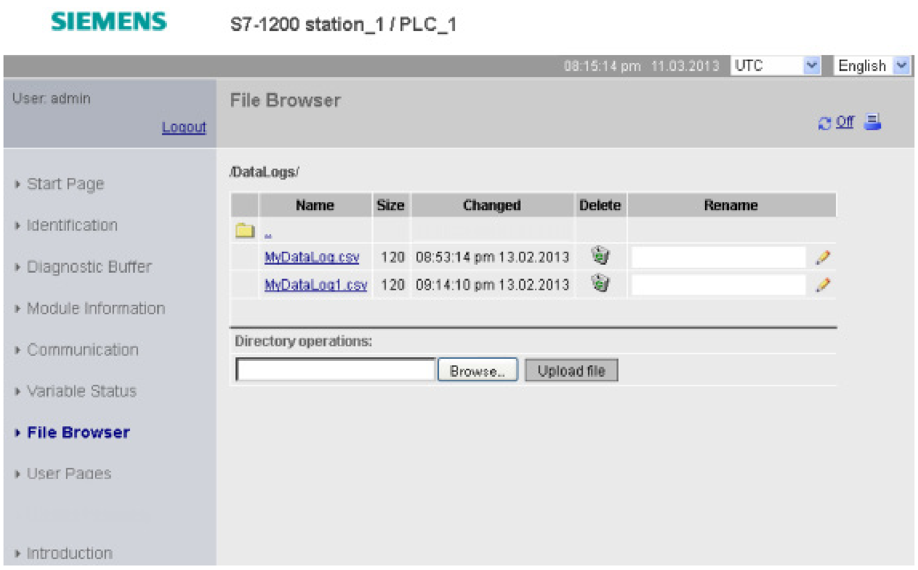
Data logs

From the "Data Logs" folder, you can open any of the data log files. If you have logged in with the "write/delete files" privilege (Page 789), you can also delete, rename, and upload files. The data log files are in comma-separated values (CSV) file format. You can save them to your computer or open them in Microsoft Excel (default) or another program.

Note

Time stamps for data logs

The Web server displays the time stamps for the data logs in either UTC time or PLC local time depending upon your selection at the top of the page.



/DataLogs/

Name	Size	Changed	Delete	Rename
..				
MyDataLog.csv	120	08:53:14 pm 13.02.2013		
MyDataLog1.csv	120	09:14:10 pm 13.02.2013		

Note: The "Delete" and "Rename" options are not available if you are not logged in with the "write/delete files" privilege.

Note

Data log management

Keep no more than 1000 data logs in a file system. Exceeding this number can prevent the Web server from having enough CPU resources to display the data logs.

If you find that the File Browser Web page is not able to display the data logs, then you must place the CPU in STOP mode in order to display and delete data logs.

Manage your data logs to ensure that you only keep the number that you need to maintain, and do not exceed 1000 data logs.

Working with a data log in Excel

The data log file is in USA/UK comma-separated values format (CSV). To open it in Excel on non-USA/UK systems, you must import it into Excel with specific settings (Page 859).

Recipe files

Like the data logs folder, the recipe folder displays any recipe files that are present in load memory. Recipe files are also in CSV format, and you can open them in Microsoft Excel, or another program. Like data logs, you must have modify privileges in order to delete, modify and save, rename or upload recipe files.

Uploading files and automatic page refresh

If you begin a file upload, the upload operation continues as long as you remain on the File Browser Web page. If you enabled automatic update to refresh the Web server pages every ten seconds, then whenever a page refresh occurs you see the incremental progress of the file upload operation. For example, if you are uploading a 2 MB file, you might see updates that show the file size in bytes at 2500, 5000, 10000, 15000, and 20000 as the file upload progresses.

If you leave the File Browser page before the upload completes, you do not get the complete file. When you return, the File Browser page displays the file name and the size of the file at the time that the upload stopped. You see no other indication that it is an incomplete file. To be sure you upload the complete file, remain on the File Browser page until the displayed file size reaches the actual size of the file.

Additional information

For information on programming with the data log instructions, and importing (Page 417) and exporting (Page 415) recipes, see the Recipes and Data logs (Page 411) chapter.

12.7 User-defined Web pages

The S7-1200 Web server also provides the means for you to create your own application-specific HTML pages that incorporate data from the PLC.

WARNING

Unauthorized access to the CPU through user-defined Web pages

Unauthorized access to the CPU through user-defined Web pages could disrupt process operation, which could result in death, severe personal injury and/or property damage.

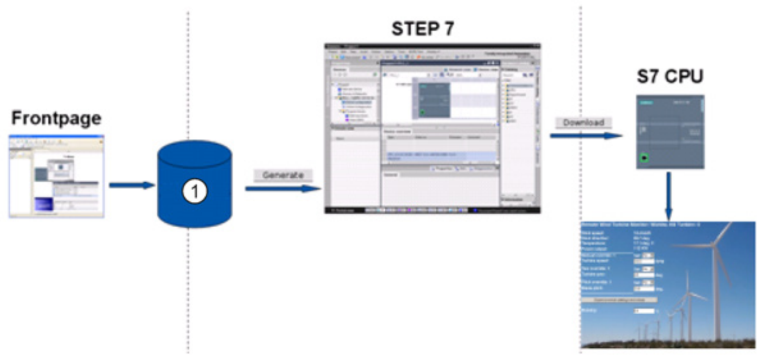
Insecure coding of user-defined Web pages introduces security vulnerabilites such as cross-site scripting (XSS), code injection, and others.

Protect your S7-1200 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).

You create user-defined Web pages using the HTML editor of your choice and download them to the CPU where they are accessible from the standard Web page menu. This process involves several tasks:

- Creating HTML pages with an HTML editor, such as Microsoft Frontpage (Page 814)
- Including AWP commands in HTML comments in the HTML code (Page 815):The AWP commands are a fixed set of commands that Siemens provides for accessing CPU information.
- Configuring STEP 7 to read and process the HTML pages (Page 830)
- Generating blocks from the HTML pages (Page 830)
- Programming STEP 7 to control the use of the HTML pages (Page 831)
- Compiling and downloading the blocks to the CPU (Page 832)
- Accessing the user-defined Web pages from your PC (Page 833)

This process is illustrated below:



① HTML files with embedded AWP commands

12.7.1 Creating HTML pages

You can use the software package of your choice to create your own HTML pages for use with the Web server. Be sure that your HTML code is compliant to the HTML standards of the W3C (World Wide Web Consortium). STEP 7 does not perform any verification of your HTML syntax.

You can use a software package that lets you design in WYSIWYG or design layout mode, but you need to be able to edit your HTML code in pure HTML form. Most Web authoring tools provide this type of editing; otherwise, you can always use a simple text editor to edit the HTML code. Include the following line in your HTML page to set the charset for the page to UTF-8:

```
<meta http-equiv="content-type" content="text/html; charset=utf-8">
```

Also be sure to save the file from the editor in UTF-8 character encoding.

You use STEP 7 to compile everything in your HTML pages into STEP 7 data blocks. These data blocks consist of one control data block that directs the display of the Web pages and one or more fragment data blocks that contain the compiled Web pages. Be aware that extensive sets of HTML pages, particularly those with lots of images, require a significant amount of load memory space (Page 833) for the fragment DBs. If the internal load memory of your CPU is not sufficient for your user-defined Web pages, use a memory card (Page 132) to provide external load memory.

To program your HTML code to use data from the S7-1200, you include AWP commands (Page 815) as HTML comments. When finished, save your HTML pages to your PC and note the folder path where you save them.

Note

The file size limit for HTML files containing AWP command is 64 kilobytes. You must keep your file size below this limit.

Refreshing user-defined Web pages

User-defined Web pages do not automatically refresh. It is your choice whether to program the HTML to refresh the page or not. For pages that display PLC data, refreshing periodically keeps the data current. For HTML pages that serve as forms for data entry, refreshing can interfere with the user entering data. If you want your entire page to automatically refresh, you can add this line to your HTML header, where "10" is the number of seconds between refreshes:

```
<meta http-equiv="Refresh" content="10">
```

You can also use JavaScript or other HTML techniques to control page or data refreshing. For this, refer to documentation on HTML and JavaScript.

12.7.2 AWP commands supported by the S7-1200 Web server

The S7-1200 Web server provides AWP commands that you embed in your user-defined Web pages as HTML comments for the following purposes:

- Reading variables (Page 817)
- Writing variables (Page 818)
- Reading special variables (Page 820)
- Writing special variables (Page 821)
- Defining enum types (Page 823)
- Assigning variables to enum types (Page 824)
- Creating fragment data blocks (Page 826)

General syntax

Except for the command to read a variable, the AWP commands are of the following syntax:
`<!-- AWP_ <command name and parameters> -->`

You use the AWP commands in conjunction with typical HTML form commands to write to variables in the CPU.

The descriptions of the AWP commands in the following pages use the following conventions:

- Items enclosed in brackets [ ] are optional.
- Items enclosed in angle brackets < > are parameter values to be specified.
- Quotation marks are a literal part of the command. They must be present as indicated.
- Special characters in tag or data block names, depending on usage, must be escaped or enclosed in quotation marks (Page 828).

Use a text editor or HTML editing mode to insert AWP commands into your pages.

Note

Expected syntax of AWP commands

The space after "`<!--`" and the space before "`-->`" in the formulation of an AWP command are essential to proper compiling of the command. Omission of the space characters can cause the compiler to be unable to generate the proper code. The compiler does not display an error in this case.

AWP command summary

The details for using each AWP command are in the topics to follow, but here is a brief summary of the commands:

Reading variables

```
:=<Varname>:
```

Writing variables

```
<!-- AWP_In_Variable Name='<Varname1>' [Use='<Varname2>'] ... -->
```

This AWP command merely declares the variable in the Name clause to be writable. Your HTML code performs writes to the variable by name from <input>, <select>, or other HTML statements within an HTML form.

Reading special variables

```
<!-- AWP_Out_Variable Name='<Type>:<Name>' [Use='<Varname>'] -->
```

Writing special variables

```
<!-- AWP_In_Variable Name='<Type>:<Name>' [Use='<Varname>']-->
```

Defining enum types

```
<!--
 AWP_Enum_Def Name='<Enum type name>' Values='<Value>, <Value>,... '
 -->
```

Referencing enum types

```
<!-- AWP_In_Variable Name='<Varname>' Enum="<Enum type name>" -->
<!-- AWP_Out_Variable Name='<Varname>' Enum="<Enum type name>" -->
```

Creating fragments

```
<!-- AWP_Start_Fragment Name='<Name>' [Type=<Type>][ID=<id>] -->
```

Importing fragments

```
<!-- AWP_Import_Fragment Name='<Name>' -->
```

12.7.2.1 Reading variables

User-defined Web pages can read variables (PLC tags) and data block tags from the CPU, provided that you have configured the tags to be accessible from an HMI .

Syntax

```
:=<Varname>:
```

Parameters

<Varname>	The variable to be read, which can be a PLC tag name from your STEP 7 program, a data block tag, I/O, or addressable memory. For memory or I/O addresses or alias names (Page 828), do not use quotation marks around the tag name. For PLC tags, use double quotation marks around the tag name. For data block tags, enclose the block name only in double quotation marks. The tag name is outside of the quotation marks. Note that you use the data block name and not a data block number.

Examples

```
:="Conveyor_speed"::="My_Data_Block".flag1:
:=I0.0:
:=MW100:
```

Example reading an aliased variable

```
<!-- AWP_Out_Variable Name='flag1' Use='"My_Data_Block".flag1' -->
:=flag1:
```

Note

Defining alias names for PLC tags and data block tags is described in the topic Using an alias for a variable reference (Page 823).

If a tag name or data block name includes special characters, you must use additional quotation marks or escape characters as described in the topic Handling tag names that contain special characters (Page 828).

12.7.2.2 Writing variables

User-defined pages can write data to the CPU. This is accomplished by using an AWP command to identify a variable in the CPU to be writable from the HTML page. The variable must be specified by PLC tag name or data block tag name. You can declare multiple variable names in one statement. To write the data to the CPU, you use standard HTTP POST commands.

A typical usage is to design a form in your HTML page with text input fields or select list choices that correspond to writable CPU variables. As with all user-defined pages, you then generate the blocks from STEP 7 such that they are included in your STEP 7 program. When a user with privileges to modify variables subsequently accesses this page and types data into the input fields or selects a choice from a select list, the Web server converts the input to the appropriate data type for the variable, and writes the value to the variable in the CPU. Note that the name clause for HTML input fields and HTML select lists uses syntax typical for the name clause of the AWP_In_Variable command. Typically enclose the name in single quotation marks and if you reference a data block, enclose the data block name in double quotation marks.

For form management details, refer to documentation for HTML.

Syntax

```
<!-- AWP_In_Variable Name='<Varname1>' [Use='<Varname2>'] ... -->
```

Parameters

<Varname1>	If no Use clause is provided, Varname1 is the variable to be written. It can be a PLC tag name from your STEP 7 program or a tag from a specific data block. If a Use clause is provided, Varname1 is an alternate name for the variable referenced in <Varname2> (Page 823). It is a local name within the HTML page.
<Varname2>	If a Use clause is provided, Varname2 is the variable to be written. It can be a PLC tag name from your STEP 7 program or a tag from a specific data block.

For both Name clauses and Use clauses, the complete name must be enclosed in single quotation marks. Within the single quotes, use double quotation marks around a PLC tag and double quotation marks around a data block name. The data block name is within the double quotes but not the data block tag name. Note that for data block tags, you use the name of the block and not a data block number.

Examples using HTML input field

```
<!-- AWP_In_Variable Name='"Target_Level"' -->
<form method="post">
<p>Input Target Level: <input name='"Target_Level"' type="text" />
</p>
</form>

<!-- AWP_In_Variable Name='"Data_block_1".Braking' -->
<form method="post">
<p>Braking: <input name='"Data_block_1".Braking' type="text" />
%</p>
</form>

<!-- AWP_In_Variable Name='"Braking"' Use='"Data_block_1".Braking' -
->
<form method="post">
<p>Braking: <input name='"Braking"' type="text" /> %</p>
</form>
```

Example using HTML select list

```
<!-- AWP_In_Variable Name='"Data_block_1".ManualOverrideEnable'-->
<form method="post">
<select name='"Data_block_1".ManualOverrideEnable'>
<option value=:"Data_block_1".ManualOverrideEnable:> </option>
<option value=1>Yes</option>
<option value=0>No</option>
</select><input type="submit" value="Submit setting" /></form>
```

Note

Only a user with privileges to modify variables can write data to the CPU. The Web server ignores the commands if the user does not have modify privileges.

If a tag name or data block name includes special characters, you must use additional quotation marks or escape characters as described in the topic "Handling tag names that contain special characters (Page 828)".

12.7.2.3 Reading special variables

The Web server provides the ability to read values from the PLC to store in special variables in the HTTP response header. You might, for example, want to read a pathname from a PLC tag to redirect the URL to another location using the HEADER:Location special variable.

Syntax

```
<!-- AWP_Out_Variable Name='<Type>:<Name>' [Use='<Varname>'] -->
```

Parameters

<Type>	The type of special variable, which is one of the following: HEADER COOKIE_VALUE COOKIE_EXPIRES
<Name>	Refer to HTTP documentation for a list of all the names of HEADER variables. A few examples are listed below: Status: response code Location: path for redirection Retry-After: how long service is expected to be unavailable to the requesting client For types COOKIE_VALUE and COOKIE_EXPIRES, <Name> is the name of a specific cookie. COOKIE_VALUE:name: value of the named cookie COOKIE_EXPIRES:name: expiration time in seconds of named cookie The Name clause must be enclosed in single or double quotation marks. If no Use clause is specified, the special variable name corresponds to a PLC tag name. Enclose the complete Name clause within single quotation marks and the PLC tag in double quotation marks. The special variable name and PLC tag name must match exactly.
<Varname>	Name of the PLC tag or data block tag into which the variable is to be read The Varname must be enclosed in single quotation marks. Within the single quotes, use double quotation marks around a PLC tag or data block name. The data block name is within the double quotes but not the data block tag name. Note that for data block tags, you use the name of the block and not a data block number.

If a tag name or data block name includes special characters, you must use additional quotation marks or escape characters as described in the topic Handling tag names that contain special characters (Page 828).

Example: Reading a special variable with no Use clause

```
<!-- AWP_Out_Variable Name='"HEADER:Status"' -->
```

In this example, the HTTP special variable "HEADER:Status" receives the value of the PLC tag "HEADER:Status". The name in the PLC tag table must match the name of the special variable exactly if no Use clause is specified.

Example: Reading a special variable with a Use clause

```
<!-- AWP_Out_Variable Name='HEADER:Status' Use='"Status"' -->
```

In this example, the HTTP special variable "HEADER:Status" receives the value of the PLC tag "Status".

12.7.2.4 Writing special variables

The Web server provides the ability to write values to the CPU from special variables in the HTTP request header. For example, you can store information in STEP 7 about the cookie associated with a user-defined Web page, the user that is accessing a page, or header information. The Web server provides access to specific special variables that you can write to the CPU when logged in as a user with privileges to modify variables.

Syntax

```
<!-- AWP_In_Variable Name='<Type>:<Name>' [Use='<Varname>']-->
```

Parameters

<Type>	The type of special variable and is one of the following: HEADER SERVER COOKIE_VALUE
<Name>	Specific variable within the types defined above, as shown in these examples: HEADER:Accept: content types that are acceptable HEADER:User-Agent: information about the user agent originating the request. SERVER:current_user_id: id of the current user; 0 if no user logged in SERVER:current_user_name: name of the current user COOKIE_VALUE:<name>: value of the named cookie Enclose the Name clause in single quotation marks. If no Use clause is specified, the special variable name corresponds to a PLC variable name. Enclose the complete Name clause within single quotation marks and the PLC tag in double quotation marks. The special variable name must match the PLC tag name exactly. Refer to HTTP documentation for a list of all the names of HEADER variables.
<Varname>	The variable name in your STEP 7 program into which you want to write the special variable, which can be a PLC tag name, or a data block tag. The Varname must be enclosed in single quotation marks. Within the single quotes, use double quotation marks around a PLC tag or data block name. The data block name is within the double quotes but not the data block tag name. Note that for data block tags, you use the name of the block and not a data block number.

Examples

```
<!-- AWP_In_Variable Name='"SERVER:current_user_id"' -->
```

In this example, the Web page writes the value of the HTTP special variable "SERVER:current_user_id" to the PLC tag named "SERVER:current_user_id ".

```
<!-- AWP_In_Variable Name=SERVER:current_user_id' Use='"my_userid"'
-->
```

In this example, the Web page writes the value of the HTTP special variable "SERVER:current_user_id" to the PLC tag named "my_userid".

Note

Only a user with privileges to modify variables can write data to the CPU. The Web server ignores the commands if the user does not have modify privileges.

If a tag name or data block name includes special characters, you must use additional quotation marks or escape characters as described in the topic "Handling tag names that contain special characters (Page 828)".

12.7.2.5 Using an alias for a variable reference

You can use an alias in your user-defined Web page for an In_Variable or an Out_Variable. For example, you can use a different symbolic name in your HTML page than the one used in the CPU, or you can equate a variable in the CPU with a special variable. The AWP Use clause provides this capability.

Syntax

```
<-- AWP_In_Variable Name='<Varname1>' Use='<Varname2>' -->
<-- AWP_Out_Variable Name='<Varname1>' Use='<Varname2>' -->
```

Parameters

<Varname1>	The alias name or special variable name Varname1 must be enclosed in single or double quotation marks.
<Varname2>	Name of the PLC variable for which you want to assign an alias name. The variable can be a PLC tag, a data block tag, or a special variable. Varname2 must be enclosed in single quotation marks. Within the single quotes, use double quotation marks around a PLC tag, special variable, or data block name. The data block name is within the double quotes but not the data block tag name. Note that for data block tags, you use the name of the block and not a data block number.

Examples

```
<-- AWP_In_Variable Name='SERVER:current_user_id'
Use='"Data_Block_10".server_user' -->
```

In this example, the special variable SERVER:current_user_id is written to the tag "server_user" in data block "Data_Block_10".

```
<-- AWP_Out_Variable Name='Weight'
Use='"Data_Block_10".Tank_data.Weight' -->
```

In this example, the value in data block structure member Data_Block_10.Tank_data.Weight can be referenced simply by "Weight" throughout the rest of the user-defined Web page.

```
<-- AWP_Out_Variable Name='Weight' Use='"Raw_Milk_Tank_Weight"' -->
```

In this example, the value in the PLC tag "Raw_Milk_Tank_Weight" can be referenced simply by "Weight" throughout the rest of the user-defined Web page.

If a tag name or data block name includes special characters, you must use additional quotation marks or escape characters as described in the topic Handling tag names that contain special characters (Page 828).

12.7.2.6 Defining enum types

You can define enum types in your user-defined pages and assign the elements in an AWP command.

Syntax

```
<!-- AWP_Enum_Def Name='<Enum type name>' Values='<Value>,
<Value>,... ' -->
```

Parameters

<Enum type name>	Name of the enumerated type, enclosed in single or double quotation marks.
<Value>	<constant>:<name> The constant indicates the numerical value for the enum type assignment. The total number is unbounded. The name is the value assigned to the enum element.

Note that the entire string of enum value assignments is enclosed in single quotation marks, and each individual enum type element assignment is enclosed in double quotation marks. The scope of an enum type definition is global for the user-defined Web pages. If you have set up your user-defined Web pages in language folders (Page 847), the enum type definition is global for all pages in the language folder.

Example

```
<!-- AWP_Enum_Def Name='AlarmEnum' Values='0:"No alarms", 1:"Tank is
full", 2:"Tank is empty"' -->
```

12.7.2.7 Referencing CPU variables with an enum type

You can assign a variable in the CPU to an enum type. This variable can be used elsewhere in your user-defined Web page in a read operation (Page 817) or a write operation (Page 818). On a read operation, the Web server will replace the numerical value that is read from the CPU with the corresponding enum text value. On a write operation, the Web server will replace the text value with the integer value of the enumeration that corresponds to the text before writing the value to the CPU.

Syntax

```
<!-- AWP_In_Variable Name='<Varname>' Enum="<EnumType>" -->
<!-- AWP_Out_Variable Name='<Varname>' Enum="<EnumType>" -->
```

Parameters

<Varname>	Name of PLC tag or data block tag to associate with the enum type, or the name of the alias name for a PLC tag (Page 823) if declared. Varname must be enclosed in single quotation marks. Within the single quotes, use double quotation marks around a PLC tag or data block name. Note that for data block tags, you use the name of the block and not a data block number. The data block name is within the double quotes but not the data block tag name.
<EnumType>	Name of the enumerated type, which must be enclosed in single or double quota-tion marks

The scope of an enum type reference is the current fragment.

Example usage in a variable read

```
<!-- AWP_Out_Variable Name='"Alarm"' Enum="AlarmEnum" -->...
<p>The current value of "Alarm" is :="Alarm":</p>
```

If the value of "Alarm" in the CPU is 2, the HTML page displays 'The current value of "Alarm" is Tank is empty' because the enum type definition (Page 823) assigns the text string "Tank is empty" to the numerical value 2.

Example usage in a variable write

```
<!-- AWP_Enum_Def Name='AlarmEnum' Values='0:"No alarms", 1:"Tank is
full", 2:"Tank is empty"' -->
<!-- AWP_In_Variable Name='"Alarm"' Enum='AlarmEnum' -->...
<form method="POST">
<p><input type="hidden" name='"Alarm"' value="Tank is full" /></p>
<p><input type="submit" value='Set Tank is full' /><p>
</form>
```

Because the enum type definition (Page 823) assigns "Tank is full" to the numerical value 1, the value 1 is written to the PLC tag named "Alarm" in the CPU.

Note that the Enum clause in the AWP_In_Variable declaration must correspond exactly to the Name clause in the AWP_Enum_Def declaration.

Example usage in a variable write with use of an alias

```
<!-- AWP_Enum_Def Name='AlarmEnum' Values='0:"No alarms", 1:"Tank is
full", 2:"Tank is empty"' -->
<!-- AWP_In_Variable Name='"Alarm"' Enum='AlarmEnum'
Use='"Data_block_4".Motor1.Alarm'-->...
<form method="POST">
<p><input type="hidden" name='"Alarm"' value="Tank is full" /></p>
<p><input type="submit" value='Set Tank is full' /><p>
</form>
```

Because the enum type definition (Page 823) assigns "Tank is full" to the numerical value 1, the value 1 is written to the alias "Alarm" which corresponds to the PLC tag named "Motor1.Alarm" in data block "Data_Block_4" in the CPU.

If a tag name or data block name includes special characters, you must use additional quotation marks or escape characters as described in the topic Handling tag names that contain special characters (Page 828).

Note

Previous releases required a separate AWP_Enum_Ref declaration to associate a variable with a defined enum type. STEP 7 and the S7-1200 support existing code with AWP_Enum_Ref declarations; however, this command is no longer needed.

12.7.2.8 Creating fragments

STEP 7 converts and stores user-defined Web pages as a control DB and fragment DBs when you click "Generate blocks" in the CPU Properties for the Web server. You can set up specific fragments for specific pages or for sections of specific pages. You can identify these fragments by a name and number with the "Start_Fragment" AWP command. Everything in the page following the AWP_Start_Fragment command belongs to that fragment until another AWP_Start_Command is issued or until end of file is reached.

Syntax

```
<!-- AWP_Start_Fragment Name='<Name>'
[Type=<Type>][ID=<id>][Mode=<Mode>] -->
```

Parameters

<Name>	Text string: name of fragment DB Fragment names must begin with a letter or underscore and be comprised of letters, numeric digits, and underscores. The fragment name is a regular expression of the form: [a-zA-Z_][a-zA-Z_0-9]*
<Type>	"manual" or "automatic" manual: The STEP 7 program must request this fragment and can respond accordingly. Operation of the fragment must be controlled with STEP 7 and the control DB variables. automatic: The Web server processes the fragment automatically. If you do not specify the type parameter, the default is "automatic".
<id>	Integer identification number. If you do not specify the ID parameter, the Web server assigns a number by default. For manual fragments, set the ID to a low number. The ID is the means by which the STEP 7 program controls a manual fragment.
<Mode>	"visible" or "hidden" visible: Contents of the fragment will display on the user-defined Web page. hidden: Contents of the fragment will not display on the user-defined Web page. If you do not specify the type parameter, the default is "visible".

Manual fragments

If you create a manual fragment for a user-defined Web page or portion of a page, then your STEP 7 program must control when the fragment is sent. The STEP 7 program must set appropriate parameters in the control DB for a user-defined page under manual control and then call the WWW instruction with the control DB as modified. For understanding the structure of the control DB and how to manipulate individual pages and fragments, see the topic Advanced user-defined Web page control (Page 851).

12.7.2.9 Importing fragments

You can create a named fragment from a portion of your HTML code and then import that fragment elsewhere in your set of user-defined Web pages. For example, consider a set of user-defined Web pages that has a start page and then several other HTML pages accessible from links on the start page. Suppose each of the separate pages is to display the company logo on the page. You could implement this by creating a fragment (Page 826) that loads the image of the company logo. Each individual HTML page could then import this fragment to display the company logo. You use the AWP Import_Fragment command for this purpose. The HTML code for the fragment only exists in one fragment, but you can import this fragment DB as many times as necessary in as many Web pages as you choose.

Syntax

```
<!-- AWP_Import_Fragment Name='<Name>' -->
```

Parameters

<Name>	Text string: name of the fragment DB to be imported

Example

Excerpt from HTML code that creates a fragment to display an image:

```
<!-- AWP_Start_Fragment Name='My_company_logo' --><p><img
src="company_logo.jpg"></p>
```

Excerpt from HTML code in another .html file that imports the fragment that displays the logo image:

```
<!-- AWP_Import_Fragment Name='My_company_logo' -->
```

Both .html files (the one that creates the fragment and the one that imports it) are in the folder structure that you define when you configure the user-defined pages in STEP 7 (Page 830).

12.7.2.10 Combining definitions

When declaring variables for use in your user-defined Web pages, you can combine a variable declaration and an alias for the variable (Page 823). You can also declare multiple In_Variables in one statement and multiple Out_Variables in one statement.

Examples

```
<!-- AWP_In_Variable Name='"Level'", Name='"Weight"', Name='"Temp"'
-->
<--! AWP_Out_Variable Name='HEADER:Status', Use='"Status"',
         Name='HEADER:Location', Use="Location",
         Name='COOKIE_VALUE:name', Use="my_cookie" -->
<!-- AWP_In_Variable Name='Alarm' Use='"Data_block_10".Alarm' -->
```

12.7.2.11 Handling tag names that contain special characters

When specifying variable names in user-defined Web pages, you must take special care if tag names contain characters that have special meanings.

Reading variables

You use the following syntax to read a variable (Page 817):
`:=<Varname>:`

The following rules apply to reading variables:

- For variable names from the PLC tag table, enclose the tag name in double quotation marks.
- For variable names that are data block tags, enclose the data block name in double quotation marks. The tag is outside of the quotation marks.
- For variable names that are direct I/O addresses, memory addresses, or alias names, do not use quotation marks around the read variable.
- For tag names or data block tag names that contain a backslash, precede the backslash with another backslash.
- If a tag name or data block tag name contains a colon, less than sign, greater than sign, or ampersand define an alias that has no special characters for the read variable, and read the variable using the alias. Precede colons in tag names in a Use clause with a backslash.

Table 12- 1 Examples of Read variables

Data block name	Tag name	Read command

n/a	ABC:DEF	`<!--AWP_Out_Variable Name='special_tag' Use ='"ABC:DEF"' -->`
		`:=special_tag:`

n/a	T\	`:="T\\":`

n/a	A \B 'C :D	`<!--AWP_Out_Variable Name='another_special_tag' Use='"A \\B \'C :D"' -->`
		`:=another_special_tag:`

n/a	a<b	`<!--AWP_Out_Variable Name='a_less_than_b' Use='"a<b"' -->`
		`:=a_less_than_b:`

Data_block_1	Tag_1	`:="Data_block_1".Tag_1:`

Data_block_1	ABC:DEF	`<!-- AWP_Out_Variable Name='special_tag' Use='"Data_block_1".ABC\:DEF'-->`
		`:=special_tag:`

DB A' B C D$ E	Tag	`:="DB A' B C D$ E".Tag:`

DB:DB	Tag:Tag	`<!--AWP_Out_Variable Name='my_tag' Use ='"DB:DB".Tag\:Tag' -->`
		`:=my_tag:`

Name and Use clauses

The AWP commands AWP_In_Variable, AWP_Out_Variable, AWP_Enum_Def, AWP_Enum_Ref, AWP_Start_Fragment and AWP_Import_Fragment have Name clauses. HTML form commands such as <input> and <select> also have name clauses. AWP_In_Variable and AWP_Out_Variable can additionally have Use clauses. Regardless of the command, the syntax for Name and Use clauses regarding the handling of special characters is the same:

- The text you provide for a Name or Use clause must be enclosed within single quotation marks. If the enclosed name is a PLC tag or Data block name, use single quotation marks for the full clause.
- Within a Name or Use clause, data block names and PLC tag names must be enclosed within double quotation marks.
- If a tag name or Data block name includes a single quote character or backslash, escape that character with a backslash. The backslash is the escape character in the AWP command compiler.

Table 12- 2 Examples of Name clauses

Data block name	Tag name	Name clause options
n/a	ABC'DEF	**`Name='"ABC\'DEF"'`**
n/a	A \B 'C :D	**`Name='"A \\B \'C :D"'`**
Data_block_1	Tag_1	**`Name='"Data_block_1".Tag_1'`**
Data_block_1	ABC'DEF	**`Name='"Data_block_1".ABC\'DEF'`**
Data_block_1	A \B 'C :D	**`Name='"Data_block_1".A \\B \'C :D'`**
DB A' B C D$ E	Tag	**`Name='"DB A\' B C D$ E".Tag'`**

Use clauses follow the same conventions as Name clauses.

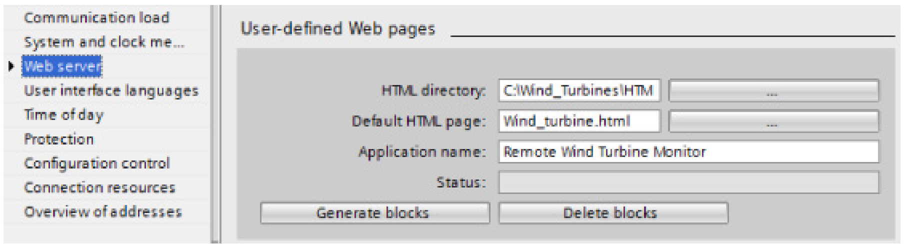
Note

Regardless of what characters you use in your HTML page, set the charset of the HTML page to UTF-8 and save it from the editor with UTF-8 character encoding.

12.7.3 Configuring use of user-defined Web pages

To configure user-defined Web pages from STEP 7, follow these steps:

1. Select the CPU in the Device Configuration view.
2. Display the "Web server" properties in the inspector window for the CPU.



HTML directory: C:\Wind_Turbines\HTM
Default HTML page: Wind_turbine.html
Application name: Remote Wind Turbine Monitor
Status:

3. If not already selected, select the check box for "Activate Web server on this module".
4. Select "Permit access only with HTTPS" to ensure that the Web server uses encrypted communication and to increase the security of your Web-accessible CPU.
5. Enter or browse to the folder name on your PC where you saved the HTML default page (start page).
6. Enter the name of the default page.
7. Provide a name for your application (optional). The Web server uses the application name to further subcategorize or group web pages. When you provide an application name, the Web server creates an URL for your user-defined page in the following format: http[s]://ww.xx.yy.zz/awp/<application name>/<pagename>.html.

   Avoid special characters in the application name. Some characters can cause the Web server to be unable to display the user-defined pages.
8. Enter filename extensions of files that include AWP commands. By default, STEP 7 analyzes files with .htm, .html, or .js extensions. If you have additional file extensions, append them.
9. Keep the default for the Web DB number, or enter a number of your choice. This is the DB number of the control DB that controls display of the Web pages.
10. Keep the default for the fragment DB start number, or enter a number of your choice. This is the first of the fragment DBs that contains the Web pages.

Generating program blocks

When you click the "Generate blocks" button, STEP 7 generates data blocks from the HTML pages in the HTML source directory that you specified and a control data block for the operation of your Web pages. You can set these attributes as needed for your application (Page 831). STEP 7 also generates a set of fragment data blocks to hold the representation of all of your HTML pages. When you generate the data blocks, STEP 7 updates the properties to display the control data block number, and the number of the first of the fragment data blocks. After you generate the data blocks, your user-defined Web pages are a part of your STEP 7 program. The blocks corresponding to these pages appear in the Web server folder, which is in the System blocks folder under Program blocks in the project navigation tree.

Deleting program blocks

To delete data blocks that you have previously generated, click the "Delete data blocks" button. STEP 7 deletes the control data block and all of the fragment data blocks from your project that correspond to user-defined Web pages.

12.7.4 Programming the WWW instruction for user-defined web pages

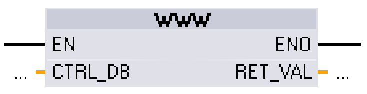
Your STEP 7 user program must include and execute the WWW instruction in order for the user-defined Web pages to be accessible from the standard Web pages. The control data block is the input parameter to the WWW instruction and specifies the content of the pages as represented in the fragment data blocks, as well as state and control information. STEP 7 creates the control data block when you click the "Create blocks" button in the configuration of user-defined Web pages (Page 830).

Programming the WWW instruction

The STEP 7 program must execute the WWW instruction for the user-defined Web pages to be accessible from the standard Web pages. You might want the user-defined Web pages available only under certain circumstances as dictated by your application requirements and preferences. In this case, your program logic can control when to call the WWW instruction.

Table 12- 3 WWW instruction

LAD / FBD	SCL	Description

	`ret_val := WWW(`
	`    ctrl_db:=_uint_in_);`
		Provides access to user-defined Web pages from standard Web pages

You must provide the control data block input parameter (CTRL_DB) which corresponds to the integer DB number of the control DB. You can find this control DB block number (called Web DB Number) in the Web Server properties of the CPU after you create the blocks for the user-defined Web pages. Enter the integer DB number as the CTRL_DB parameter of the WWW instruction. The return value (RET_VAL) contains the function result. Note that the WWW instruction executes asynchronously and that the RET_VAL output might have an initial value of 0 although an error can occur later. The program can check the state of the control DB to ensure that the application started successfully, or check RET_VAL with a subsequent call to WWW.

Table 12- 4 Return value

RET_VAL	Description
0	No error

16#00yx	x: The request represented by the respective bit is in the waiting state:
	x=1: request 0
	x=2: request 1
	x=4: request 2
	x=8: request 3
	The x values can be logically OR-ed to represent waiting states of multiple requests. If x = 6, for example, requests 1 and 2 are waiting.
	y: 0: no error; 1: error exists and "last_error" has been set in the control DB (Page 851)

16#803a	The control DB is not loaded.
16#8081	The control DB is of the wrong type, format, or version.
16#80C1	No resources are available to initialize the web application.

Usage of the Control DB

STEP 7 creates the control data block when you click "Generate blocks" and displays the control DB number in the User-defined Web pages properties. You can find the control DB as well in the Program blocks folder in the project navigation tree.

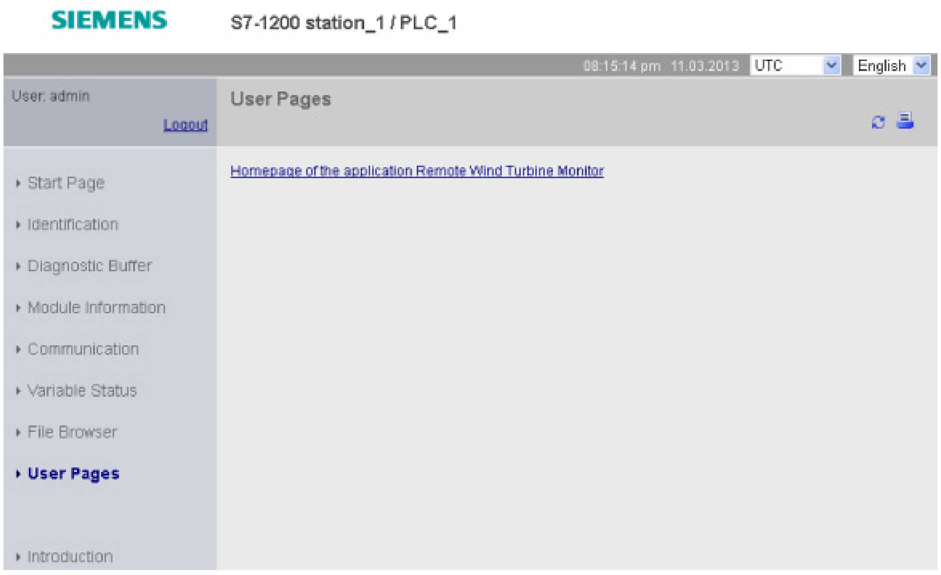
Typically, your STEP 7 program uses the control DB directly as created by the "Generate blocks" process with no additional manipulation. However, the STEP 7 user program can set global commands in the control DB to deactivate the web server or to subsequently re-enable it. Also, for user-defined pages that you create as manual fragment DBs (Page 830), the STEP 7 user program must control the behavior of these pages through a request table in the control DB. For information on these advanced tasks, see the topic Advanced user-defined Web page control (Page 851).

12.7.5 Downloading the program blocks to the CPU

After you have generated the blocks for user-defined Web pages, they are part of your STEP 7 program just like any other program blocks. You follow the normal process to download the program blocks to the CPU. Note that you can only download user-defined Web page program blocks when the CPU is in STOP mode.

12.7.6 Accessing the user-defined Web pages

You access your user-defined Web pages from the standard Web pages (Page 791). The standard Web pages display a link for "User Pages" on the left side menu where the links to the other pages appear. The mobile device navigation page also provides a link to "User Pages". When you click the "User Pages" link, your Web browser goes to the page that provides a link to your default page. From within the user-defined pages, navigation is according to how you designed your specific pages.



Homepage of the application Remote Wind Turbine Monitor

12.7.7 Constraints specific to user-defined Web pages

The constraints for standard Web pages (Page 855) also apply to user-defined Web pages. In addition, user-defined Web pages have some specific considerations.

Load memory space

Your user-defined Web pages become data blocks when you click "Generate blocks", which require load memory space. If you have a memory card installed, you have up to the capacity of your memory card as external load memory space for the user-defined Web pages.

If you do not have a memory card installed, these blocks take up internal load memory space, which is limited according to your CPU model.

You can check the amount of load memory space that is used and the amount that is available from the Online and Diagnostic tools in STEP 7. You can also look at the properties for the individual blocks that STEP 7 generates from your user-defined Web pages and see the load memory consumption.

Note

If you need to reduce the space required for your user-defined Web pages, reduce your use of images if applicable.

Quotation marks in text strings

Avoid using text strings that contain embedded single or double quotation marks in data block tags that you use for any purpose in user-defined Web pages. Because HTML syntax often uses single quotes or double quotes as delimiters, quotation marks within text strings can break the display of user-defined Web pages.

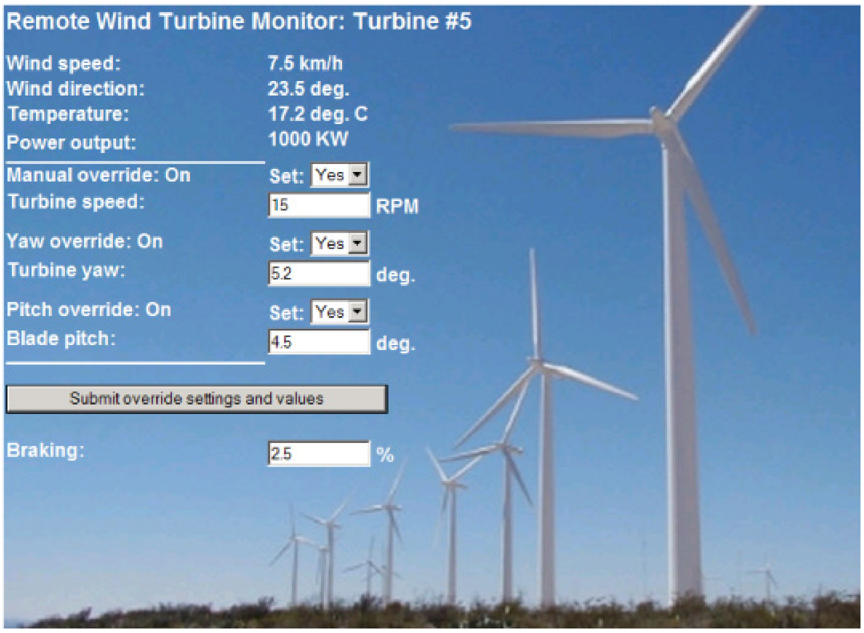
For data block tags of type String that you use in user-defined Web pages, observe the following rules:

- Do not enter single or double quotation marks in the data block tag string value in STEP 7.
- Do not let the user program make assignments of strings containing quotes to these data block tags.

12.7.8 Example of a user-defined web page

12.7.8.1 Web page for monitoring and controlling a wind turbine

As an example of a user-defined Web page, consider a Web page that is used to remotely monitor and control a wind turbine:



Remote Wind Turbine Monitor: Turbine #5

Wind speed:	7.5 km/h	
Wind direction:	23.5 deg.	
Temperature:	17.2 deg. C	
Power output:	1000 KW	
Manual override: On	Set: Yes	
Turbine speed:	15	RPM
Yaw override: On	Set: Yes	
Turbine yaw:	5.2	deg.
Pitch override: On	Set: Yes	
Blade pitch:	4.5	deg.
Braking:	2.5	%

Description

In this application, each wind turbine in a wind turbine farm is equipped with an S7-1200 for control of the turbine. Within the STEP 7 program, each wind turbine has a data block with data specific to that wind turbine.

The user-defined Web page provides remote turbine access from a PC. A user can connect to standard web pages of the CPU of a particular wind turbine and access the user-defined "Remote Wind Turbine Monitor" Web page to see the data for that turbine. A user with privileges to modify variables can also put the turbine in manual mode and control the variables for turbine speed, yaw, and pitch from the Web page. A user with privileges to modify variables can also set a braking value regardless of whether the turbine is under manual or automatic control.

The STEP 7 program would check the Boolean values for overriding automatic control, and if set, would use the user-entered values for turbine speed, yaw, and pitch. Otherwise, the program would ignore these values.

Files used

This user-defined Web page example consists of three files:

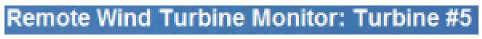
- **Wind_turbine.html**: This is the HTML page that implements the display shown above, using AWP commands to access controller data.
- **Wind_turbine.css**: This is the cascading style sheet that contains formatting styles for the HTML page. Use of a cascading style sheet is optional, but it can simplify the HTML page development.
- **Wind_turbine.jpg**: This is the background image that the HTML page uses. Use of images in user-defined Web pages is, of course, optional, and does require additional space in the CPU.

These files are not provided with your installation, but are described as an example.

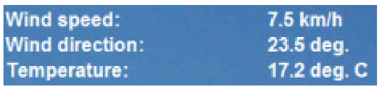
Implementation

The HTML page uses AWP commands to read values from the PLC (Page 817) for the display fields and to write values to the PLC (Page 818) for data coming from user input. This page also uses AWP commands for enum type definition (Page 823) and reference (Page 824) for handling ON/OFF settings.

The first part of the page displays a header line that includes the wind turbine number.

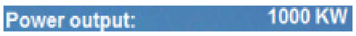

Remote Wind Turbine Monitor: Turbine #5

The next part of the page displays atmospheric conditions at the wind turbine. I/O at the turbine site provide the wind speed, wind direction, and current temperature.



Wind speed: 7.5 km/h
Wind direction: 23.5 deg.
Temperature: 17.2 deg. C

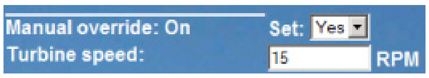
Next, the page displays the power output of the turbine as read from the S7-1200.



Power output: 1000 KW

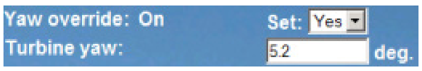
The following sections allow for manual control of the turbine, overriding the normal automatic control by the S7-1200. These types are as follows:

- Manual override: enables manual override of the turbine. The STEP 7 user program requires that the manual override setting be true before enabling the use of any of the manual settings for turbine speed, or yaw or pitch.



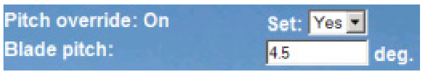
Manual override: On Set: Yes
Turbine speed: 15 RPM

- Yaw override: enables manual override of the yaw setting, and a manual setting for the yaw. The STEP 7 user program requires that both manual override and yaw override be true in order to apply the yaw setting.



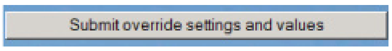
Yaw override: On Set: Yes
Turbine yaw: 5.2 deg.

- Pitch override: enables manual override of the pitch of the blades. The STEP 7 user program requires that both manual override and pitch override be true in order to apply the blade pitch setting.

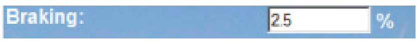


Pitch override: On Set: Yes
Blade pitch: 4.5 deg.

The HTML page includes a submit button to post the override settings to the controller.



The braking user input field provides a manual setting for a braking percentage. The STEP 7 user program does not require manual override to accept a braking value.

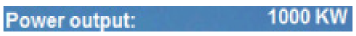


Braking: 2.5 %

In addition, the HTML page uses an AWP command to write the special variable (Page 821) that contains the user ID of the user that is accessing the page to a tag in the PLC tag table.

12.7.8.2 Reading and displaying controller data

The "Remote Wind Turbine Monitor" HTML page uses numerous AWP commands for reading data from the controller (Page 817) and displaying it on the page. For example, consider the HTML code for displaying the power output as shown in this portion of the example Web page:



Power output: 1000 KW

Example HTML code

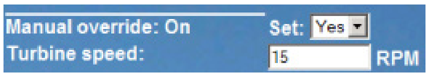
The following excerpt from the "Remote Wind Turbine Monitor" HTML page displays the text "Power Output:" in the left cell of a table row and reads the variable for the power output and displays it in the right cell of the table row along with the text abbreviation for kilowatts, kW.

The AWP command :="Data_block_1".PowerOutput: performs the read operation. Note that data blocks are referenced by name, not by data block number (that is, "Data_block_1" and not "DB1").

```
<tr style="height:2%;">
<td>
<p>Power output:</p>
</td>
<td>
<p style="margin-bottom:5px;"> :="Data_block_1".PowerOutput: kW</p>
</td>
</tr>
```

12.7.8.3 Using an enum type

The "Remote Wind Turbine Monitor" HTML page uses enum types for the three instances where HTML page displays "ON" or "OFF" for a Boolean value, and for where the user sets a Boolean value. The enum type for "ON" results in a value of 1, and the enum type for "OFF" results in a value of 0. For example, consider the HTML code for reading and writing the Manual Override Enable setting in "Data_block_1".ManualOverrideEnable value using an enum type:


Manual override: On
Set: Yes
Turbine speed: 15 RPM

Example HTML code

The following excerpts from the "Remote Wind Turbine Monitor" HTML page show how to declare an enum type called "OverrideStatus" with values of "Off" and "On" for 0 and 1, and then setting an enum type reference to OverrideStatus for the ManualOverrideEnable Boolean tag in the data block named "Data_block_1".

```
<!-- AWP_In_Variable Name='"Data_block_1".ManualOverrideEnable'
Enum="OverrideStatus" -->

<!-- AWP_Enum_Def Name="OverrideStatus" Values='0:"Off",1:"On"' -->
```

Where the HTML page includes a display field in a table cell for the current state of ManualOverrideEnable, it uses just a normal read variable command, but with the use of the previously declared and referenced enum type, the page displays "Off" or "On" rather than 0 or 1.

```
<td style="width:24%; border-top-style: Solid; border-top-width:
2px; border-top-color: #ffffff;">
<p>Manual override: :="Data_block_1".ManualOverrideEnable:</p>
</td>
```

The HTML page includes a drop-down select list for the user to change the value of ManualOverrideEnable. The select list uses the text "Yes" and "No" to display in the select lists. With the use of the enum type, "Yes" is correlated to the value "On" of the enum type, and "No" is correlated to the value "Off". The empty selection leaves the value of ManualOverrideEnable as it is.

```
<select name='"Data_block_1".ManualOverrideEnable'>
<option value=':"Data_block_1".ManualOverrideEnable:'> </option>
<option value="On">Yes</option>
<option selected value="Off">No</option>
</select>
```

The select list is included within a form on the HTML page. When the user clicks the submit button, the page posts the form, which writes a value of "1" to the Boolean ManualOverrideEnable in Data_block_1 if the user had selected "Yes", or "0" if the user had selected "No".

12.7.8.4 Writing user input to the controller

The "Remote Wind Turbine Monitor" HTML page includes several AWP commands for writing data to the controller (Page 818). The HTML page declares AWP_In_Variables for Boolean variables so that a user with privileges to modify variables can put the wind turbine under manual control and enable manual override for the turbine speed, yaw override, and/or blade pitch override. The page also uses AWP_In_Variables to allow a user with privileges to modify variables to subsequently set floating-point values for the turbine speed, yaw, pitch, and braking percentage. The page uses an HTTP form post command to write the AWP_In_Variables to the controller.

For example, consider the HTML code for manually setting the braking value:

Example HTML code

The following excerpt from the "Remote Wind Turbine Monitor" HTML page first declares an AWP_In_Variable for "Data_block_1" that enables the HTML page to write to any tags in the data block "Data_block_1". The page displays the text "Braking:" in the left cell of a table row. In the right cell of the table row is the field that accepts user input for the "Braking" tag of "Data_block_1". This user input value is within an HTML form that uses the HTTP method "POST" to post the entered text data to the CPU. The page then reads the actual braking value from the controller and displays it in the data entry field.

A user with privileges to modify variables can subsequently use this page to write a braking value to the data block in the CPU that controls braking.

```
<!-- AWP_In_Variable Name='"Data_block_1"' -->
...
<tr style="vertical-align: top; height: 2%;">
<td style="width: 22%;"><p>Braking:</p></td>
<td>
<form method="POST">
<p><input name='"Data_block_1".Braking' size="10" type="text"> %</p>
</form>
</td>
</tr>
```

Note

Note that if a user-defined page has a data entry field for a writable data block tag that is a string data type, the user must enclose the string in single quotation marks when entering the string value in the field.

Note

Note that if you declare an entire data block in an AWP_In_Variable declaration such as <!-- AWP_In_Variable Name='"Data_block_1"' -->, then every tag within that data block can be written from the user-defined Web page. Use this when you intend for all of the tags in a data block to be writable. Otherwise, if you only want specific data block tags to be writable from the user-defined Web page, declare it specifically with a declaration such as <!-- AWP_In_Variable Name='"Data_block_1".Braking' -->

12.7.8.5 Writing a special variable

The "Remote Wind Turbine Monitor" Web page writes the special variable SERVER:current_user_id to a PLC tag in the CPU, providing that the user has modify privileges. In this case, the PLC tag value contains the user ID of the user who is accessing the "Remote Wind Turbine Monitor" Web page.

The Web page writes the special variable to the PLC and requires no user interface.

Example HTML code

```
<!-- AWP_In_Variable Name="SERVER:current_user_id" Use="User_ID"-->
```

12.7.8.6 Reference: HTML listing of remote wind turbine monitor Web page

Wind_turbine.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<!--
This test program simulates a Web page to monitor and control a Wind
Turbine
Required PLC tags and Data Block Tags in STEP 7:

PLC Tag:
User_ID: Int

Data Blocks:
Data_block_1

Tags in Data_Block_1:

TurbineNumber: Int
WindSpeed: Real
WindDirection: Real
Temperature: Real
PowerOutput: Real
ManualOverrideEnable: Bool
TurbineSpeed: Real
YawOverride: Bool
Yaw: Real
PitchOverride: Bool
Pitch: Real
Braking: Real
The user-defined Web page displays current values for the PLC data,
and provides a select list to set the three Booleans using an
enumerated type assignment. The "Submit" button posts the selected
Boolean values as well as the data entry fields for TurbineSpeed,
Yaw, and Pitch. The value for Braking can be set without use of the
"Submit" button.

No actual STEP 7 program is required to use this page.
Theoretically, the STEP 7 program would only act on the values of
TurbineSpeed, Yaw, and Pitch, if the associated Booleans were set.
The only STEP 7 requirement is to call the WWW instruction with the
DB number of the generated data blocks for this page.
-->
<!-- AWP_In_Variable Name='"Data_block_1"' -->
<!-- AWP_In_Variable Name='"Data_block_1".ManualOverrideEnable'
Enum="OverrideStatus" -->
<!-- AWP_In_Variable Name='"Data_block_1".PitchOverride'
Enum="OverrideStatus" -->
<!-- AWP_In_Variable Name='"Data_block_1".YawOverride'
Enum="OverrideStatus" -->
<!-- AWP_In_Variable Name="SERVER:current_user_id" Use="User_ID"-->
<!-- AWP_Enum_Def Name="OverrideStatus" Values='0:"Off",1:"On"' -->

<html>
```

```
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-
8"><link rel="stylesheet" href="Wind_turbine.css">
<title>Remote Wind Turbine Monitor</title>
</head>
<body>
<table cellpadding="0" cellspacing="2">
<tr style="height: 2%;">
<td colspan="2">
<h2>Remote Wind Turbine Monitor: Turbine
#:="Data_block_1".TurbineNumber:</h2>
</td>

<tr style="height: 2%;"><td style="width: 25%;"><p>Wind
speed:</p></td>
<td><p> :="Data_block_1".WindSpeed: km/h</p></td>
</tr>

<tr style="height: 2%;">
<td style="width: 25%;"><p>Wind direction:</p></td>
<td><p> :="Data_block_1".WindDirection: deg.</p></td>
</tr>

<tr style="height: 2%;"><td style="width:
25%;"><p>Temperature:</p></td>
<td><p> :="Data_block_1".Temperature: deg. C</p></td>
</tr>

<tr style="height: 2%;">
<td style="width: 25%;"><p>Power output:</p></td>
<td><p style="margin-bottom:5px;"> :="Data_block_1".PowerOutput:
kW</p>
</td>
</tr>

<form method="POST" action="">
<tr style="height: 2%;" >
<td style="width=25%; border-top-style: Solid; border-top-width:
2px; border-top-color: #ffffff;">
<p>Manual override: :="Data_block_1".ManualOverrideEnable:</p>
</td>
<td class="Text">Set:

<select name='"Data_block_1".ManualOverrideEnable'>
<option value=':="Data_block_1".ManualOverrideEnable:'> </option>
<option value="On">Yes</option>
<option value="Off">No</option>
</select>

</td>
</tr>

<tr style="vertical-align: top; height: 2%;"><td style="width:
25%;"><p>Turbine speed:</p></td>
<td>
```

```
<p style="margin-bottom:5px;"><input
name='"Data_block_1".TurbineSpeed' size="10"
value=':="Data_block_1".TurbineSpeed:' type="text"> RPM</p>
</td>
</tr>

<tr style="vertical-align: top; height: 2%;">
<td style="width: 25%;">
<p>Yaw override: :="Data_block_1".YawOverride: </p>
</td>
<td class="Text">Set:

<select name='"Data_block_1".YawOverride'>
<option value=':="Data_block_1".YawOverride:'> </option>
<option value="On">Yes</option>
<option value="Off">No</option>
</select>

</td>
</tr>

<tr style="vertical-align: top; height: 2%;">
<td style="width: 25%;">
<p>Turbine yaw:</p>
</td>
<td>
<p style="margin-bottom:5px;"><input name='"Data_block_1".Yaw'
size="10" value=':="Data_block_1".Yaw:' type="text"> deg.</p>
</td>
</tr>

<tr style="vertical-align: top; height: 2%;">
<td style="width: 25%;">
<p>Pitch override: :="Data_block_1".PitchOverride: </p>
</td>
<td class="Text">Set:

<select name='"Data_block_1".PitchOverride'>
<option value=':="Data_block_1".PitchOverride:'> </option>
<option value="On">Yes</option>
<option value="Off">No</option>
</select>

</td>
</tr>

<tr style="vertical-align: top; height: 2%;">
<td style="width=25%; border-bottom-style: Solid; border-bottom-
width: 2px; border-bottom-color: #ffffff;">
<p>Blade pitch:</p>
</td>
<td>
<p style="margin-bottom:5px;"><input name='"Data_block_1".Pitch'
size="10" value=':="Data_block_1".Pitch:' type="text"> deg.</p>
</td>
```

```
</tr>
<tr style="height: 2%;">
<td colspan="2">
<input type="submit" value="Submit override settings and values">
</td>
</tr>
</form>

<tr style="vertical-align: top; height: 2%;">
<td style="width: 25%;"><p>Braking:</p></td>
<td>
<form method="POST" action="">
<p> <input name='"Data_block_1".Braking' size="10"
value=':="Data_block_1".Braking:' type="text"> %</p>
</form>
</td>
</tr>
<tr><td></td></tr>

</table>
</body>
</html>
```

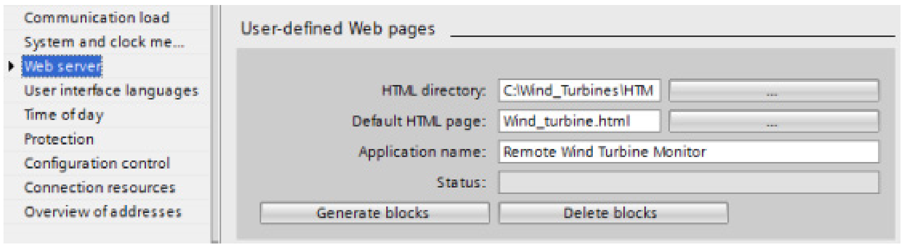
Wind_turbine.css

```
BODY {
 background-image: url('./Wind_turbine.jpg');
 background-position: 0% 0%;
 background-repeat: no-repeat;
 background-size: cover;
}
H2 {
 font-family: Arial;
 font-weight: bold;
 font-size: 14.0pt;
 color: #FFFFFF;
 margin-top:0px;
 margin-bottom:10px;
}
P {
 font-family: Arial;
 font-weight: bold;
 color: #FFFFFF;
 font-size: 12.0pt;
 margin-top:0px;
 margin-bottom:0px;
}
TD.Text {
 font-family: Arial;
 font-weight: bold;
 color: #FFFFFF;
 font-size: 12.0pt;
 margin-top:0px;
 margin-bottom:0px;
}
```

12.7.8.7 Configuration in STEP 7 of the example Web page

To include the "Remote Wind Turbine Monitor" HTML page as a user-defined Web page for the S7-1200, you configure the data about the HTML page in STEP 7 and create data blocks from the HTML page.

Access the CPU Properties for the S7-1200 that controls the wind turbine, and enter the configuration information in the User-defined web pages properties of the Web Server:



HTML directory: C:\Wind_Turbines\HTM
Default HTML page: Wind_turbine.html
Application name: Remote Wind Turbine Monitor
Status:

Configuration fields

- HTML directory: This field specifies the fully-qualified pathname to the folder where the default page (home page or start page) is located on the computer. The "..." button allows you to browse to the folder that you need.
- Default HTML page: This field specifies the filename of the default page or home page of the HTML application. The "..." button allows you to select the file that you need. For this example, WindTurbine.html is the default HTML page. The Remote Wind Turbine Monitor example only consists of a single page, but in other user-defined applications the default page can call up additional pages from links on the default page. Within the HTML code, the default page must reference other pages relative to the HTML source folder.
- Application name: This optional field contains the name that the Web browser includes in the address field when displaying the page. For this example, it is "Remote Wind Turbine Monitor", but you can use any name.

No other fields require configuration.

Final steps

To use the Remote Wind Turbine Monitor as configured, generate the blocks, program the WWW instruction (Page 831) with the number of the generated control DB as an input parameter, download the program blocks, and put the CPU in run mode.

When an operator subsequently accesses the standard Web pages for the S7-1200 that controls the wind turbine, the "Remote Wind Turbine Monitor" Web page is accessible from the "User Pages" link on the navigation bar. This page now provides the means to monitor and control the wind turbine.

12.7.9 Setting up user-defined Web pages in multiple languages

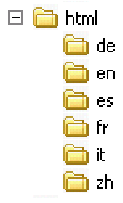
The Web server provides the means for you to provide user-defined Web pages in the following languages:

- German (de)
- English (en)
- Spanish (es)
- French (fr)
- Italian (it)
- Simplified Chinese (zh)

You do this by setting up your HTML pages in a folder structure (Page 847) that corresponds to the languages and by setting a specific cookie named "siemens_automation_language" from your pages (Page 847). The Web server responds to this cookie, and switches to the default page in the corresponding language folder.

12.7.9.1 Creating the folder structure

To provide user-defined Web pages in multiple languages, you set up a folder structure under your HTML directory. The two-letter folder names are specific and must be named as shown below:



de: German
en: English
es: Spanish
fr: French
it: Italian
zh: Simplified Chinese

At the same level, you can also include any other folders that your pages need, for example, folders for images or scripts.

You can include any subset of the language folders. You do not have to include all six languages. Within the language folders, you create and program your HTML pages in the appropriate language.

12.7.9.2 Programming the language switch

The Web server performs switching between languages through the use of a cookie named "siemens_automation_language". This is a cookie defined and set in the HTML pages, and interpreted by the Web server to display a page in the appropriate language from the language folder of the same name. The HTML page must include a JavaScript to set this cookie to one of the pre-defined language identifiers: "de", "en", "es", "fr", "it', or "zh".

For example, if the HTML page sets the cookie to "de", the Web server switches to the "de" folder and displays the page with the default HTML page name as defined in the STEP 7 configuration (Page 851).

Example

The following example uses a default HTML page named "langswitch.html" in each of the language folders. Also in the HTML directory is a folder named "script". The script folder includes a JavaScript file named "lang.js". Each langswitch.html page uses this JavaScript to set the language cookie, "siemens_automation_language".

HTML for "langswitch.html" in "en" folder

The header of the HTML page sets the language to English, sets the character set to UTF-8, and sets the path to the JavaScript file lang.js.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Language" content="en">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Language switching english page</title>
<script type="text/javascript" src="script/lang.js" ></script>
```

The body of the file uses a select list for the user to select between German and English. English ("en") is pre-selected for the language. When the user changes the language, the page calls the DoLocalLanguageChange() JavaScript function with the value of the selected option.

```
<!-- Language Selection -->
<table>
   <tr>
      <td align="right" valign="top" nowrap>
       <!-- change language immediately on selection change -->
         <select name="Language"
                   onchange="DoLocalLanguageChange(this)"
                   size="1">
           <option value="de" >German</option>
           <option value="en" selected >English</option>
          </select>
      </td>
   </tr>
</table><!-- Language Selection End-->
```

HTML for "langswitch.html" in "de" folder

The header for the German langswitch.html page is the same as English, except the language is set to German.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Language" content="de"><meta http-
equiv="Content-Type" content="text/html; charset=utf-8">
<title>Sprachumschaltung Deutsche Seite</title>
<script type="text/javascript" src="script/lang.js" ></script>
</head>
```

The HTML in the German page is identical to that of the English page, except that the default value of the selected language is German ("de").

```
<!-- Language Selection -->
<table>
   <tr>
     <td align="right" valign="top" nowrap>
       <!-- change language immediately on change of the selection -
->
       <select name="Language"
               onchange="DoLocalLanguageChange(this)"
               <size="1">
            <option value="de" selected >Deutsch</option>
            <option value="en" >Englisch</option>
       </select>
      </td>
   </tr>
</table><!-- Language Selection End-->
```

JavaScript "lang.js" in "script" folder

The function "DoLocalLanguageChange()" is in the lang.js file. This function calls the "SetLangCookie()" function and then reloads the window that is displaying the HTML page.

The function "SetLangCookie()" constructs an assignment that assigns the value from the select list to the "siemens_automation_language" cookie of the document. It also sets the path to the application so that the switched page, and not the requesting page, receives the value of the cookie.

Optionally, in the commented section, the page could set an expiration value for the cookie.

```
function DoLocalLanguageChange(oSelect) {
        SetLangCookie(oSelect.value);
        top.window.location.reload();
    }
function SetLangCookie(value) {
        var strval = "siemens_automation_language=";
        // This is the cookie by which the Web server
        // detects the desired language
        // This name is required by the Web server.
        strval = strval + value;
        strval = strval + "; path=/ ;";
        // Set path to the application, since otherwise
        // path would be set to the requesting page
        // and this page would not get the cookie.
        /* OPTIONAL
           use expiration if this cookie should live longer
           than the current browser session:
           var now = new Date();
           var endttime = new Date(now.getTime() + expiration);
           strval = strval + "; expires=" +
                    endttime.toGMTString() + ";";
        */
        document.cookie = strval;
    }
```

Note

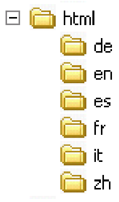
If your user-defined Web page implementation includes HTML files within language-specific folders (en, de, for example) and also HTML files that are not in the language-specific folders, note that you cannot define enum types with the AWP_Enum_Def command in files in both locations. If you use enums, you must define them either within files in the language - specific folders or within files outside of the language-specific folders. You cannot make enum declarations in files in both places.

12.7.9.3 Configuring STEP 7 to use a multi-language page structure

The procedure for configuring multi-language user-defined Web pages is similar to the general process for configuring user-defined Web pages (Page 830). When you have folders set up for languages, however, you set your HTML directory setting to the folder that contains the individual language folders. You do not set the HTML directory to be one of the language folders.

When you select the default HTML page, you navigate into the language folder and select the HTML page that is to be the start page. When you subsequently generate blocks and download the blocks to the CPU, the Web server displays the start page in the language folder that you configured.

For example, if the folder structure shown here was at C:\, the setting for HTML directory would be C:\html, and if English were to be the initial page display, you would navigate to en\langswitch.html for the default HTML page setting.



12.7.10 Advanced user-defined Web page control

When you generate data blocks for your user-defined Web pages, STEP 7 creates a control DB that it uses to control display of and interaction with the user-defined pages. STEP 7 also creates a set of fragment DBs that represent the individual pages. Under normal circumstances, you do not need to know the structure of the control DB or how to manipulate it.

If you want to turn a web application on and off, for example, or manipulate individual manual fragments, you use the control DB tags and the WWW instruction to do so.

Structure of the control DB

The control DB is an extensive data structure, and is accessible when programming your STEP 7 user program. Only some of the control data block tags are described here.

Commandstate structure

"Commandstate" is a structure that contains global commands and global states for the Web server.

Global commands in the "Commandstate" structure

The global commands apply to the Web server in general. You can deactivate the Web server or restart it from the control DB parameters.

Block tag	Data type	Description
init	BOOL	Evaluate the control DB and initialize the Web application
deactivate	BOOL	Deactivate the Web application

Global states in the Commandstate structure

The global states apply to the Web server in general and contain status information about the Web application.

Block tag	Data type	Description
initializing	BOOL	Web application is reading control DB
error	BOOL	Web application could not be initialized
deactivating	BOOL	Web application is terminating
deactivated	BOOL	Web application is terminated
initialized	BOOL	Web application is initialized
last_error	INT	Last error returned from a WWW instruction call (Page 831) when the return code of WWW is 16#0010: 16#0001: fragment DB structure is inconsistent 16#0002: the application name already exists 16#0003: no resources (memory) 16#0004: control DB structure is inconsistent 16#0005: fragment DB not available 16#0006: fragment DB not for AWP 16#0007: enumeration data is inconsistent 16#000D: conflicting size of the control DB

Request table

The request table is an array of structures containing commands and states that apply to individual fragment DBs. If you created fragments with the AWP_Start_Fragment (Page 826) command of type "manual", the STEP 7 user program must control these pages through the control DB. The request states are read-only and provide information about the current fragment. You use the request commands to control the current fragment.

Block tag	Data type	Description
requesttab	ARRAY [1 .. 4] OF STRUCT	Array of structures for individual fragment DB control. The Web server can process up to four fragments at a time. The array index for a particular fragment is arbitrary when the Web server is processing multiple fragments or fragments from multiple browser sessions.

Struct members of requesttab struct

Block tag	Data type	Description
page_index	UINT	Number of the current web page
fragment_index	UINT	Number of the current fragment - can be set to a different fragment
// Request Commands		
continue	BOOL	Enables current page/fragment for sending and continues with the next fragment
repeat	BOOL	Enables current page/fragment for resending and continues with the same fragment
abort	BOOL	Close http connection without sending
finish	BOOL	Send this fragment; page is complete - do not process any additional fragments
// Request states		The request states are read-only
idle	BOOL	Nothing to do, but active
waiting	BOOL	Fragment is waiting to be enabled
sending	BOOL	Fragment is sending
aborting	BOOL	User has aborted current request

Operation

Whenever your program makes changes to the control DB, it must call the WWW instruction with the number of the modified control DB as its parameter. The global commands and request commands take effect when the STEP 7 user program executes the WWW instruction (Page 831).

The STEP 7 user program can set the fragment_index explicitly, thus causing the Web server to process the specified fragment with a request command. Otherwise, the Web server processes the current fragment for the current page when the WWW instruction executes.

Possible techniques for using the fragment_index include:

- Processing the current fragment: Leave fragment_index unchanged and set the continue command.
- Skip the current fragment: Set fragment_index to 0 and set the continue command.
- Replace current fragment with a different fragment: Set the fragment_index to the new fragment ID and set the continue command.

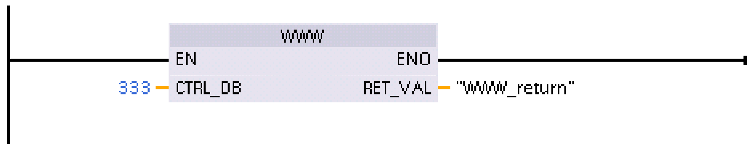
To check global states or request states that might be changing, the STEP 7 user program must call the WWW instruction to evaluate the current values of these states. A typical usage might be to call the WWW instruction periodically until a specific state occurs.

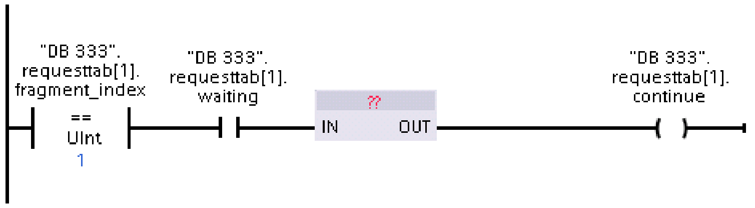
Note

If the STEP 7 user program sets more than one request command, the WWW instruction processes only one in this order of precedence: abort, finish, repeat, continue. The WWW instruction clears all of the request commands after processing.

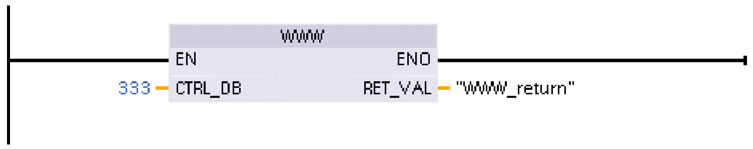
Examples

The following example shows a STEP 7 user program that is checking for a fragment with an ID of 1 to be in the waiting state, following a prior call to the WWW instruction. It might also wait for other application-specific conditions to occur. Then it performs whatever processing is necessary for the fragment, such as setting data block tags, performing calculations, or other application-specific tasks. Afterwards, it sets the continue flag so that the Web server will execute this fragment.





When the program calls the WWW instruction with this modified control DB, the user-defined Web page with this fragment can be displayed from the Web browser.



Note that this is a simplified example; the fragment to check could be in any one of the four requesttab structs in the array.

12.8 Constraints

The following IT considerations can affect your use of the Web server:

- Typically, you must use the IP address of the CPU to access the standard Web pages or user-defined Web pages, or the IP address of a wireless router with a port number. If your Web browser does not allow connecting directly to an IP address, see your IT administrator. If your local policies support DNS, you can connect to the IP address through a DNS entry to that address.
- Firewalls, proxy settings, and other site-specific restrictions can also restrict access to the CPU. See your IT administrator to resolve these issues.
- The standard Web pages use JavaScript and cookies. If your Web browser settings disable JavaScript or cookies, enable them. If you cannot enable them, some features are restricted (Page 856). Use of JavaScript and cookies in user-defined Web pages is optional. If used, you must enable them in your browser.
- The Web server supports Secure Sockets Layer (SSL). You can access the standard Web pages and user-defined Web pages with an URL of either http://ww.xx.yy.zz or https://ww.xx.yy.zz, where "ww.xx.yy.zz" represents the IP address of the CPU.
- Siemens provides a security certificate for secure access to the Web server. From the Introduction standard Web page (Page 800), you can download and import the certificate into the Internet options of your Web browser (Page 857). If you choose to not import the certificate, you will get a security verification prompt every time you access the Web server with https://.

Number of connections

The Web server supports a maximum of 30 active HTTP connections. Various actions consume the 30 connections, depending on the Web browser that you use and the number of different objects per page (.css files, images, additional .html files). Some connections persist while the Web server is displaying a page; other connections do not persist after the initial connection.

If, for example, you are using Mozilla Firefox 8, which supports a maximum of six persistent connections, you could use five browser or browser tab instances before the Web server starts dropping connections. In the case where a page is not using all six connections, you could have additional browser or browser tab instances.

Also be aware that the number of active connections can affect page performance.

Note

Log off prior to closing Web server

If you have logged in to the Web server, be sure to log off prior to closing your Web browser. The Web server supports a maximum of seven concurrent logins.

12.8.1 Feature restrictions when the Internet options disable JavaScript

The standard Web pages use HTML, JavaScript, and cookies. If your site restricts the use of JavaScript and cookies, then enable them for the pages to function properly. If you cannot enable JavaScript for your Web browser, the features that use JavaScript controls cannot run.

General

The pages do not update dynamically. You must manually refresh the page with the Refresh icon (Page 794) to view fresh data.

Diagnostic Buffer page

The Diagnostic Buffer page uses JavaScript as follows:

- Displaying the event details: With JavaScript, you select a row in the diagnostic buffer to see the details in the bottom section. Without JavaScript, you must click the event field hyperlink of a diagnostic buffer entry to see the event data in the bottom section.
- Changing the range of diagnostic buffer entries to view: With JavaScript, you use the drop-down list at the top to select the range of diagnostic buffer entries to view, and the page automatically updates. Without JavaScript, you use the drop-down list at the top to select the range of diagnostic buffer entries to view, but you must then click the "Go" link to update the diagnostic buffer page with the range you selected from the drop-down list.

Note that the "Go" and the event field hyperlinks are only visible when JavaScript is not enabled. They are not necessary and therefore are not present when JavaScript is enabled.

Module Information page

Without JavaScript enabled, the following restrictions apply:

- You cannot filter the data.
- You cannot sort fields.

Variable Status page

Without JavaScript enabled, the following restrictions apply:

- After you enter each variable, you must manually set the focus to the "New variable" row to enter a new variable.
- Selecting a display format does not automatically change the data value display to the selected format. You must click the "Monitor value" button to refresh the display with the new format.

12.8.2 Feature restrictions when the Internet options do not allow cookies

If you disable cookies in your Web browser, the following restrictions apply:

- You cannot log in.
- You cannot change the language setting.
- You cannot switch from UTC time to PLC time. Without cookies, all times are in UTC time.

12.8.3 Importing the Siemens security certificate

You can import the Siemens security certificate into your Internet options so that you won't be prompted for security verification when you enter https://ww.xx.yy.zz in your Web browser, where "ww.xx.yy.zz" is the device IP address. If you use an http:// URL and not an https:// URL, then you do not need to download and install the certificate.

Downloading the certificate

You use the "download certificate" link from the Introduction page (Page 800) to download the Siemens security certificate to your PC. The procedure varies according to which Web browser you use:

Importing the certificate to Internet Explorer

1. Click the "download certificate" link from the Introduction page. A "File Download - Security Warning" dialog pops up.
2. From the "File Download - Security Warning" dialog, click "Open" to open the file. A "Certificate" dialog appears.
3. From the "Certificate" dialog, click the "Install Certificate" button to launch the Certificate Import Wizard.
4. Follow the dialogs of the "Certificate Import Wizard" to import the certificate, letting the operating system automatically select the certificate store.

Importing the certificate to Mozilla Firefox

1. Click the "download certificate" link from the Intro page. An "Opening MiniWebCA_Cer.crt" dialog pops up.
2. Click "Save file" from the "Opening MiniWebCA_Cer.crt" dialog. A "Downloads" dialog appears.
3. From the "Downloads" dialog, double-click "MiniWebCA_Cer.crt". If you have attempted the download more than once, multiple copies show up. Just double-click any one of the "MiniWebCA_Cer.crt" entries.
4. Click "OK" if prompted to open an executable file.
5. Click "Open" on the "Open File - Security Warning" dialog if it appears. A "Certificate" dialog appears.
6. On the "Certificate" dialog, click the "Install Certificate" button.
7. Follow the dialogs of the "Certificate Import Wizard" to import the certificate, letting the operating system automatically select the certificate store.
8. If the "Security Warning" dialog appears, click "Yes" to confirm installation of the certificate.

Other browsers

Follow the conventions of your Web browser to import and install the Siemens certificate.

After you have installed the Siemens security certificate "S7-1200 Controller Family" in the Internet options for your Web browser content, you will not be required to verify a security prompt when you access the Web server with https:// ww.xx.yy.zz.

Note

The security certificate remains constant through reboots of the CPU; however, if you change the IP address of the device you must download a new certificate if you are using a browser other than Internet Explorer or Mozilla Firefox.

12.8.4 Importing CSV format data logs to non-USA/UK versions of Microsoft Excel

Data log files are in the comma-separated values (CSV) file format. You can open these files directly in Excel from the Data Logs page when your system is running the USA or UK version of Excel. In other countries, however, this format is not widely used because commas occur frequently in numerical notation.

To open a data log file that you have saved, follow these steps for non USA/UK versions of Excel:

1. Open Excel and create an empty workbook.
2. From the "Data > Import External Data" menu, select the "Import Data" command.
3. Navigate to and select the data log file you want to open. The Text Import Wizard starts.
4. From the Text Import Wizard, change the default option for "Original data type" from "Fixed width" to "Delimited".
5. Click the Next button.
6. From the Step 2 dialog, select the "Comma" check box to change the delimiter type from "Tab" to "Comma".
7. Click the Next button.
8. From the Step 3 dialog, you can optionally change the Date format from MDY (month/day/year) to another format.
9. Complete the remaining steps of the Text Import Wizard to import the file.

13 Communication processor and Modbus TCP

13.1 Using the serial communication interfaces

Two communication modules (CMs) and one communication board (CB) provide the interface for PtP communications:

- CM 1241 RS232 (Page 1263)
- CM 1241 RS422/485 (Page 1264)
- CB 1241 RS485 (Page 1260)

You can connect up to three CMs (of any type) plus a CB for a total of four communication interfaces. Install the CM to the left of the CPU or another CM. Install the CB on the front of the CPU. Refer to the installation guidelines (Page 64) for information on module installation and removal.

The serial communication interfaces have the following characteristics:

- Have an isolated port
- Support Point-to-Point protocols
- Are configured and programmed through the point-to-point communication processor instructions
- Display transmit and receive activity by means of LEDs
- Display a diagnostic LED (CMs only)
- Are powered by the CPU: No external power connection is needed.

Refer to the technical specifications for communication interfaces (Page 1250).

LED indicators

The communication modules have three LED indicators:

- Diagnostic LED (DIAG): This LED flashes red until it is addressed by the CPU. After the CPU powers up, it checks for CMs and addresses them. The diagnostic LED begins to flash green. This means that the CPU has addressed the CM, but has not yet provided the configuration to it. The CPU downloads the configuration to the configured CMs when the program is downloaded to the CPU. After a download to the CPU, the diagnostic LED on the communication module should be a steady green.
- Transmit LED (Tx): The transmit LED illuminates when data is being transmitted out the communication port.
- Receive LED (Rx): This LED illuminates when data is being received by the communication port.

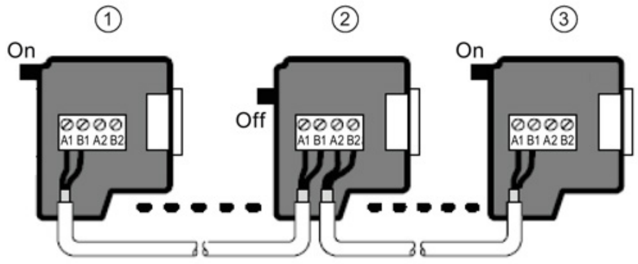
The communication board provides transmit (TxD) and receive (RxD) LEDs. It has no diagnostic LED.

13.2 Biasing and terminating an RS485 network connector

Siemens provides an RS485 network connector (Page 1280) that you can use to easily connect multiple devices to an RS485 network. The connector has two sets of terminals that allow you to attach the incoming and outgoing network cables. The connector also includes switches for selectively biasing and terminating the network.

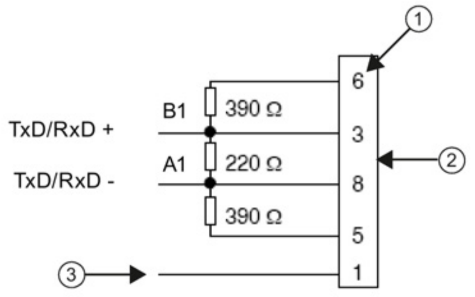
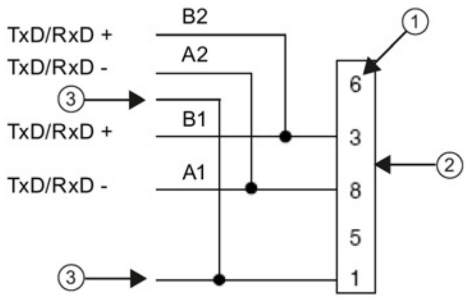
Note

You terminate and bias only the two ends of the RS485 network. The devices in between the two end devices are not terminated or biased. Bare cable shielding: Approximately 12 mm (1/2 in) must contact the metal guides of all locations.



① Switch position = On: Terminated and biased
② Switch position = Off: No termination or bias
③ Switch position = On: Terminated and biased

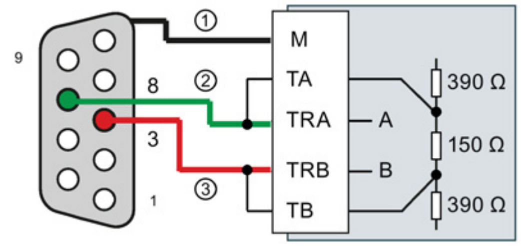
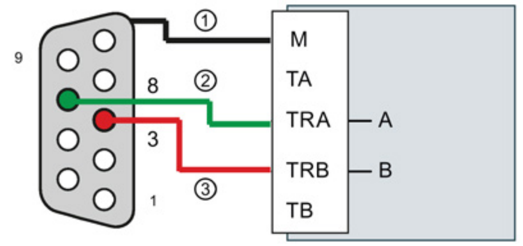
Table 13- 1 Termination and bias for the RS485 connector

Terminating device (bias ON)	Non-terminating device (bias OFF)
 TxD/RxD + TxD/RxD - B1 A1 390 Ω 220 Ω 390 Ω 6 3 8 5 1 ① ② ③	 TxD/RxD + TxD/RxD - TxD/RxD + TxD/RxD - B2 A2 B1 A1 6 3 8 5 1 ① ② ③

① Pin number
② Network connector
③ Cable shield

The CB 1241 provides internal resistors for terminating and biasing the network. To terminate and bias the connection, connect TRA to TA and connect TRB to TB to include the internal resistors to the circuit. CB 1241 does not have a 9-pin connector. The following table shows the connections to a 9-pin connector on the communications partner.

Table 13- 2 Termination and bias for the CB 1241

Terminating device (bias ON)	Non-terminating device (bias OFF)
 9 8 3 1 ① ② ③ M TA TRA TRB TB A B 390 Ω 150 Ω 390 Ω	 9 8 3 1 ① ② ③ M TA TRA TRB TB A B

① Connect M to the cable shield

② A = TxD/RxD - (Green wire / Pin 8)

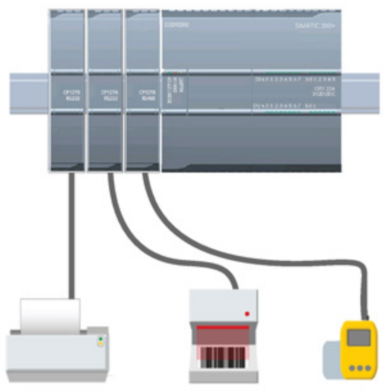
③ B = TxD/RxD + (Red wire / Pin 3)

13.3 Point-to-point (PtP) communication

The CPU supports the following Point-to-Point communication (PtP) for character-based serial protocols:

- PtP
- USS (Page 911)
- Modbus (Page 931)

PtP provides maximum freedom and flexibility, but requires extensive implementation in the user program.



PtP enables a wide variety of possibilities:

- The ability to send information directly to an external device such as a printer
- The ability to receive information from other devices such as barcode readers, RFID readers, third-party camera or vision systems, and many other types of devices
- The ability to exchange information, sending and receiving data, with other devices such as GPS devices, third-party camera or vision systems, radio modems, and many more

This type of PtP communication is serial communication that uses standard UARTs to support a variety of baud rates and parity options. The RS232 and RS422/485 communication modules (CM 1241) and the RS485 communication board (CB 1241) provide the electrical interfaces for performing the PtP communications.

PtP over PROFIBUS or PROFINET

Version V4.1 of the S7-1200 CPU together with STEP 7 V13 SP1 extends the capability of PtP to use a PROFINET or PROFIBUS distributed I/O rack to communicate to various devices (RFID readers, GPS device, and others):

- PROFINET (Page 616): You connect the Ethernet interface of the S7-1200 CPU to a PROFINET interface module. PtP communication modules in the rack with the interface module can then provide serial communications to the PtP devices.
- PROFIBUS (Page 752): You insert a PROFIBUS communication module in the left side of the rack with the S7-1200 CPU. You connect the PROFIBUS communication module to a rack containing a PROFIBUS interface module. PtP communication modules in the rack with the interface module can then provide serial communications to the PtP devices.

For this reason, the S7-1200 supports two sets of PtP instructions:

- Legacy point-to-point instructions (Page 977): These instructions existed prior to version V4.0 of the S7-1200 and only work with serial communications using a CM 1241 communication module or CB 1241 communication board.
- Point-to-point instructions (Page 877): These instructions provide all of the functionaity of the legacy instructions, plus the ability to connect to PROFINET and PROFIBUS distributed I/O. The point-to-point instructions allow you to configure the communications between the PtP communication modules in the distributed I/O rack and the PtP devices.

Note

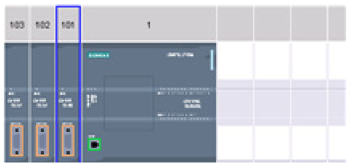
With version V4.1 of the S7-1200, you can use the point-to-point instructions for all types of point-to-point communication: serial, serial over PROFINET, and serial over PROFIBUS. STEP 7 provides the legacy point-to-point instructions only to support existing programs. The legacy instructions still function, however, with V4.1 CPUs as well as V4.0 and earlier CPUs. You do not have to convert prior programs from one set of instructions to the other.

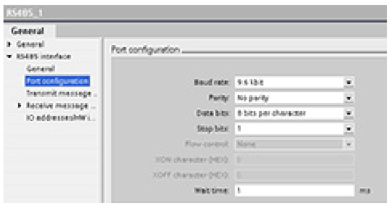
13.3.1 Configuring the communication ports

You can use either of the following methods to configure the communication interfaces:

- Use the device configuration in STEP 7 to configure the port parameters (baud and parity), the send parameters and the receive parameters. The CPU stores the device configuration settings and applies the settings after a power cycle and a RUN to STOP transition.
- Use the Port_Config (Page 879), Send_Config (Page 882), and Receive_Config (Page 884) instructions to set the parameters. The port settings set by the instructions are valid while the CPU is in RUN mode. The port settings revert to the device configuration settings after a STOP transition or power cycle.

After configuring the hardware devices (Page 145), you configure parameters for the communication interfaces by selecting one of the CMs in your rack or the CB, if configured.





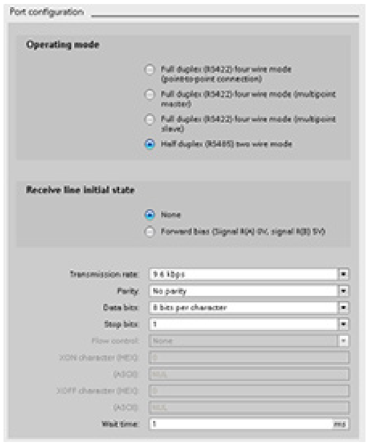
Baud rate: 9.6 kbit
Parity: No parity
Data bits: 8 bits per character
Stop bits: 1
Flow control: None
XON character (HEX): 0
XOFF character (HEX): 0
Wait time: 1 ms

The "Properties" tab of the inspector window displays the parameters of the selected CM or CB. Select "Port configuration" to edit the following parameters:

- Baud rate
- Parity
- Data bits per character
- Number of stop bits
- Flow control (RS232 only)
- Wait time

For the CM 1241 RS232 and CB RS485 (except for flow control (Page 867), which only the CM 1241 RS232 supports), the port configuration parameters are the same regardless of whether you are configuring an RS232 or an RS485 communication module or the RS485 communication board. The parameter values can differ.

For the CM 1241 RS422/485, you have additional options for port configuration as shown below. The 422 mode of the CM 1241 RS422/485 module also supports software flow control.



Port configuration

Operating mode

- ( ) Full duplex (RS422) four wire mode (point-to-point connection)
- ( ) Full duplex (RS422) four wire mode (multipoint master)
- ( ) Full duplex (RS422) four wire mode (multipoint slave)
- [✓ Half duplex (RS485) two wire mode]

Receive line initial state

- [✓ None]
- ( ) Forward bias (Signal R(A) 0V, signal R(B) 5V)

Transmission rate:	9.6 kbps
Parity:	No parity
Data bits:	8 bits per character
Stop bits:	1
Flow control:	None
XON character (HEX):	0
(ASCII):	NUL
XOFF character (HEX):	0
(ASCII):	NUL
Wait time:	1 ms

Select "Port configuration" to edit the following RS422/485 parameters:

- "Operating mode":
  – Full duplex (RS422) four wire mode (point-to-point connection)
  – Full duplex (RS422) four wire mode (multipoint master)
  – Full duplex (RS422) four wire mode (multipoint slave)
  – Half duplex (RS485) two wire mode
- "Receive line initial state":
  – None
  – Forward bias (Signal R(A) 0V, signal R(B) 5V)

The STEP 7 user program can also configure the port or change the existing configuration with the Port_Config instruction (Page 879). The instruction topic provides more detail about the operational mode and initial line state as well as other parameters.

Parameter	Definition
Baud rate	The default value for the baud rate is 9.6 Kbits per second. Valid choices are: 300 baud, 600 baud, 1.2 Kbits, 2.4 Kbits, 4.8 Kbits, 9.6 Kbits, 19.2 Kbits, 38.4 Kbits, 57.6 Kbits, 76.8 Kbits, and 115.2 Kbits.
Parity	The default value for parity is no parity. Valid choices are: No parity, even, odd, mark (parity bit always set to 1), and space (parity bit always set to 0).
Data bits per character	The number of data bits in a character. Valid choices are 7 or 8.
Number of stop bits	The number of stop bits can be either one or two. The default is one.
Flow control	For the RS232 communication module, you can select either hardware or software flow control (Page 867). If you select hardware flow control, you can select whether the RTS signal is always on, or RTS is switched. If you select software flow control, you can define the XON and XOFF characters. The RS485 communication interfaces do not support flow control. The 422 mode of the CM 1241 RS422/485 module supports software flow control.
Wait time	Wait time specifies the time that the CM or CB waits to receive CTS after asserting RTS, or for receiving an XON after receiving an XOFF, depending on the type of flow control. If the wait time expires before the communication interface receives an expected CTS or XON, the CM or CB aborts the transmit operation and returns an error to the user program. You specify the wait time in milliseconds. The range is 0 to 65535 milliseconds.
Operating mode	This selects the operating mode RS422 or RS485 and network configurations.
Receive line initial state	This selects the bias options. Valid values are none, forward bias and reverse bias. Reverse bias is used to allow cable break detection.

13.3.1.1 Managing flow control

Flow control refers to a mechanism for balancing the sending and receiving of data transmissions so that no data is lost. Flow control ensures that a transmitting device is not sending more information than a receiving device can handle. Flow control can be accomplished through either hardware or software. The RS232 CM supports both hardware and software flow control. The RS485 CM and CB do not support flow control. The 422 mode of the CM 1241 RS422/485 module supports software flow control. You specify the type of flow control either when you configure the port (Page 865) or with the PORT_CFG instruction (Page 977).

Hardware flow control works through the Request-to-send (RTS) and Clear-to-send (CTS) communication signals. With the RS232 CM, the RTS signal is output from pin 7 and the CTS signal is received through pin 8. The RS232 CM is a DTE (Data Terminal Equipment) device which asserts RTS as an output and monitors CTS as an input.

Hardware flow control: RTS switched

If you enable RTS switched hardware flow control for an RS232 CM, the module sets the RTS signal active to send data. It monitors the CTS signal to determine whether the receiving device can accept data. When the CTS signal is active, the module can transmit data as long as the CTS signal remains active. If the CTS signal goes inactive, then the transmission must stop.

Transmission resumes when the CTS signal becomes active. If the CTS signal does not become active within the configured wait time, the module aborts the transmission and returns an error to the user program. You specify the wait time in the port configuration (Page 865).

The RTS switched flow control is useful for devices that require a signal that the transmit is active. An example would be a radio modem that uses RTS as a "Key" signal to energize the radio transmitter. The RTS switched flow control will not function with standard telephone modems. Use the RTS always on selection for telephone modems.

Hardware flow control: RTS always on

In RTS always on mode, the CM 1241 sets RTS active by default. A device such as a telephone modem monitors the RTS signal from the CM and utilizes this signal as a clear-to-send. The modem only transmits to the CM when RTS is active, that is, when the telephone modem sees an active CTS. If RTS is inactive, the telephone module does not transmit to the CM.

To allow the modem to send data to the CM at any time, configure "RTS always on" hardware flow control. The CM thus sets the RTS signal active all the time. The CM will not set RTS inactive even if the module cannot accept characters. The transmitting device must ensure that it does not overrun the receive buffer of the CM.

Data Terminal Ready (DTR) and Data Set Ready (DSR) signal utilization

The CM sets DTR active for either type of hardware flow control. The module transmits only when the DSR signal becomes active. The state of DSR is only evaluated at the start of the send operation. If DSR becomes inactive after transmission has started, the transmission will not be paused.

Software flow control

Software flow control uses special characters in the messages to provide flow control. You configure Hex characters that represent XON and XOFF.

XOFF indicates that a transmission must stop. XON indicates that a transmission can resume. XOFF and XON must not be the same character.

When the transmitting device receives an XOFF character from the receiving device, it stops transmitting. Transmitting resumes when the transmitting device receives an XON character. If it does not receive an XON character within the wait time that is specified in the port configuration (Page 865), the CM aborts the transmission and returns an error to the user program.

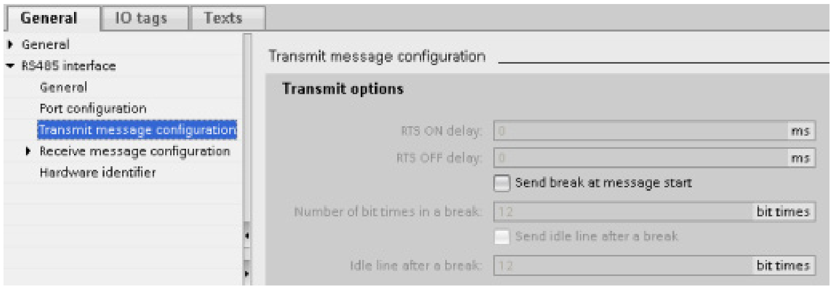
Software flow control requires full-duplex communication, as the receiving partner must be able to send XOFF to the transmitting partner while a transmission is in progress. Software flow control is only possible with messages that contain only ASCII characters. Binary protocols cannot utilize software flow control.

13.3.2 Configuring the transmit (send) and receive parameters

Before the CPU can engage in PtP communications, you must configure parameters for transmitting (or sending) messages and receiving messages. These parameters dictate how communications operate when messages are being transmitted to or received from a target device.

13.3.2.1 Configuring transmit (send) parameters

From the device configuration of the CPU, you configure how a communication interface transmits data by setting the "Transmit message configuration" properties for the selected interface.



RTS ON delay: 0 ms
RTS OFF delay: 0 ms
☐ Send break at message start
Number of bit times in a break: 12 bit times
☐ Send idle line after a break
Idle line after a break: 12 bit times

You can also dynamically configure or change the transmit message parameters from the user program by using the Send_Config (Page 882) instruction.

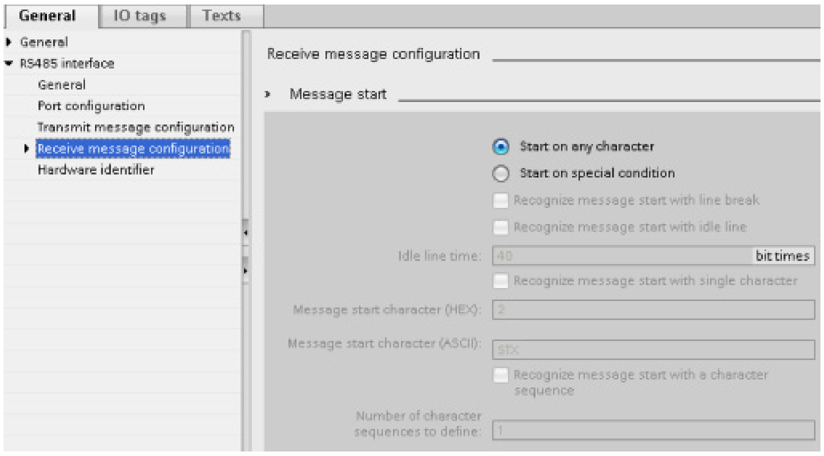
Note

Parameter values set from the Send_Config instruction in the user program override the "Transmit message configuration" properties. Note that the CPU does not retain parameters set from the Send_Config instruction in the event of power down.

Parameter	Definition
RTS On delay	Specifies the amount of time to wait after activating RTS before transmission is initiated. The range is 0 to 65535 ms, with a default value of 0. This parameter is valid only when the port configuration (Page 865) specifies hardware flow control. CTS is evaluated after the RTS On delay time has expired. This parameter is applicable for RS232 modules only.
RTS Off delay	Specifies the amount of time to wait before de-activating RTS after completion of transmission. The range is 0 to 65535 ms, with a default value of 0. This parameter is valid only when the port configuration (Page 865) specifies hardware flow control. This parameter is applicable for RS232 modules only.
Send break at message start Number of bit times in a break	Specifies that upon the start of each message, a break will be sent after the RTS On delay (if configured) has expired and CTS is active. You specify how many bit times constitute a break where the line is held in a spacing condition. The default is 12 and the maximum is 65535, up to a limit of eight seconds.
Send idle line after a break Idle line after a break	Specifies that an idle line will be sent before message start. It is sent after the break, if a break is configured. The "Idle line after a break" parameter specifies how many bit times constitute an idle line where the line is held in a marking condition. The default is 12 and the maximum is 65535, up to a limit of eight seconds.

13.3.2.2 Configuring receive parameters

From the device configuration of the CPU, you configure how a communication interface receives data, and how it recognizes both the start of and the end of a message. You set these parameters in the "Receive message configuration" properties for the selected interface.



Receive message configuration

Message start

[✓ Start on any character]
Start on special condition
Recognize message start with line break
Recognize message start with idle line
Idle line time: 40 bit times
Recognize message start with single character
Message start character (HEX): 2
Message start character (ASCII): STX
Recognize message start with a character sequence
Number of character sequences to define: 1

You can also dynamically configure or change the receive message parameters from the user program by using the Receive_Config instruction (Page 884).

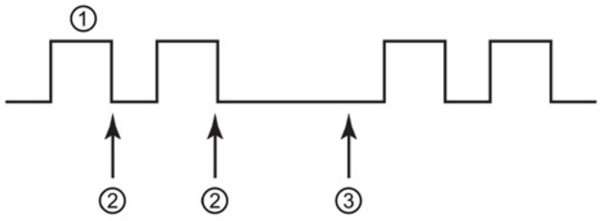
Note

Parameter values set from the Receive_Config instruction in the user program override the "Receive message configuration" properties. Note that the CPU does not retain parameters set from the RCV_CFG instruction in the event of power down or transtion to STOP.

Message start conditions

You can determine how the communication interface recognizes the start of a message. The start characters and the characters comprising the message go into the receive buffer until a configured end condition is met.

You can specify multiple start conditions. If you specify more than one start condition, all of the start conditions must be met before the message is considered started. For example, if you configure an idle line time and a specific start character, the CM or CB will first look for the idle line time requirement to be met and then the CM will look for the specified start character. If some other character is received (not the specified start character), the CM or CB will restart the start of message search by again looking for an idle line time.

Parameter	Definition
Start on Any Character	The Any Character condition specifies that any successfully received character indicates the start of a message. This character is the first character within a message.
Line Break	The Line Break condition specifies that a message receive operation starts after a break character is received.
Idle Line	The Idle Line condition specifies that a message reception starts once the receive line has been idle or quiet for the number of specified bit times. Once this condition occurs, the start of a message begins.  ① Characters ② Restarts the idle line timer ③ Idle line is detected and message receive is started
Special condition: Recognize message start with single character	Specifies that a particular character indicates the start of a message. This character is then the first character within a message. Any character that is received before this specific character is discarded. The default character is STX.
Special condition: Recognize message start with a character sequence	Specifies that a particular character sequence from up to four configured sequences indicates the start of a message. For each sequence, you can specify up to five characters. For each character position, you specify either a specific hex character, or that the character is ignored in sequence matching (wild-card character). The last specific character of a character sequence terminates that start condition sequence. Incoming sequences are evaluated against the configured start conditions until a start condition has been satisfied. Once the start sequence has been satisfied, evaluation of end conditions begins. You can configure up to four specific character sequences. You use a multiple-sequence start condition when different sequences of characters can indicate the start of a message. If any one of the character sequences is matched, the message is started.

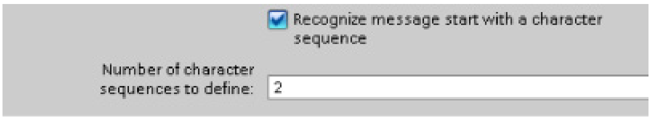
The order of checking start conditions is:

- Idle line
- Line break
- Characters or character sequences

While checking for multiple start conditions, if one of the conditions is not met, the CM or CB will restart the checking with the first required condition. After the CM or CB establishes that the start conditions have been met, it begins evaluating end conditions.

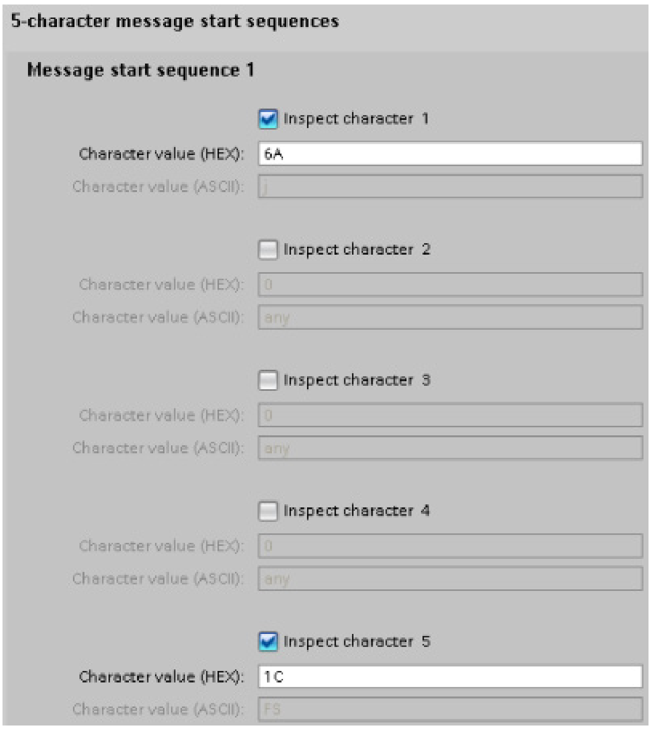
Example configuration: Start message on one of two character sequences

Consider the following start message condition configuration:



✓ Recognize message start with a character sequence

Number of character sequences to define: 2



5-character message start sequences

Message start sequence 1

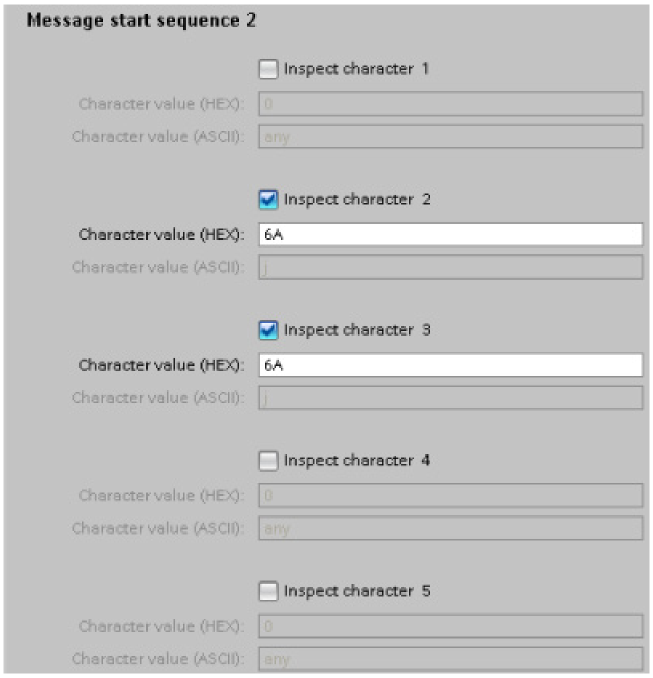
✓ Inspect character 1
Character value (HEX): 6A
Character value (ASCII): j

☐ Inspect character 2
Character value (HEX): 0
Character value (ASCII): any

☐ Inspect character 3
Character value (HEX): 0
Character value (ASCII): any

☐ Inspect character 4
Character value (HEX): 0
Character value (ASCII): any

✓ Inspect character 5
Character value (HEX): 1C
Character value (ASCII): FS


Message start sequence 2

☐ Inspect character 1
Character value (HEX): 0
Character value (ASCII): any

☑ Inspect character 2
Character value (HEX): 6A
Character value (ASCII): j

☑ Inspect character 3
Character value (HEX): 6A
Character value (ASCII): j

☐ Inspect character 4
Character value (HEX): 0
Character value (ASCII): any

☐ Inspect character 5
Character value (HEX): 0
Character value (ASCII): any

With this configuration, the start condition is satisfied when either pattern occurs:

- When a five-character sequence is received where the first character is 0x6A and the fifth character is 0x1C. The characters at positions 2, 3, and 4 can be any character with this configuration. After the fifth character is received, evaluation of end conditions begins.
- When two consecutive 0x6A characters are received, preceded by any character. In this case, evaluation of end conditions begins after the second 0x6A is received (3 characters). The character preceding the first 0x6A is included in the start condition.

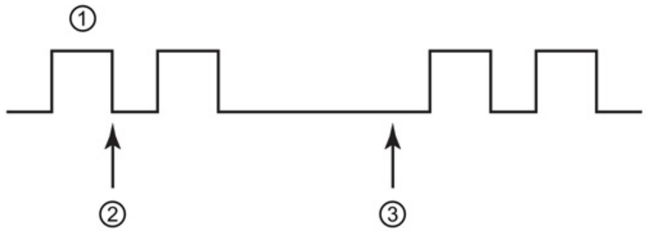
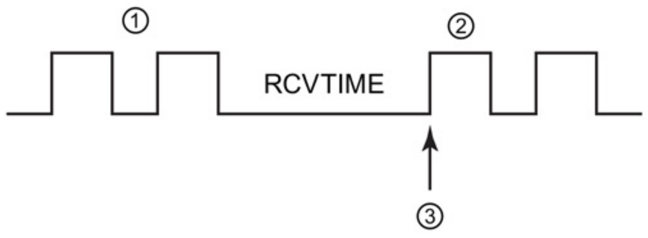
Example sequences that would satisfy this start condition are:

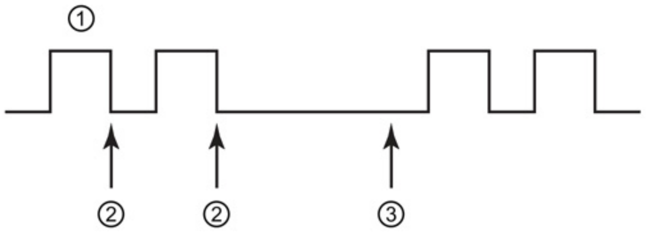
- <any character> 6A 6A
- 6A 12 14 18 1C
- 6A 44 A5 D2 1C

Message end conditions

You also configure how the communication interface recognizes the end of a message. You can configure multiple message end conditions. If any one of the configured conditions occurs, the message ends.

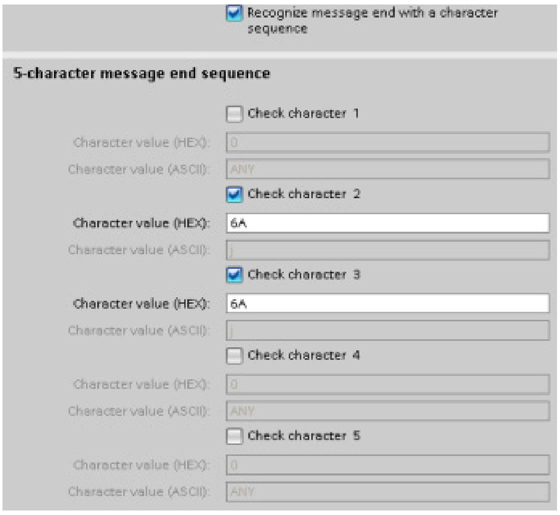
For example, you could specify an end condition with an end of message timeout of 300 milliseconds, an inter-character timeout of 40 bit times, and a maximum length of 50 bytes. The message will end if the message takes longer than 300 milliseconds to receive, or if the gap between any two characters exceeds 40 bit times, or if 50 bytes are received.

Parameter	Definition
Recognize message end by message timeout	The message end occurs when the configured amount of time to wait for the message end has expired. The message timeout period begins when a start condition has been satisfied. The default is 200 ms and the range is 0 to 65535 ms.  ① Received characters ②Start Message condition satisfied: message timer starts ③ Message timer expires and terminates the message
Recognize message end by response timeout	The message end occurs when the configured amount of time to wait for a response expires before a valid start sequence is received. The response timeout period begins when a transmission ends and the CM or CB begins the receive operation. The default response timeout is 200 ms and the range is 0 to 65535 ms. If a character is not received within the response time period, RCVTIME, then an error is returned to the corresponding RCV_PTP instruction. The response timeout does not define a specific end condition. It only specifies that a character must be successfully received within the specified time. You must configure another end condition to indicate the actual end of a message.  ① Transmitted characters ② Received characters ③ First character must be successfully received by this time.

Parameter	Definition
Recognize message end by inter-character gap	The message end occurs when the maximum configured timeout between any two consecutive characters of a message has expired. The default value for the inter-character gap is 12 bit times and the maximum number is 65535 bit times, up to a maximum of eight seconds.  ① Received characters ② Restarts the intercharacter timer ③ The intercharacter timer expires and terminates the message.
Recognize message end by receiving a fixed number of characters	The message end occurs when the specified number of characters has been received. The valid range for the fixed length is 1 to 4096. Note that for the S7-1200, this end condition is only valid for V4.0 CPUs or higher.
Recognize message end by max length	The message end occurs when the configured maximum number of characters has been received. The valid range for maximum length is 1 to 1023. This condition can be used to prevent a message buffer overrun error. When this end condition is combined with timeout end conditions and the timeout condition occurs, any valid received characters are provided even if the maximum length is not reached. This allows support for varying length protocols when only the maximum length is known.
Read message length from message	The message itself specifies the length of the message. The message end occurs when a message of the specified length has been received. The method for specifying and interpreting the message length is described below.
Recognize message end with a character	The message end occurs when a specified character is received.
Recognize message end with a character sequence	The message end occurs when a specified character sequence is received. You can specify a sequence of up to five characters. For each character position, you specify either a specific hex character, or that the character is ignored in sequence matching. Leading characters that are ignored characters are not part of the end condition. Trailing characters that are ignored characters are part of the end condition.

Example configuration: End message with a character sequence

Consider the following end message condition configuration:



[✓] Recognize message end with a character sequence

5-character message end sequence

[ ] Check character 1
Character value (HEX): 0
Character value (ASCII): ANY

[✓] Check character 2
Character value (HEX): 6A
Character value (ASCII): j

[✓] Check character 3
Character value (HEX): 6A
Character value (ASCII): j

[ ] Check character 4
Character value (HEX): 0
Character value (ASCII): ANY

[ ] Check character 5
Character value (HEX): 0
Character value (ASCII): ANY

In this case, the end condition is satisfied when two consecutive 0x6A characters are received, followed by any two characters. The character preceding the 0x6A 0x6A pattern is not part of the end character sequence. Two characters following the 0x6A 0x6A pattern are required to terminate the end character sequence. The values received at character positions 4 and 5 are irrelevant, but they must be received to satisfy the end condition.

Note

If you want your character sequence to indicate the end of the message, put the sequence in the last character positions. In the example above, if you wanted 0x6A 0x6A to end the message with no trailing characters, you would configure 0x6A in character positions 4 and 5.

Specification of message length within the message

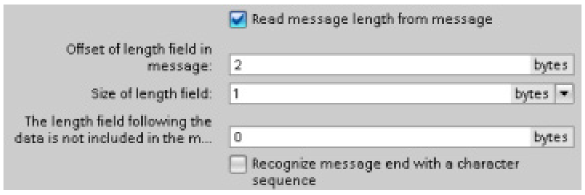
When you select the special condition where the message length is included in the message, you must provide three parameters that define information about the message length.

The actual message structure varies according to the protocol in use. The three parameters are as follows:

- n: the character position (1-based) within the message that starts the length specifier
- Length size: The number of bytes (one, two, or four) of the length specifier
- Length m: the number of characters following the length specifier that are not included in the length count

The ending characters do not need to be contiguous. The "Length m" value can be used to specify the length of a checksum field whose size is not included in the length field.

These fields appear in the Receive message configuration of the device properties:


[✓] Read message length from message
Offset of length field in message: 2 bytes
Size of length field: 1 bytes
The length field following the data is not included in the m...: 0 bytes
[ ] Recognize message end with a character sequence

Example 1: Consider a message structured according to the following protocol:

STX	Len (n)	Characters 3 to 14 counted by the length											
		ADR	PKE		INDEX		PWD		STW		HSW		BCC
1	2	3	4	5	6	7	8	9	10	11	12	13	14
STX	0x0C	xx	xxxx		xxxx		xxxx		xxxx		xxxx		xx

Configure the receive message length parameters for this message as follows:

- n = 2 (The message length starts with byte 2.)
- Length size = 1 (The message length is defined in one byte.)
- Length m = 0 (There are no additional characters following the length specifier that are not counted in the length count. Twelve characters follow the length specifier.)

In this example, the characters from 3 to 14 inclusive are the characters counted by Len (n).

Example 2: Consider another message structured according to the following protocol:

SD1	Len (n)	Len (n)	SD2	Characters 5 to 10 counted by length						FCS	ED
				DA	SA	FA	Data unit=3 bytes				
1	2	3	4	5	6	7	8	9	10	11	12
xx	0x06	0x06	xx	xx	xx	xx	xx	xx	xx	xx	xx

Configure the receive message length parameters for this message as follows:

- n = 3 (The message length starts at byte 3.)
- Length size = 1 (The message length is defined in one byte.)
- Length m = 3 (There are three characters following the length specifier that are not counted in the length. In the protocol of this example, the characters SD2, FCS, and ED are not counted in the length count. The other six characters are counted in the length count; therefore the total number of characters following the length specifier is nine.)

In this example, the characters from 5 to 10 inclusive are the characters counted by Len (n).

13.3.3 Point-to-point instructions

13.3.3.1 Common parameters for Point-to-Point instructions

Table 13- 3 Common input parameters for the PTP instructions

Parameter	Description
REQ	Many of the PtP instructions use the REQ input to initiate the operation on a low to high transition. The REQ input must be high (TRUE) for one execution of an instruction, but the REQ input can remain TRUE for as long as desired. The instruction does not initiate another operation until it has been called with the REQ input FALSE so that the instruction can reset the history state of the REQ input. This is required so that the instruction can detect the low to high transition to initiate the next operation. When you place a PtP instruction in your program, STEP 7 prompts you to identify the instance DB. Use a unique DB for each PtP instruction call. This ensures that each instruction properly handles inputs such as REQ.
PORT	A port address is assigned during communication device configuration. After configuration, a default port symbolic name can be selected from the parameter assistant drop-list. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "Constants" tab of the PLC tag table.
Bit time resolution	Several parameters are specified in a number of bit times at the configured baud rate. Specifying the parameter in bit times allows the parameter to be independent of baud rate. All parameters that are in units of bit times can be specified to a maximum number of 65535. However, the maximum amount of time that a CM or CB can measure is eight seconds.

The DONE, NDR, ERROR, and STATUS output parameters of the PtP instructions provide execution completion status for the PtP operations.

Table 13- 4 DONE, NDR, ERROR, and STATUS output parameters

Parameter	Data type	Default	Description
DONE	Bool	FALSE	Set TRUE for one execution to indicate that the last request completed without errors; otherwise, FALSE.
NDR	Bool	FALSE	Set TRUE for one execution to indicate that the requested action has completed without error and that the new data has been received; otherwise, FALSE.
ERROR	Bool	FALSE	Set TRUE for one execution to indicate that the last request completed with errors, with the applicable error code in STATUS; otherwise, FALSE.
STATUS	Word	0	Result status: • If the DONE or NDR bit is set, then STATUS is set to 0 or to an informational code. • If the ERROR bit is set, then STATUS is set to an error code. • If none of the above bits are set, then the instruction returns status results that describe the current state of the function. STATUS retains its value for the duration of the execution of the function.

Note

The DONE, NDR, and ERROR parameters are set for one execution only. Your program logic must save temporary output state values in data latches, so you can detect state changes in subsequent program scans.

Table 13- 5 Common condition codes

STATUS (W#16#....)	Description
0000	No error
7000	Function is not busy
7001	Function is busy with the first call.
7002	Function is busy with subsequent calls (polls after the first call).
8x3A	Illegal pointer in parameter x
8070	All internal instance memory in use, too many concurrent instructions in progress
8080	Port number is illegal.
8081	Timeout, module error, or other internal error
8082	Parameterization failed because parameterization is in progress in background.
8083	Buffer overflow: The CM or CB returned a received message with a length greater than the length parameter allowed.

STATUS (W#16#....)	Description
8090	Internal error: Wrong message length, wrong sub-module, or illegal message Contact customer support.
8091	Internal error: Wrong version in parameterization message Contact customer support.
8092	Internal error: Wrong record length in parameterization message Contact customer support.

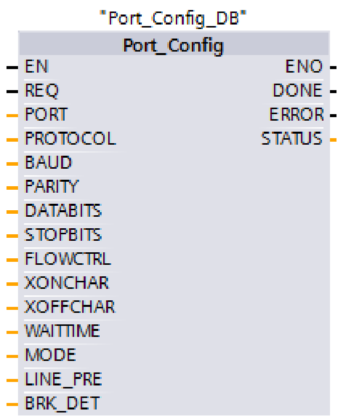
Table 13- 6 Common error classes

Class description	Error classes	Description
Port configuration	16#81Ax	Used to define common port configuration errors
Transmit configuration	16#81Bx	Used to define common transmit configuration errors
Receive configuration	16#81Cx 16#82Cx	Used to define common receive configuration errors
Transmission runtime	16#81Dx	Used to define common transmission runtime errors
Reception runtime	16#81Ex	Used to define common reception runtime errors
Signal handling	16#81Fx	Used to define common errors associated with all signal han-dling
Pointer errors	16#8p01 to 16#8p51	Used for ANY pointer errors where "p" is the parameter number of the instruction
Embedded protocol errors	16#848x 16#858x	Used for embedded protocol errors

13.3.3.2 Port_Config (Configure communication parameters dynamically) instruction

Table 13- 7 Port_Config (Port Configuration) instruction

LAD / FBD	SCL	Description

 "Port_Config_DB"
Port_Config
EN
REQ
PORT
PROTOCOL
BAUD
PARITY
DATABITS
STOPBITS
FLOWCTRL
XONCHAR
XOFFCHAR
WAITTIME
MODE
LINE_PRE
BRK_DET
ENO
DONE
ERROR
STATUS
	`"Port_Config_DB"(`
	`REQ:=_bool_in_,`
	`PORT:=_word_in_,`
	`PROTOCOL:=_uint_in_,`
	`BAUD:=_uint_in_,`
	`PARITY:=_uint_in_,`
	`DATABITS:=_uint_in_,`
	`STOPBITS:=_uint_in_,`
	`FLOWCTRL:=_uint_in_,`
	`XONCHAR:=_char_in_,`
	`XOFFCHAR:=_char_in_,`
	`WAITTIME:=_uint_in_,`
	`MODE:=_uint_in_,`
	`LINE_PRE:=_uint_in_,`
	`BRK_DET:=_uint_in_,`
	`DONE=>_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_);`
		Port_Config allows you to change port parameters such as baud rate from your program.
		You can set up the initial static configuration of the port in the device configuration properties, or just use the default values. You can execute the Port_Config in-struction in your program to change the configuration.

1 STEP 7 automatically creates the DB when you insert the instruction.

The CPU does not permanently store the values you set with the Port_Config instruction. The CPU restores the parameters configured in the device configuration when the CPU transitions from RUN to STOP mode and after a power cycle. See Configuring the communication ports (Page 865) and Managing flow control (Page 867) for more information.

Table 13- 8 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	Activate the configuration change on rising edge of this input. (Default value: False)
PORT	IN	PORT	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table. (Default value: 0)
PROTOCOL	IN	UInt	0 - Point-to-Point communication protocol (Default value) 1..n - future definition for specific protocols
BAUD	IN	UInt	Port baud rate (Default value: 6): 1 = 300 baud, 2 = 600 baud, 3 = 1200 baud, 4 = 2400 baud, 5 = 4800 baud, 6 = 9600 baud, 7 = 19200 baud, 8 = 38400 baud, 9 = 57600 baud, 10 = 76800 baud, 11 = 115200 baud
PARITY	IN	UInt	Port parity (Default value: 1): 1 = No parity, 2 = Even parity, 3 = Odd parity, 4 = Mark parity, 5 = Space parity
DATABITS	IN	UInt	Bits per character (Default value: 1): 1 = 8 data bits, 2 = 7 data bits
STOPBITS	IN	UInt	Stop bits (Default value: 1): 1 = 1 stop bit, 2 = 2 stop bits
FLOWCTRL	IN	UInt	Flow control (Default value: 1): 1 = No flow control, 2 = XON/XOFF, 3 = Hardware RTS always ON, 4 = Hardware RTS switched
XONCHAR	IN	Char	Specifies the character that is used as the XON character. This is typically a DC1 character (16#11). This parameter is only evaluated if flow control is enabled. (Default value: 16#11)
XOFFCHAR	IN	Char	Specifies the character that is used as the XOFF character. This is typically a DC3 character (16#13). This parameter is only evaluated if flow control is enabled. (Default value: 16#13)
WAITTIME	IN	UInt	Specifies how long to wait for a XON character after receiving a XOFF character, or how long to wait for the CTS signal after enabling RTS (0 to 65535 ms). This parameter is only evaluated if flow control is enabled. (Default value: 2000)
MODE	IN	UInt	Specifies the selection of the module's operating mode. • RS232 Mode (default for RS232 CM or CB) • RS422 Point-to-Point, transmitter always enabled • RS422 Multi-Point Master, transmitter always enabled • RS422 Multi-Point Slave, transmitter enabled while sending • RS485 Mode (Half-duplex, 2 wire connection) (default for RS422/RS485 CM or CB)

Parameter and type		Data type	Description
LINE_PRE	IN	UInt	Specifies the inactive (idle) line condition. For RS422 and RS485 modules the idle line condition is established by applying a bias voltage to the R(A) and R(B) signals. The following selections are possible: • Not biased (No Presetting) (default) • Biased with R(A) > R(B) ≥ 0V; RS422 only • Biased with R(B) > R(A) ≥ 0V; RS422 and RS485
BRK_DET	IN	UInt	Enables/disables communications cable break detection. Enabling cable break detection causes the module to indicate a fault when the communications cable is not connected to the module. In RS422 Point-to-Point mode cable break detection is only possible when Receive Line Presetting is used with bias applied so that R(A) > R(B) ≥ 0V. • No Cable Break Detection (default) • Cable Break Detection enabled
DONE	OUT	Bool	TRUE for one execution after the last request was completed with no error
ERROR	OUT	Bool	TRUE for one execution after the last request was completed with an error
STATUS	OUT	Word	Execution condition code (Default value: 0)

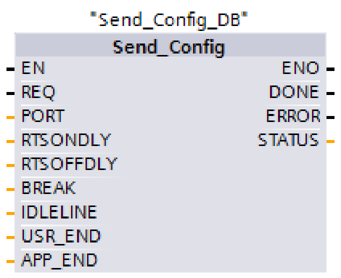
Table 13- 9 Condition codes

STATUS (W#16#....)	Description
81A0	Specific protocol does not exist.
81A1	Specific baud rate does not exist.
81A2	Specific parity option does not exist.
81A3	Specific number of data bits does not exist.
81A4	Specific number of stop bits does not exist.
80A5	Specific type of flow control does not exist.
81A6	Wait time is 0 and flow control enabled
81A7	XON and XOFF are illegal values (for example, the same value)
81A8	Error in the block header (for example, wrong block type or wrong block length)
81A9	Reconfiguration rejected because a configuration is in progress
81AA	Invalid RS422/RS485 mode of operation
81AB	Invalid presetting of the receive line for break detection
81AC	Invalid RS232 break handling
8280	Negative acknowledgement while reading the module
8281	Negative acknowledgement while writing the module
8282	DP slave or module not available

13.3.3.3 Send_Config (Configure serial transmission parameters dynamically) instruction

Table 13- 10 Send_Config (Send Configuration) instruction

LAD / FBD	SCL	Description

	`"Send_Config_DB"(`
	`REQ:=_bool_in_,`
	`PORT:=_word_in_,`
	`RTSONDLY:=_uint_in_,`
	`RTSOFFDLY:=_uint_in_,`
	`BREAK:=_uint_in_,`
	`IDLELINE:=_uint_in_,`
	`USR_END:=_string_in_,`
	`APP_END:=_string_in_,`
	`DONE=>_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_);`
		Send_Config allows the dynamic configuration of serial transmission parameters for a PtP communication port. Any queued messages within a CM or CB are discarded when Send_Config is executed.

1 STEP 7 automatically creates the DB when you insert the instruction.

You can set up the initial static configuration of the port in the device configuration properties, or just use the default values. You can execute the Send_Config instruction in your program to change the configuration.

The CPU does not permanently store the values you set with the Send_Config instruction. The CPU restores the parameters configured in the device configuration when the CPU transitions from RUN to STOP mode and after a power cycle. See Configuring transmit (send) parameters (Page 868).

Table 13- 11 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	Activate the configuration change on the rising edge of this input. (Default value: False)
PORT	IN	PORT	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table. (Default value: 0)
RTSONDLY	IN	UInt	Number of milliseconds to wait after enabling RTS before any Tx data transmission occurs. This parameter is only valid when hardware flow control is enabled. The valid range is 0 - 65535 ms. A value of 0 disables the feature. (Default value: 0)
RTSOFFDLY	IN	UInt	Number of milliseconds to wait after the Tx data transmission occurs before RTS is disabled: This parameter is only valid when hardware flow control is enabled. The valid range is 0 - 65535 ms. A value of 0 disables the feature. (Default value: 0)
BREAK	IN	UInt	This parameter specifies that a break will be sent upon the start of each message for the specified number of bit times. The maximum is 65535 bit times up to an eight second maximum. A value of 0 disables the feature. (Default value: 12)

Parameter and type		Data type	Description
IDLELINE	IN	UInt	This parameter specifies that the line will remain idle for the specified number of bit times before the start of each message. The maximum is 65535 bit times up to an eight second maximum. A value of 0 disables the feature. (Default value: 0)
USR_END*	IN	STRING[2]	Specifies the number and the characters in the end delimiter. The end delimiter is embedded in the transmit buffer (characters only) and marks the end of the transmitted message (characters are transmitted until the end delimiter is encountered). The end delimiter is appended to the end of the message. • STRING[2,0,xx,yy] – End delimiter is not used (default) • STRING[2,1,xx,yy] – End delimiter is a single character • STRING[2,2,xx,yy] – End delimiter is two characters Either USR_END or APP_END must have a length of zero.
APP_END*	IN	STRING[5]	Specifies the number and the characters to be appended to the transmitted message (only the characters are appended). STRING[5,0,aa,bb,cc,dd,ee] – End char is not used (default) • STRING[5,1,aa,bb,cc,dd,ee] – Transmit one end character • STRING[5,2,aa,bb,cc,dd,ee] – Transmit two end characters • STRING[5,3,aa,bb,cc,dd,ee]– Transmit three end characters • STRING[5,4,aa,bb,cc,dd,ee] – Transmit four end characters • STRING[5,5,aa,bb,cc,dd,ee] – Transmit five end characters
DONE	OUT	Bool	TRUE for one execution after the last request was completed with no error
ERROR	OUT	Bool	TRUE for one execution after the last request was completed with an error
STATUS	OUT	Word	Execution condition code (Default value: 0)

* Not supported for the CM and CB 1241s; you must use an empty string ("") for the parameter.

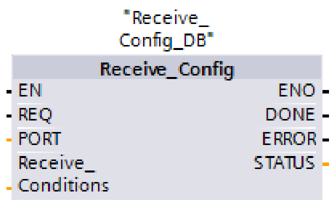
Table 13- 12 Condition codes

STATUS (W#16#....)	Description
81B0	Transmit interrupt configuration is not allowed. Contact customer suport.
81B1	Break time is greater than the maximum allowed value.
81B2	Idle time is greater than the maximum allowed value.
81B3	Error in the block header, for example, wrong block type or wrong block length
81B4	Reconfiguration rejected because a configuration is in progress
81B5	The number of end delimiters specified is greater than two or the number of end characters is greater than five
81B6	Send configuration rejected when configured for firmware embedded protocols
8280	Negative acknowledgement while reading the module
8281	Negative acknowledgement while writing the module
8282	DP slave or module not available

13.3.3.4 Receive_Config (Configure serial receive parameters dynamically) instruction

Table 13- 13 Receive_Config (Receive Configuration) instruction

LAD / FBD	SCL	Description

 "Receive_ Config_DB"
Receive_Config
EN
ENO
REQ
DONE
PORT
ERROR
Receive_ Conditions
STATUS
	`"Receive_Config_DB"(`
	`REQ:=_bool_in_,`
	`PORT:=_uint_in_,`
	`Re-`
	`ceive_Conditions:=_struct_in_,`
	`DONE=>_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_);`
		Receive_Config performs dynamic configuration of serial receiver parameters for a PtP communication port. This instruction configures the conditions that signal the start and end of a received message. Any queued messages within a CM or CB are discarded when Receive_Config is executed.

1 STEP 7 automatically creates the DB when you insert the instruction.

You can set up the initial static configuration of the communication port in the device configuration properties, or just use the default values. You can execute the Receive_Config instruction in your program to change the configuration.

The CPU does not permanently store the values you set with the Receive_Config instruction. The CPU restores the parameters configured in the device configuration when the CPU transitions from RUN to STOP mode and after a power cycle. See the topic "Configuring receive parameters (Page 869)" for more information.

Table 13- 14 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	Activate the configuration change on the rising edge of this input. (Default value: False)
PORT	IN	PORT	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table. (Default value: 0)
CONDITIONS	IN	CONDITIONS	The Conditions data structure specifies the starting and ending message conditions as described below.
DONE	OUT	Bool	TRUE for one scan, after the last request was completed with no error
ERROR	OUT	Bool	TRUE for one scan, after the last request was completed with an error
STATUS	OUT	Word	Execution condition code (Default value: 0)

Start conditions for the Receive_P2P instruction

The Receive_P2P instruction uses the configuration specified by the Receive_Config instruction to determine the beginning and ending of point-to-point communication messages. The start of a message is determined by the start conditions. The start of a message can be determined by one or a combination of start conditions. If more than one start condition is specified, all the conditions must be satisfied before the message is started.

See the topic "Configuring receive parameters (Page 869)" for a description of the message start conditions.

Parameter CONDITIONS data type structure part 1 (start conditions)

Table 13- 15 CONDITIONS structure for start conditions

Parameter and type		Data type	Description
STARTCOND	IN	UInt	Specifies the start condition (Default value: 1) • 01H - Start Char • 02H - Any Char • 04H - Line Break • 08H - Idle Line • 10H - Sequence 1 • 20H - Sequence 2 • 40H - Sequence 3 • 80H - Sequence 4
IDLETIME	IN	UInt	The number of bit times required for idle line timeout. (Default value: 40). Only used with an idle line condition. 0 to 65535
STARTCHAR	IN	Byte	The start character used with the start character condition. (Default value: B#16#2)
STRSEQ1CTL	IN	Byte	Sequence 1 ignore/compare control for each character: (Default value: B#16#0) These are the enabling bits for each character in start sequence • 01H - Character 1 • 02H - Character 2 • 04H - Character 3 • 08H - Character 4 • 10H - Character 5 Disabling the bit associated with a character means any character will match, in this sequence position.
STRSEQ1	IN	Char[5]	Sequence 1 start characters (5 characters). Default value: 0
STRSEQ2CTL	IN	Byte	Sequence 2 ignore/compare control for each character. Default value: B#16#0)
STRSEQ2	IN	Char[5]	Sequence 2 start characters (5 characters). Default value: 0
STRSEQ3CTL	IN	Byte	Sequence 3 ignore/compare control for each character. Default value: B#16#0
STRSEQ3	IN	Char[5]	Sequence 3 start characters (5 characters). Default value: 0

Parameter and type		Data type	Description
STRSEQ4CTL	IN	Byte	Sequence 4 ignore/compare control for each character. Default value: B#16#0
STRSEQ4	IN	Char[5]	Sequence 4 start characters (5 characters), Default value: 0

Example

Consider the following received hexadecimal coded message: "**68** 10 aa **68** bb 10 aa 16" and the configured start sequences shown in the table below. Start sequences begin to be evaluated when the first 68H character is successfully received. Upon successfully receiving the fourth character (the second 68H), then start condition 1 is satisfied. Once the start conditions are satisfied, the evaluation of the end conditions begins.

The start sequence processing can be terminated due to various parity, framing, or inter-character timing errors. These errors result in no received message, because the start condition was not satisfied.

Table 13- 16 Start conditions

Start condition	First Character	First Character +1	First Character +2	First Character +3	First Character +4
1	**68**H	xx	xx	**68**H	xx
2	10H	aaH	xx	xx	xx
3	dcH	aaH	xx	xx	xx
4	e5H	xx	xx	xx	xx

End conditions for the Receive_P2P instruction

The end of a message is determined by the specification of end conditions. The end of a message is determined by the first occurrence of one or more configured end conditions. The section "Message end conditions" in the topic "Configuring receive parameters (Page 869)" describes the end conditions that you can configure in the Receive_Config instruction.

You can configure the end conditions in either the properties of the communication interface in the device configuration, or from the Receive_Config instruction. Whenever the CPU transitions from STOP to RUN, the receive parameters (both start and end conditions) return to the device configuration settings. If the STEP 7 user program executes Receive_Config, then the settings are changed to the Receive_Config conditions.

Parameter CONDITIONS data type structure part 2 (end conditions)

Table 13- 17 CONDITIONS structure for end conditions

Parameter	Parameter type	Data type	Description
ENDCOND	IN	UInt 0	This parameter specifies message end condition: • 01H - Response time • 02H - Message time • 04H - Inter-character gap • 08H - Maximum length • 10H - N + LEN + M • 20H - Sequence
MAXLEN	IN	UInt 1	Maximum message length: Only used when the maximum length end condition is selected. 1 to 1024 bytes
N	IN	UInt 0	Byte position within the message of the length field. Only used with the N + LEN + M end condition. 1 to 1022 bytes
LENGTHSIZE	IN	UInt 0	Size of the byte field (1, 2, or 4 bytes). Only used with the N + LEN + M end condition.
LENGTHM	IN	UInt 0	Specify the number of characters following the length field that are not included in the value of the length field. This is only used with the N + LEN + M end condition. 0 to 255 bytes
RCVTIME	IN	UInt 200	Specify how long to wait for the first character to be received. The receive operation will be terminated with an error if a character is not successfully received within the specified time. This is only used with the response time condition. (0 to 65535 bit times with an 8 second maximum) This parameter is not a message end condition since evaluation terminates when the first character of a response is received. It is an end condition only in the sense that it terminates a receiver operation because no response is received when a response is expected. You must select a separate end condition.
MSGTIME	IN	UInt 200	Specify how long to wait for the entire message to be completely received once the first character has been received. This parameter is only used when the message timeout condition is selected. (0 to 65535 milliseconds)
CHARGAP	IN	UInt 12	Specify the number of bit times between characters. If the number of bit times between characters exceeds the specified value, then the end condition will be satisfied. This is only used with the inter-character gap condition. (0 to 65535 bit times up to 8 second maximum)

Parameter	Parameter type	Data type	Description
ENDSEQ1CTL	IN	Byte B#16#0	Sequence 1 ignore/compare control for each character: These are the enabling bits for each character for the end sequence. Character 1 is bit 0, character 2 is bit 1, …, character 5 is bit 4. Disabling the bit associated with a character means any character will match, in this sequence position.
ENDSEQ1	IN	Char[5] 0	Sequence 1 start characters (5 characters)

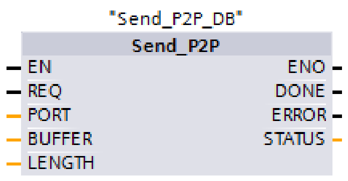
Table 13- 18 Condition codes

STATUS (W#16#....)	Description
81C0	Illegal start condition selected
81C1	Illegal end condition selected, no end condition selected
81C2	Receive interrupt enabled and this is not possible.
81C3	Maximum length end condition is enabled and max length is 0 or > 1024.
81C4	Calculated length is enabled and N is >= 1023.
81C5	Calculated length is enabled and length is not 1, 2 or 4.
81C6	Calculated length is enabled and M value is > 255.
81C7	Calculated length is enabled and calculated length is > 1024.
81C8	Response timeout is enabled and response timeout is zero.
81C9	Inter-character gap timeout is enabled and it is zero.
81CA	Idle line timeout is enabled and it is zero.
81CB	End sequence is enabled but all chars are "don't care".
81CC	Start sequence (any one of 4) is enabled but all characters are "don't care".
81CD	Invalid receive message overwrite protection selection error
81CE	Invalid receive message buffer handling on STOP to RUN transition selection error
81CF	Error in the block header, for example, wrong block type or wrong block length
8281	Negative acknowledgement while writing the module
8282	DP slave or module not available
82C0	Reconfiguration rejected because a configuration is in progress
82C1	The specified value for the number of messages that the module can buffer is greater than the maximum permitted value .
82C2	Receive configuration rejected when configured for firmware embedded protocols
8351	Data type not allowed at this Variant pointer

13.3.3.5 Send_P2P (Transmit send buffer data) instruction

Table 13- 19 Send_P2P (Send Point-to-Point data) instruction

LAD / FBD	SCL	Description

 "Send_P2P_DB"
Send_P2P
EN
ENO
REQ
DONE
PORT
ERROR
BUFFER
STATUS
LENGTH
	`"Send_P2P_DB"(`
	`    REQ:=_bool_in_,`
	`    PORT:=_word_in_,`
	`    BUFFER:=_variant_in_,`
	`    LENGTH:=_uint_in_,`
	`    DONE=>_bool_out_,`
	`    ERROR=>_bool_out_,`
	`    STATUS=>_word_out_);`
		Send_P2P initiates the transmission of the data and transfers the assigned buffer to the communication interface. The CPU program continues while the CM or CB sends the data at the assigned baud rate. Only one send operation can be pending at a given time. The CM or CB returns an error if a second Send_P2P is executed while the CM or CB is already transmitting a message.

1 STEP 7 automatically creates the DB when you insert the instruction.

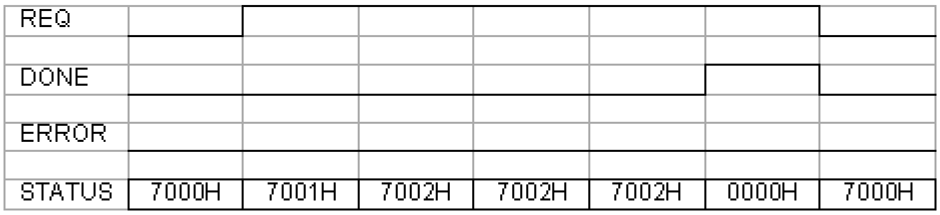
Table 13- 20 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	Activates the requested transmission on the rising edge of this transmission enable input. This initiates transfer of the contents of the buffer to the Point-to-Point communication interface. (Default value: False)
PORT	IN	PORT	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table. (Default value: 0)
BUFFER	IN	Variant	This parameter points to the starting location of the transmit buffer. (Default value: 0) **Note:** Boolean data or Boolean arrays are not supported.
LENGTH	IN	UInt	Transmitted frame length in bytes (Default value: 0) When transmitting a complex structure, always use a length of 0. When the length is 0, the instruction transmits the entire frame.
DONE	OUT	Bool	TRUE for one scan, after the last request was completed with no error
ERROR	OUT	Bool	TRUE for one scan, after the last request was completed with an error
STATUS	OUT	Word	Execution condition code (Default value: 0)

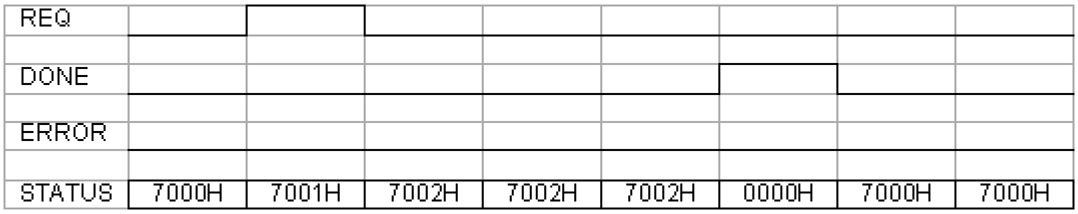
While a transmit operation is in progress, the DONE and ERROR outputs are FALSE. When a transmit operation is complete, either the DONE or the ERROR output will be set TRUE to show the status of the transmit operation. While DONE or ERROR is TRUE, the STATUS output is valid.

The instruction returns a status of 16#7001 if the communication interface accepts the transmit data. Subsequent Send_P2P executions return 16#7002, if the CM or CB is still busy transmitting. When the transmit operation is complete, the CM or CB returns the status of the transmit operation as 16#0000 (if no errors occurred). Subsequent executions of Send_P2P with REQ low return a status of 16#7000 (not busy).

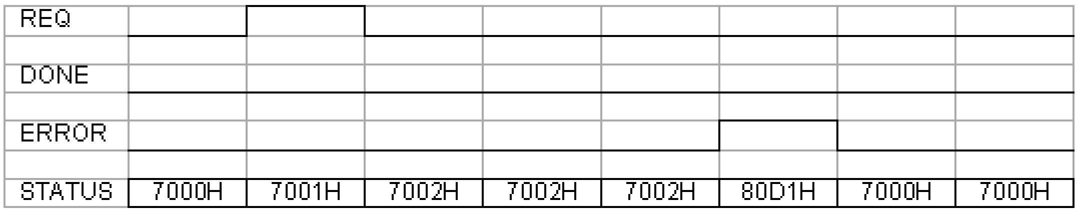
The following diagrams show the relationship of the output values to REQ. This assumes that the instruction is called periodically to check for the status of the transmission process. In the diagram below, it is assumed that the instruction is called every scan (represented by the STATUS values).



The following diagram shows how the DONE and STATUS parameters are valid for only one scan if the REQ line is pulsed (for one scan) to initiate the transmit operation.



The following diagram shows the relationship of DONE, ERROR and STATUS parameters when there is an error.



The DONE, ERROR and STATUS values are only valid until Send_P2P executes again with the same instance DB.

Table 13- 21 Condition codes

STATUS (W#16#....)	Description
81D0	New request while transmitter active
81D1	Transmit aborted because of no CTS within wait time
81D2	Transmit aborted because of no DSR from the DCE device
81D3	Transmit aborted because of queue overflow (transmit more than 1024 bytes)
81D5	Reverse bias signal (wire break condition)
81D6	Transmission request rejected because end delimiter was not found in the transmit buffer
81D7	Internal error / error in synchronization between FB and CM
81D8	Transmission attempt rejected because the port has not been configured
81DF	CM has reset the interface to the FB due to one of the following reasons • The module has restarted (Power cycle) • The CPU has reached a breakpoint and set ODIS (output disable) • The module has been reparameterized In each case the module indicates this code in the Status parameter. The module resets Status and Error to zero after the first received record for SEND_P2P.
8281	Negative acknowledgement while writing the module
8282	DP slave or module not available
8301	Illegal syntax ID at an ANY pointer
8322	Range length error when reading a parameter
8324	Range error when reading a parameter
8328	Alignment error when reading a parameter
8332	The parameter contains a DB number that is higher than the highest permitted number (DB number error).
833A	The DB for the BUFFER parameter does not exist.

Interaction of the LENGTH and BUFFER parameters

The minimum size of data that can be transmitted by the SEND_P2P instruction is one byte. The BUFFER parameter determines the size of the data to be transmitted. You cannot use the data type Bool or arrays of Bool for the BUFFER parameter.

You can always set the LENGTH parameter to 0 and ensure that SEND_P2P sends the entire data structure represented by the BUFFER parameter. If you only want to send part of a data structure in the BUFFER parameter, you can set LENGTH as follows:

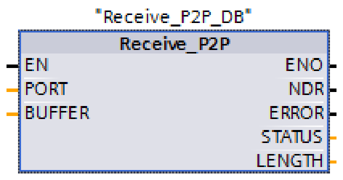
Table 13- 22 LENGTH and BUFFER parameters

LENGTH	BUFFER	Description
= 0	Not used	The complete data is sent as defined at the BUFFER parameter. You do not need to specify the number of transmitted bytes when LENGTH = 0.
> 0	Elementary data type	The LENGTH value must contain the byte count of this data type. For example, for a Word value, the LENGTH must be two. For a Dword or Real, the LENGTH must be four. Otherwise, nothing is transferred and the error 8088H is returned.
	Structure	The LENGTH value can contain a byte count less than the complete byte length of the structure, in which case the instruction sends only the first n bytes of the structure from the BUFFER, where n = LENGTH . Since the internal byte organization of a structure cannot always be determined, you might get unexpected results. In this case, use a LENGTH of 0 to send the complete structure.
	Array	The LENGTH value must contain a byte count that is less than or equal to the complete byte length of the array and which must be a multiple of the data element byte count. For example, the LENGTH parameter for an array of Words must be a multiple of two and for an array of Reals, a multiple of four. When LENGTH is specified, the instruction transfers the number of array elements that correspond to the LENGTH value in bytes. If your BUFFER, for example, contains an array of 15 Dwords (60 total bytes), and you specify a LENGTH of 20, then the first five Dwords in the array are transferred. The LENGTH value must be a multiple of the data element byte count. Otherwise, STATUS = 8088H, ERROR = 1, and no transmission occurs.
	String	The LENGTH parameter contains the number of characters to be transmitted. Only the characters of the String are transmitted. The maximum and actual length bytes of the String are not transmitted.

13.3.3.6 Receive_P2P (Enable receive messages) instruction

Table 13- 23 Receive_P2P (Receive Point-to-Point) instruction

LAD / FBD	SCL	Description

 "Receive_P2P_DB"
Receive_P2P
EN
PORT
BUFFER
ENO
NDR
ERROR
STATUS
LENGTH
	`"Receive_P2P_DB"(`
	`    PORT:=_word_in_,`
	`    BUFFER:=_variant_in_,`
	`    NDR=>_bool_out_,`
	`    ERROR=>_bool_out_,`
	`    STATUS=>_word_out_,`
	`    LENGTH=>_uint_out_);`
		Receive_P2P checks for messages that have been received in the CM or CB. If a message is available, it will be transferred from the CM or CB to the CPU. An error returns the appropriate STATUS value.

[1] STEP 7 automatically creates the DB when you insert the instruction.

Table 13- 24 Data types for the parameters

Parameter and type		Data type	Description
PORT	IN	PORT	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table. (Default value: 0)
BUFFER	IN	Variant	This parameter points to the starting location of the receive buffer. This buffer should be large enough to receive the maximum length message. Boolean data or Boolean arrays are not supported. (Default value: 0)
NDR	OUT	Bool	TRUE for one execution when new data is ready and operation is complete with no errors.
ERROR	OUT	Bool	TRUE for one execution after the operation was completed with an error.
STATUS	OUT	Word	Execution condition code (Default value: 0)
LENGTH	OUT	UInt	Length of the returned message in bytes (Default value: 0)

The STATUS value is valid when either NDR or ERROR is TRUE. The STATUS value provides the reason for termination of the receive operation in the CM or CB. This is typically a positive value, indicating that the receive operation was successful and that the receive process terminated normally. If the STATUS value is negative (the Most Significant Bit of the hexadecimal value is set), the receive operation was terminated for an error condition such as parity, framing, or overrun errors.

Each PtP communication interface can buffer up to a maximum of 1024 bytes. This could be one large message or several smaller messages. If more than one message is available in the CM or CB, the Receive_P2P instruction returns the oldest message available. A subsequent Receive_P2P instruction execution returns the next oldest message available.

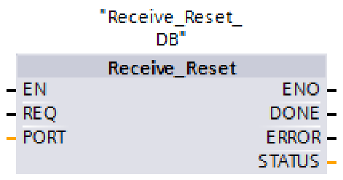
Table 13- 25 Condition codes

STATUS (W#16#...)	Description
0000	No buffer present
0094	Message terminated due to received maximum character length
0095	Message terminated because of message timeout
0096	Message terminated because of inter-character timeout
0097	Message terminated because of response timeout
0098	Message terminated because the "N+LEN+M" length condition was satisfied
0099	Message terminated because of end sequence was satisfied
8085	LENGTH parameter has a value of 0 or is greater than 4KB.
8088	The LENGTH parameter or the received length is longer than the area specified in BUFFER or the received length is longer than the area specified in BUFFER.
8090	Incorrect configuration message, wrong message length, wrong submodule, illegal message
81E0	Message terminated because the receive buffer is full
81E1	Message terminated due to parity error
81E2	Message terminated due to framing error
81E3	Message terminated due to overrun error
81E4	Message terminated because calculated length exceeds buffer size
81E5	Reverse bias signal (wire break condition)
81E6	The message queue is full. This error is reported without data. If it occurs, the module toggles between an error free data transfer and this error.
81E7	Internal error, error in synchronization between instruction and CM: set wehn a sequence error is detected
81E8	Message terminated, inter-character timeout expired before the end of message criteria was satisfied
81E9	Modbus CRC error detected (Only used by modules that support CRC generation/checking for the Modbus protocol)
81EA	Modbus telegram is too short (Only used by modules that support CRC generation/checking for the Modbus protocol)
81EB	Message terminated, because maximum message size exceeded
8201	Illegal syntax ID at an ANY pointer
8223	Range length error when writing a parameter. The parameter is located either entirely or partly outside the range of an address or that the length of a bit range is not a multiple of 8 with an ANY pointer.
8225	Range error when writing a parameter. The parameter is located in a range that is illegal for the system function.
8229	Alignment error when writing a parameter. The referenced parameter is located at bit address that is not equal to 0.
8230	Parameter is located in a read-only global DB
8231	Parameter is located in a read-only instance DB.
8232	Parameter contains a DB number that is higher than the highest block number allowed (DB number error).
823A	The DB for the BUFFER parameter does not exist.
8280	Negative acknowledgement while reading the module
8282	DP slave or module not available

13.3.3.7 Receive_Reset (Delete receive buffer) instruction

Table 13- 26 Receive_Reset (Receiver Reset) instruction

LAD / FBD	SCL	Description

 "Receive_Reset_DB"
Receive_Reset
EN
REQ
PORT
ENO
DONE
ERROR
STATUS
	```
	"Receive_Reset_DB"(
	    REQ:=_bool_in_,
	    PORT:=_word_in_,
	    DONE=>_bool_out_,
	    ERROR=>_bool_out_,
	    STATUS=>_word_out_);
	```
		Receive_Reset clears the receive buffers in the CM or CB.

[1] STEP 7 automatically creates the DB when you insert the instruction.

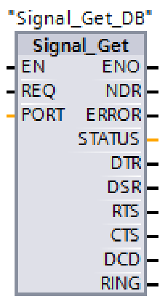
Table 13- 27 Data types for parameters

Parameter and type		Data type	Description
REQ	IN	Bool	Activates the receiver reset on the rising edge of this enable input (Default value: False)
PORT	IN	PORT	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table. (Default value: 0)
DONE	OUT	Bool	When TRUE for one scan, indicates that the last request was completed without errors.
ERROR	OUT	Bool	When TRUE, shows that the last request was completed with errors. Also, when this output is TRUE, the STATUS output will contain related error codes.
STATUS	OUT	Word	Error code (Default value: 0)

13.3.3.8 Signal_Get (Query RS-232 signals) instruction

Table 13- 28 Signal_Get (Get RS232 signals) instruction

LAD / FBD	SCL	Description

 "Signal_Get_DB"
Signal_Get
EN
REQ
PORT
ENO
NDR
ERROR
STATUS
DTR
DSR
RTS
CTS
DCD
RING
	```
	"Signal_Get_DB"(
	    REQ:=_bool_in_,
	    PORT:=_uint_in_,
	    NDR=>_bool_out_,
	    ERROR=>_bool_out_,
	    STATUS=>_word_out_,
	    DTR=>_bool_out_,
	    DSR=>_bool_out_,
	    RTS=>_bool_out_,
	    CTS=>_bool_out_,
	    DCD=>_bool_out_,
	    RING=>_bool_out_);
	```
		Signal_Get reads the current states of RS232 communication signals.
		This function is valid only for the RS232 CM.

[1] STEP 7 automatically creates the DB when you insert the instruction.

Table 13- 29 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	Get RS232 signal state values on the rising edge of this input (Default value: False)
PORT	IN	PORT	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table.
NDR	OUT	Bool	TRUE for one scan, when new data is ready and the operation is complete with no errors
ERROR	OUT	Bool	TRUE for one scan, after the operation was completed with an error
STATUS	OUT	Word	Execution condition code (Default value: 0)
DTR	OUT	Bool	Data terminal ready, module ready (output). Default value: False
DSR	OUT	Bool	Data set ready, communication partner ready (input). Default value: False
RTS	OUT	Bool	Request to send, module ready to send (output). Default value: False
CTS	OUT	Bool	Clear to send, communication partner can receive data (input). Default value: False
DCD	OUT	Bool	Data carrier detect, receive signal level (always False, not supported)
RING	OUT	Bool	Ring indicator, indication of incoming call (always False, not supported)

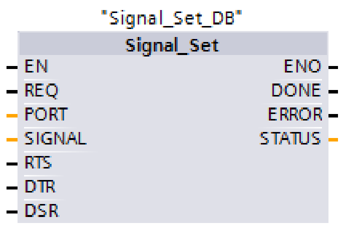
Table 13- 30 Condition codes

STATUS (W#16#....)	Description
81F0	CM or CB is RS485 and no signals are available
81F4	Error in the block header, for example, wrong block type or wrong block length
8280	Negative acknowledgement while reading the module
8282	DP slave or module not available

13.3.3.9 Signal_Set (Set RS-232 signals) instruction

Table 13- 31 Signal_Set (Set RS232 signals) instruction

LAD / FBD	SCL	Description

 "Signal_Set_DB"
Signal_Set
EN
REQ
PORT
SIGNAL
RTS
DTR
DSR
ENO
DONE
ERROR
STATUS
	`"Signal_Set_DB"(`
	`    REQ:=_bool_in_,`
	`    PORT:=_word_in_,`
	`    SIGNAL:=_byte_in_,`
	`    RTS:=_bool_in_,`
	`    DTR:=_bool_in_,`
	`    DSR:=_bool_in_,`
	`    DONE=>_bool_out_,`
	`    ERROR=>_bool_out_,`
	`    STATUS=>_word_out_);`
		Signal_Set sets the states of RS232 communication signals.
		This function is valid only for the RS232 CM.

¹ STEP 7 automatically creates the DB when you insert the instruction.

Table 13- 32 Data types for parameters

Parameter and type		Data type	Description
REQ	IN	Bool	Start the set RS232 signals operation, on the rising edge of this input (Default value: False)
PORT	IN	PORT	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table. (Default value: 0)
SIGNAL	IN	Byte	Selects which signal to set: (multiple allowed). Default value: 0 • 01H = Set RTS • 02H = Set DTR • 04H = Set DSR
RTS	IN	Bool	Request to send, module ready to send value to set (true or false), Default value: False
DTR	IN	Bool	Data terminal ready, module ready to send value to set (true or false). Default value: False
DSR	IN	Bool	Data set ready (only applies to DCE type interfaces), not used.
DONE	OUT	Bool	TRUE for one execution after the last request was completed with no error
ERROR	OUT	Bool	TRUE for one execution after the last request was completed with an error
STATUS	OUT	Word	Execution condition code (Default value: 0)

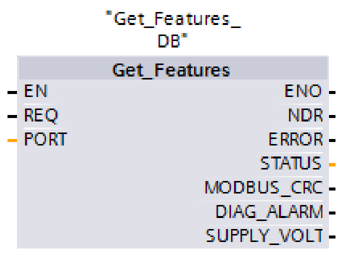
Table 13- 33 Condition codes

STATUS (W#16#....)	Description
81F0	CM or CB is RS485 and no signals can be set
81F1	Signals cannot be set because of Hardware flow control
81F2	Cannot set DSR because module is DTE
81F3	Cannot set DTR because module is DCE
81F4	Error in the block header, for example, wrong block type or wrong block length
8280	Negative acknowledgement while reading the module
8281	Negative acknowledgement while writing the module
8282	DP slave or module not available

13.3.3.10 Get_Features

Table 13- 34 Get_Features (Get advanced features) instruction

LAD / FBD	SCL	Description

 "Get_Features_ DB"
Get_Features
EN
REQ
PORT
ENO
NDR
ERROR
STATUS
MODBUS_CRC
DIAG_ALARM
SUPPLY_VOLT
	`"Get_Features_DB"(`
	`REQ:=_bool_in_,`
	`PORT:=_word_in_,`
	`NDR:=_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_,`
	`MODBUS_CRC=>_bool_out_,`
	`DIAG_ALARM=>_bool_out_,`
	`SUPPLY_VOLT=>_bool_out);`
		Get_Features performs reads the advanced feature capabilities of a moduler.

1 STEP 7 automatically creates the DB when you insert the instruction.

Use the Get_Features instruction to read the advanced feature capabilites of a module.

Table 13- 35 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	Activate the configuration change on the rising edge of this input. (Default value: False)
PORT	IN	PORT	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table. (Default value: 0)
NDR	OUT	Bool	Indicates that new data is ready.
ERROR	OUT	Bool	TRUE for one scan, after the last request was completed with an error
STATUS	OUT	Word	Execution condition code (Default value: 0)
MODBUS_CRC*	OUT	Bool	MODBUS CRC generation and checking

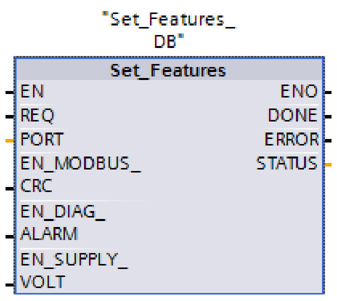
Parameter and type		Data type	Description
DIAG_ALARM*	OUT	Bool	Diagnostic alarm generation
SUPPLY_VOLT*	OUT	Bool	Diagnostics for missing supply voltage L+ is available

*Get_Features returns TRUE (1) if the feature is available, FALSE (0) if the feature is not available

13.3.3.11 Set_Features

Table 13- 36 Set_Features (Get advanced features) instruction

LAD / FBD	SCL	Description

 "Set_Features_ DB"
Set_Features
EN
REQ
PORT
EN_MODBUS_ CRC
EN_DIAG_ ALARM
EN_SUPPLY_ VOLT
ENO
DONE
ERROR
STATUS
	`"Set_Features_DB"(`
	`REQ:=_bool_in_,`
	`PORT:=_word_in_,`
	`EN_MODBUS_CRC:=_bool_in_,`
	`EN_DIAG_ALARM:=_bool_in_,`
	
	`EN_SUPPLY_VOLT:=_bool_in_,`
	`DONE=>_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_);`
		Set_Features enables the advanced features that a module supports.

[1] STEP 7 automatically creates the DB when you insert the instruction.

Use the Get_Features instruction to read the advanced feature capabilites of a module.

Table 13- 37 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	Activate the configuration change on the rising edge of this input. (Default value: False)
PORT	IN	PORT	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table. (Default value: 0)
EN_MODBUS_CRC	IN	Bool	Enable MODBUS CRC generation and checking: • 0: CRC calculation tuned OFF (default) • 1: CRC calculation turned ON Note: Only V2.1 CMs, V4.1 CPUs with CBs, and CM PtP modules for distributed I/O support this parameter.
EN_DIAG_ALARM	IN	Bool	Enable diagnostic alarm generation: • 0: Diagnostic alarm turned OFF • 1: Diagnostic alarm turned ON (default)

Parameter and type		Data type	Description
EN_SUPPLY_VOLT	IN	Bool	Enable diagnostics for missing supply voltage L+: • 0: Supply voltage diagnostic disabled (default) • 1: Supply voltage diagnostic enabled
DONE	OUT	Bool	Indicates that set features is done
ERROR	OUT	Bool	TRUE for one scan, after the last request was completed with an error
STATUS	OUT	Word	Execution condition code (Default value: 0)

13.3.4 Programming the PtP communications

STEP 7 provides extended instructions that enable the user program to perform Point-to-Point communications with a protocol designed and implemented in the user program. These instructions fall into two categories:

- Configuration instructions
- Communication instructions

Configuration instructions

Before your user program can engage in PtP communication, you must configure the communication interface port and the parameters for sending data and receiving data.

You can perform the port configuration and message configuration for each CM or CB through the device configuration or through these instructions in your user program:

- Port_Config (Page 879)
- Send_Config (Page 882)
- Receive_Config (Page 884)

Communication instructions

The PtP communication instructions enable the user program to send messages to and receive messages from the communication interfaces. For information about transferring data with these instructions, see the section on data consistency (Page 185).

All of the PtP functions operate asynchronously. The user program can use a polling architecture to determine the status of transmissions and receptions. Send_P2P and Receive_P2P can execute concurrently. The communication modules and communication board buffer the transmit and receive messages as necessary up to a maximum buffer size of 1024 bytes.

The CMs and CB send messages to and receive messages from the actual point-to-point devices. The message protocol is in a buffer that is either received from or sent to a specific communication port. The buffer and port are parameters of the send and receive instructions:

- Send_P2P (Page 889)
- Receive_P2P (Page 893)

Additional instructions provide the capability to reset the receive buffer, and to get and set specific RS232 signals:

- Receive_Reset (Page 895)
- Signal_Get (Page 895)
- Signal_Set (Page 897)

13.3.4.1 Polling architecture

The STEP 7 user program must call the S7-1200 point-to-point instructions cyclically/periodically to check for received messages. Polling the send tells the user program when the transmit has completed.

Polling architecture: master

The typical sequence for a master is as follows:

1. A Send_P2P (Page 889) instruction initiates a transmission to the CM or CB.
2. The Send_P2P instruction executes on subsequent scans to poll for the transmit complete status.
3. When the Send_P2P instruction indicates that the transmission is complete, the user code can prepare to receive the response.
4. The Receive_P2P (Page 893) instruction executes repeatedly to check for a response. When the CM or CB has collected a response message, the Receive_P2P instruction copies the response to the CPU and indicates that new data has been received.
5. The user program can process the response.
6. Go to step 1 and repeat the cycle.

Polling architecture: slave

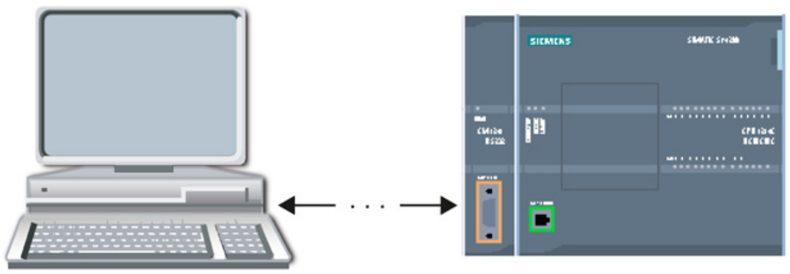
The typical sequence for a slave is as follows:

1. The user program executes the Receive_P2P instruction every scan.
2. When the CM or CB has received a request, the Receive_P2P instruction indicates that new data is ready and the request is copied into the CPU.
3. The user program services the request and generates a response.
4. Use a Send_P2P instruction to send the response back to the master.
5. Repeatedly execute Send_P2P to be sure the transmit occurs.
6. Go to step 1 and repeat the cycle.

The slave must be responsible for calling Receive_P2P frequently enough to receive a transmission from the master before the master times out while waiting for a response. To accomplish this task, the user program can call RCV_PTP from a cyclic OB, where the cycle time is sufficient to receive a transmission from the master before the timeout period expires. If you set the cycle time for the OB to provide for two executions within the timeout period of the master, the user program can receive transmissions without missing any.

13.3.5 Example: Point-to-Point communication

In this example, an S7-1200 CPU communicates to a PC with a terminal emulator through a CM 1241 RS232 module. The point-to-point configuration and STEP 7 program in this example illustrate how the CPU can receive a message from the PC and echo the message back to the PC.



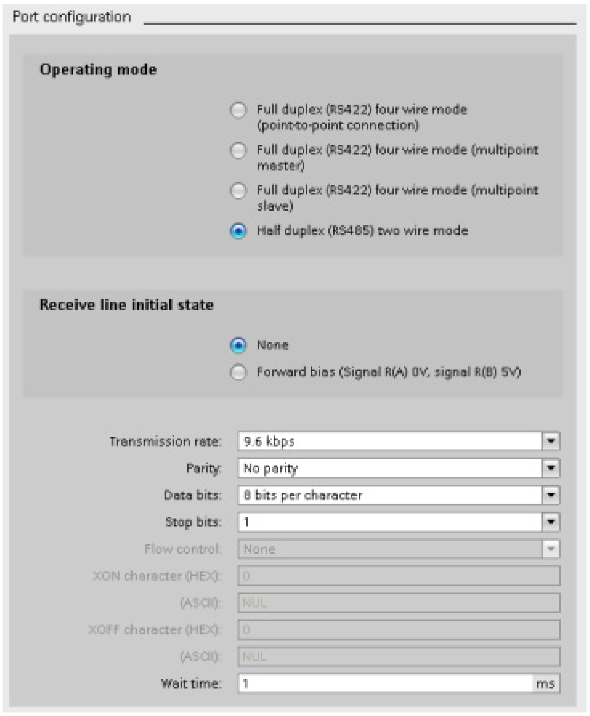
You must connect the communication interface of the CM 1241 RS232 module to the RS232 interface of the PC, which is normally COM1. Because both of these ports are Data Terminal Equipment (DTE), you must switch the receive and transmit pins (2 and 3) when connecting the two ports, which you can accomplish by either of the following methods:

- Use a NULL modem adapter to swap pins 2 and 3 together with a standard RS232 cable.
- Use a NULL modem cable, which already has pins 2 and 3 swapped. You can usually identify a NULL modem cable as one with two female 9-pin D connector ends.

13.3.5.1 Configuring the communication module

You can configure the CM 1241 from the Device configuration in STEP 7 or with user program instructions. This example uses the Device configuration method.

- Port configuration: Click the communication port of the CM module from the Device configuration, and configure the port as shown:



Port configuration

Operating mode

- ○ Full duplex (RS422) four wire mode (point-to-point connection)
- ○ Full duplex (RS422) four wire mode (multipoint master)
- ○ Full duplex (RS422) four wire mode (multipoint slave)
- ● Half duplex (RS485) two wire mode

Receive line initial state

- ● None
- ○ Forward bias (Signal R(A) 0V, signal R(B) 5V)

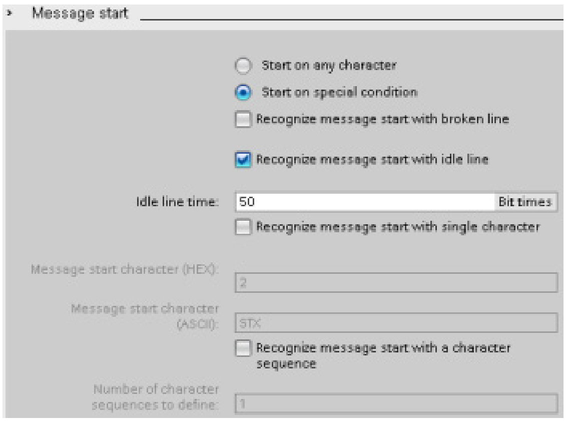
Transmission rate:	9.6 kbps
Parity:	No parity
Data bits:	8 bits per character
Stop bits:	1
Flow control:	None
XON character (HEX):	0
(ASCII):	NUL
XOFF character (HEX):	0
(ASCII):	NUL
Wait time:	1 ms

Note

The configuration settings for "Operating mode" and "Receive line initial state" are only applicable for the CM 1241 (RS422/RS485) module. The other CM 1241 modules do not have these port configuration settings. Refer to Configuring the RS422 and RS485 (Page 905).

- Transmit message configuration: Accept the default for transmit message configuration. No break is to be sent at message start.

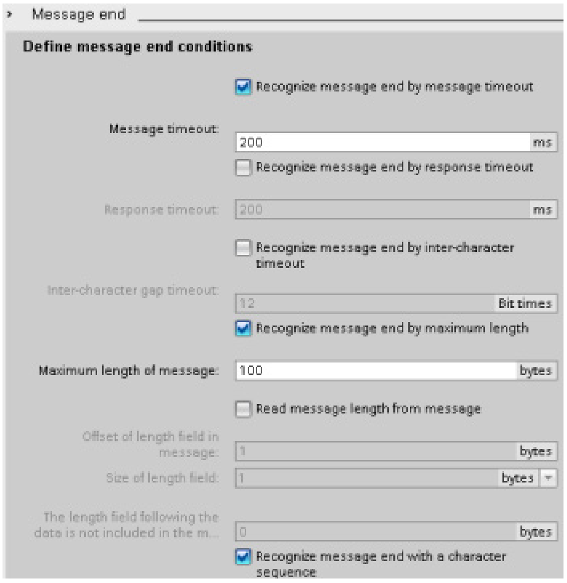
- Receive message start configuration: Configure the CM 1241 to start receiving a message when the communication line is inactive for at least 50 bit times (about 5 milliseconds at 9600 baud = 50 * 1/9600):



Message start

Start on any character
[✓ Start on special condition]
Recognize message start with broken line
[✓ Recognize message start with idle line]
Idle line time: 50 Bit times
Recognize message start with single character
Message start character (HEX): 2
Message start character (ASCII): STX
Recognize message start with a character sequence
Number of character sequences to define: 1

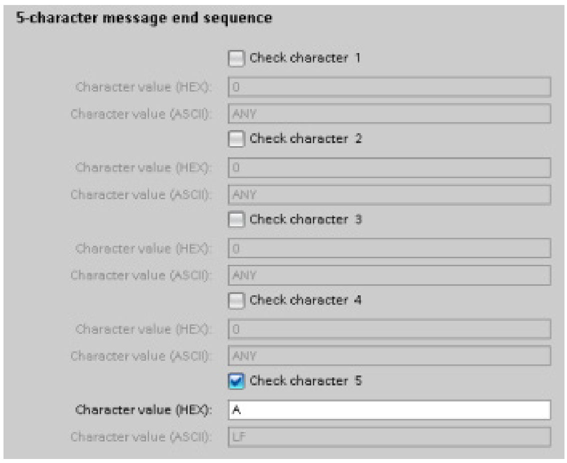
- Receive message end configuration: Configure the CM 1241 to end a message when it receives a maximum of 100 bytes or a linefeed character (10 decimal or a hexadecimal). The end sequence allows up to five end characters in sequence. The fifth character in the sequence is the linefeed character. The preceding four end sequence characters are "don’t care" or unselected characters. The CM 1241 does not evaluate the "don’t care" characters but looks for a linefeed character preceded by zero or more "don’t care" characters to indicate the message end.



Message end

Define message end conditions

[✓ Recognize message end by message timeout]
Message timeout: 200 ms
Recognize message end by response timeout
Response timeout: 200 ms
Recognize message end by inter-character timeout
Inter-character gap timeout: 12 Bit times
[✓ Recognize message end by maximum length]
Maximum length of message: 100 bytes
Read message length from message
Offset of length field in message: 1 bytes
Size of length field: 1 bytes
The length field following the data is not included in the m...: 0 bytes
[✓ Recognize message end with a character sequence]



5-character message end sequence

☐ Check character 1
Character value (HEX): 0
Character value (ASCII): ANY

☐ Check character 2
Character value (HEX): 0
Character value (ASCII): ANY

☐ Check character 3
Character value (HEX): 0
Character value (ASCII): ANY

☐ Check character 4
Character value (HEX): 0
Character value (ASCII): ANY

✓ Check character 5
Character value (HEX): A
Character value (ASCII): LF

13.3.5.2 RS422 and RS485 operating modes

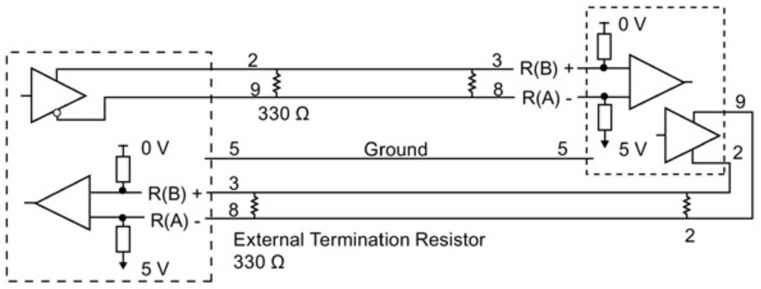
Configuring the RS422

For RS422 mode, there are three operating modes depending on your network configuration. Select one of these operating modes based on the devices in your network. The different selections for Receive line initial state reference the cases shown below for more details.

- Full duplex (RS422) four wire mode (point-to-point connection): select this option when there are two devices on your network. In the Receive line initial state:
  - Select none when you supply the bias and termination (Case 3).
  - Select forward bias to use internal bias and termination (Case 2).
  - Select reverse bias to use internal bias and termination, and enable cable break detection for both devices (Case 1).
- Full duplex (RS422) four wire mode (multipoint master): select this option for the master device when you have a network with one master and multiple slaves. In the Receive line initial state:
  - Select none when you supply the bias and termination (Case 3).
  - Select forward bias to use internal bias and termination (Case 2).
  - Cable break detection is not possible in this mode.
- Full duplex (RS422) four wire mode (multipoint slave): Select this option for all the slave devices when you have a network with one master and multiple slaves. In the Receive line initial state:
  - Select none when you supply the bias and termination (Case 3).
  - Select forward bias to use internal bias and termination (Case 2).
  - Select reverse bias to use internal bias and termination, and enable cable break detection for the slaves (Case 1).

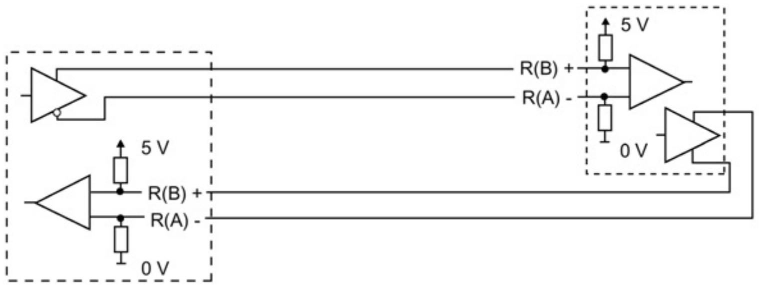
Case 1: RS422 with cable break detection

- Mode of operation: RS422
- Receive line initial state: Reverse bias (biased with R(A) > R(B) > 0V)
- Cable break: Cable break detection enabled (transmitter always active)



Case 2: RS422 No cable break detection, forward bias

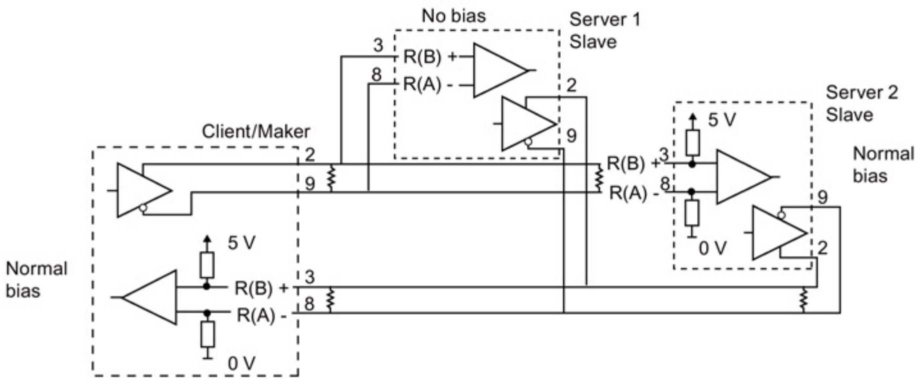
- Mode of operation: RS422
- Receive line initial state: Forward bias (biased with R(B) > R(A) > 0 V)
- Cable break: No cable break detection (transmitter enabled only while transmitting)



Case 3: RS422: No cable break detection, no bias

- Mode of operation: RS422
- Receive line initial state: no bias
- Cable break: No cable break detection (transmitter enabled only while transmitting)

Bias and termination are added by the user at the end nodes of the network.



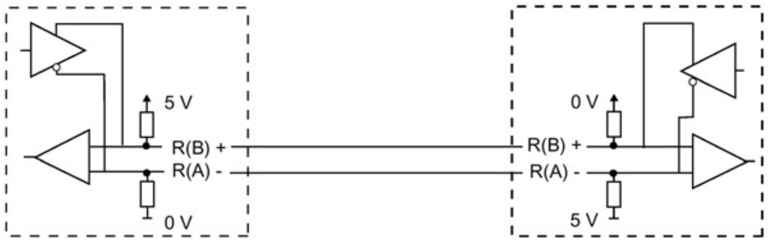
Configuring the RS485

For RS485 mode, there is only one operating mode. The different selections for Receive line initial state reference the cases shown below for more details.

- Half duplex (RS485) two wire mode. In the Receive line initial state:
  - Select none when you supply the bias and termination (Case 5).
  - Select forward bias to use internal bias and termination (Case 4).

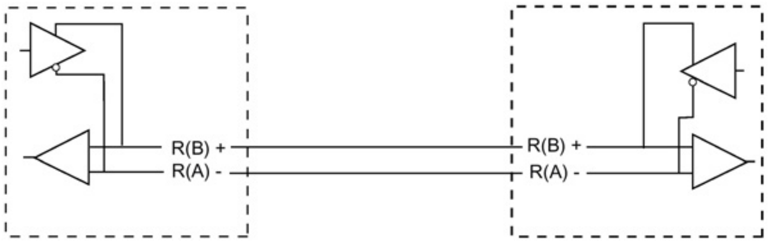
Case 4: RS485: Forward bias

- Mode of operation: RS485
- Receive line initial state: Forward bias (biased with R(B) > R(A) > 0 V)



Case 5: RS485: No bias (external bias)

- Mode of operation: RS485
- Receive line initial state: No bias (external bias required)

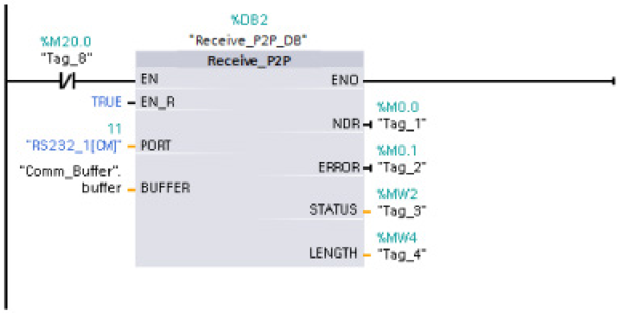


13.3.5.3 Programming the STEP 7 program

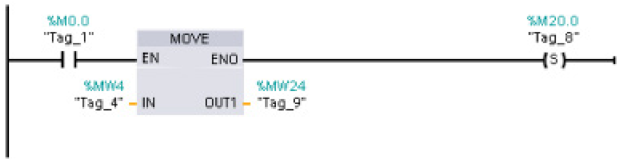
The example program uses a global data block for the communication buffer, a RCV_PTP instruction (Page 987) to receive data from the terminal emulator, and a SEND_PTP instruction (Page 985) to echo the buffer back to the terminal emulator. To program the example, add the data block configuration and Main program block OB 1 as described below.

Global data block "Comm_Buffer": Create a global data block (DB) and name it "Comm_Buffer". Create one value in the data block called "buffer" with a data type of "array [0 .. 99] of byte".

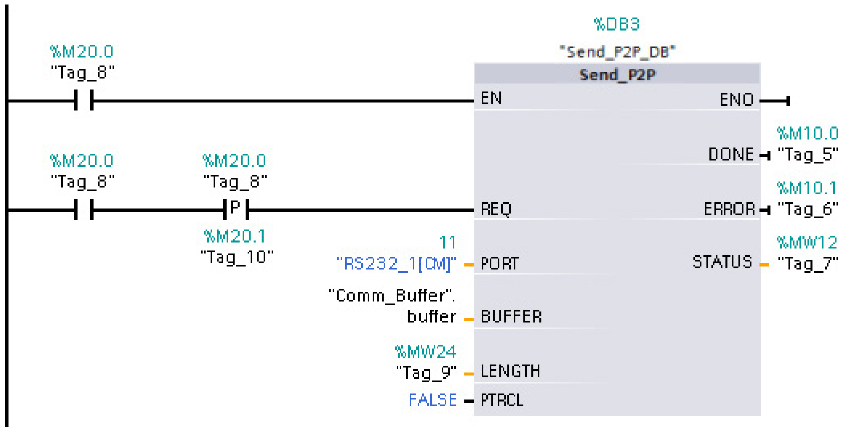
Network 1: Enable the RCV_PTP instruction whenever SEND_PTP is not active. Tag_8 at MW20.0 indicates when sending is complete in Network 4, and when the communication module is thus ready to receive a message.



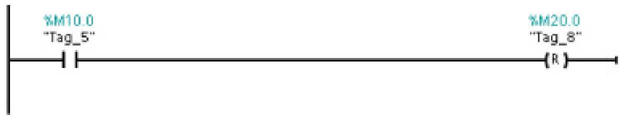
Network 2: Use the NDR value (Tag_1 at M0.0) set by the RCV_PTP instruction to make a copy of the number of bytes received and to set a flag (Tag_8 at M20.0) to trigger the SEND_PTP instruction.



Network 3: Enable the SEND_PTP instruction when the M20.0 flag is set. Also use this flag to set the REQ input to TRUE for one scan. The REQ input tells the SEND_PTP instruction that a new request is to be transmitted. The REQ input must only be set to TRUE for one execution of SEND_PTP. The SEND_PTP instruction is executed every scan until the transmit completes. The transmit is complete when the last byte of the message has been transmitted from the CM 1241. When the transmit is complete, the DONE output (Tag_5 at M10.0) is set TRUE for one execution of SEND_PTP.



Network 4: monitor the DONE output of SEND_PTP and reset the transmit flag (Tag_8 at M20.0) when the transmit operation is complete. When the transmit flag is reset, the RCV_PTP instruction in Network 1 is enabled to receive the next message.



13.3.5.4 Configuring the terminal emulator

You must set up the terminal emulator to support the example program. You can use most any terminal emulator on your PC, such as HyperTerminal. Make sure that the terminal emulator is in the disconnected mode before editing the settings as follows:

1. Set the terminal emulator to use the RS232 port on the PC (normally COM1).
2. Configure the port for 9600 baud, 8 data bits, no parity (none), 1 stop bit and no flow control.
3. Change the settings of the terminal emulator to emulate an ANSI terminal.
4. Configure the terminal emulator ASCII setup to send a line feed after every line (after the user presses the Enter key).
5. Echo the characters locally so that the terminal emulator displays what is typed.

13.3.5.5 Running the example program

To exercise the example program, follow these steps:

1. Download the STEP 7 program to the CPU and ensure that it is in RUN mode.
2. Click the "connect" button on the terminal emulator to apply the configuration changes and open a terminal session to the CM 1241.
3. Type characters at the PC and press Enter.

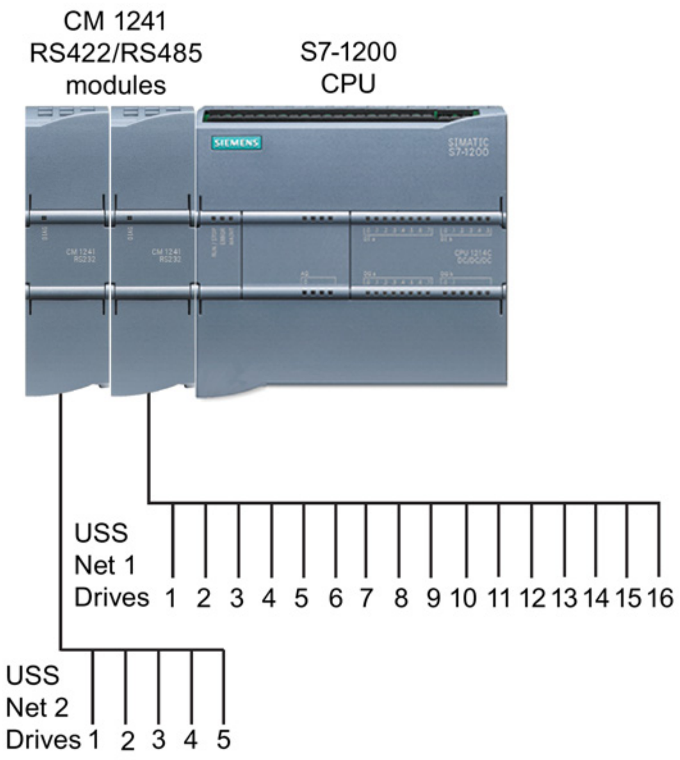
The terminal emulator sends the characters to the CM 1241 and to the CPU. The CPU program then echoes the characters back to the terminal emulator.

13.4 Universal serial interface (USS) communication

13.4.1 Overview

The USS instructions control the operation of motor drives which support the universal serial interface (USS) protocol. You can use the USS instructions to communicate with multiple drives through RS485 connections to CM 1241 RS485 communication modules or a CB 1241 RS485 communication board. Up to three CM 1241 RS422/RS485 modules and one CB 1241 RS485 board can be installed in a S7-1200 CPU. Each RS485 port can operate up to sixteen drives.

The USS protocol uses a master-slave network for communications over a serial bus. The master uses an address parameter to send a message to a selected slave. A slave itself can never transmit without first receiving a request to do so. Direct message transfer between the individual slaves is not possible. USS communication operates in half-duplex mode. The following USS illustration shows a network diagram for an example drive application.



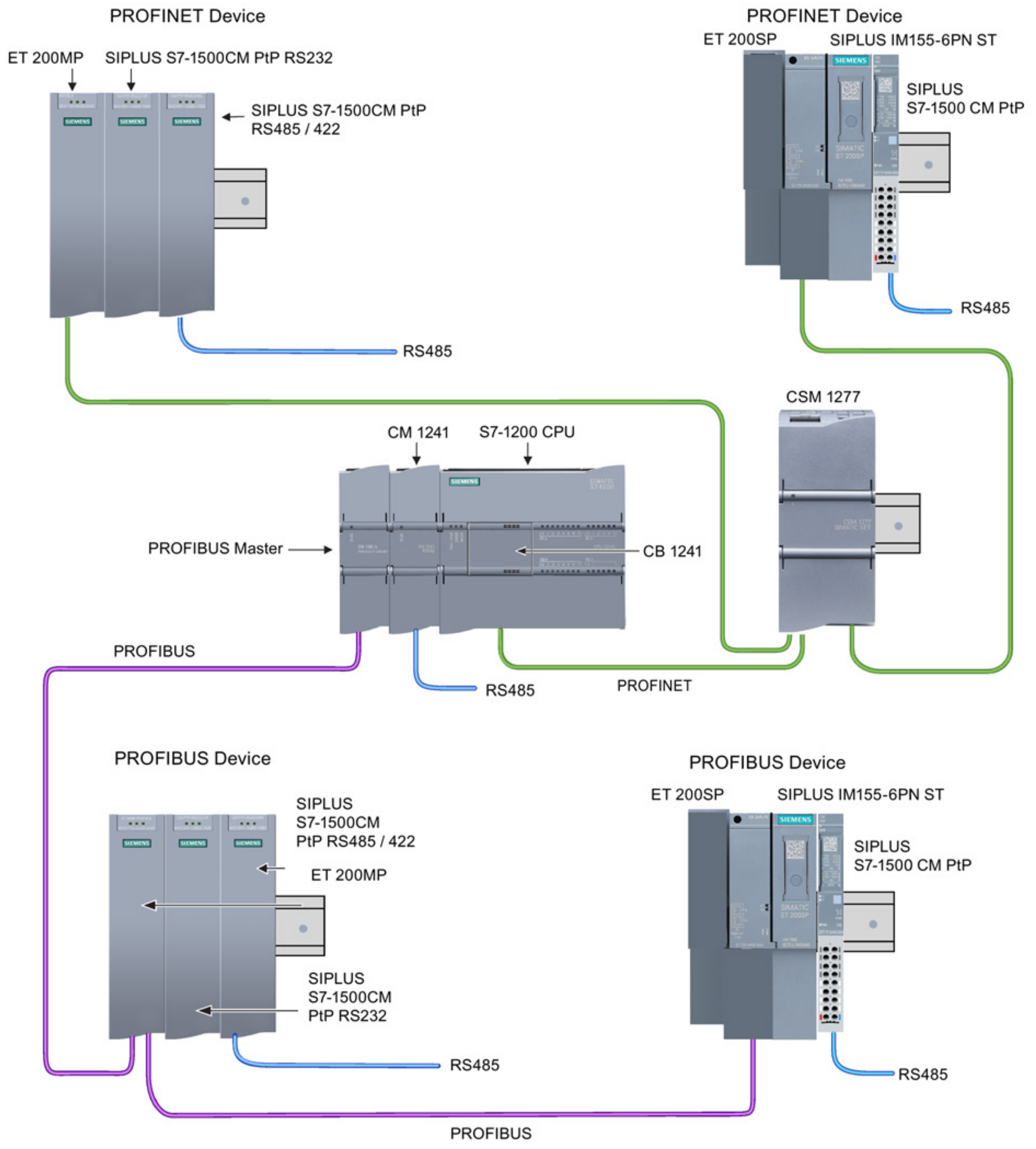
USS communications through PROFIBUS or PROFINET

Version V4.1 of the S7-1200 CPU together with STEP 7 V13 SP1 extends the capability of USS to use a PROFINET or PROFIBUS distributed I/O rack to communicate to various devices (RFID readers, GPS device, and others):

- PROFINET (Page 616): You connect the Ethernet interface of the S7-1200 CPU to a PROFINET interface module. PtP communication modules in the rack with the interface module can then provide serial communications to the PtP devices.
- PROFIBUS (Page 752): You insert a PROFIBUS communication module in the left side of the rack with the S7-1200 CPU. You connect the PROFIBUS communication module to a rack containing a PROFIBUS interface module. PtP communication modules in the rack with the interface module can then provide serial communications to the PtP devices.

For this reason, the S7-1200 supports two sets of PtP instructions:

- Legacy USS instructions (Page 997): These USS instructions existed prior to version V4.0 of the S7-1200 and only work with serial communications using a CM 1241 communication module or CB 1241 communication board.
- USS instructions (Page 918): These USS instructions provide all of the functionality of the legacy instructions, plus the ability to connect to PROFINET and PROFIBUS distributed I/O. The point-to-point instructions allow you to configure the communications between the PtP communication modules in the distributed I/O rack and the PtP devices.



Note

With version V4.1 of the S7-1200, you can use the point-to-point instructions for all types of point-to-point communication: serial, serial over PROFINET, and serial over PROFIBUS. STEP 7 provides the legacy point-to-point instructions only to support existing programs. The legacy instructions still function with all S7-1200 CPUs. You do not have to convert prior programs from one set of instructions to the other.

13.4.2 Selecting the version of the USS instructions

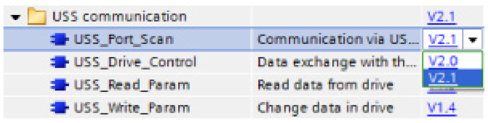
There are two versions of USS instructions available in STEP 7:

- Version 2.0 (legacy instructions) was initially available in STEP 7 Basic/Professional V13.
- Version 2.1 is available in STEP 7 Basic/Professional V13, SP1.

For compatibility and ease of migration, you can choose which instruction version to insert into your user program.

You cannot use both versions of the instructions with the same module, but two different modules can use different versions of the instructions.

Click the icon on the instruction tree task card to enable the headers and columns of the instruction tree.



USS communication		V2.1
USS_Port_Scan	Communication via US...	V2.1
USS_Drive_Control	Data exchange with th...	V2.0
USS_Read_Param	Read data from drive	V2.1
USS_Write_Param	Change data in drive	V1.4

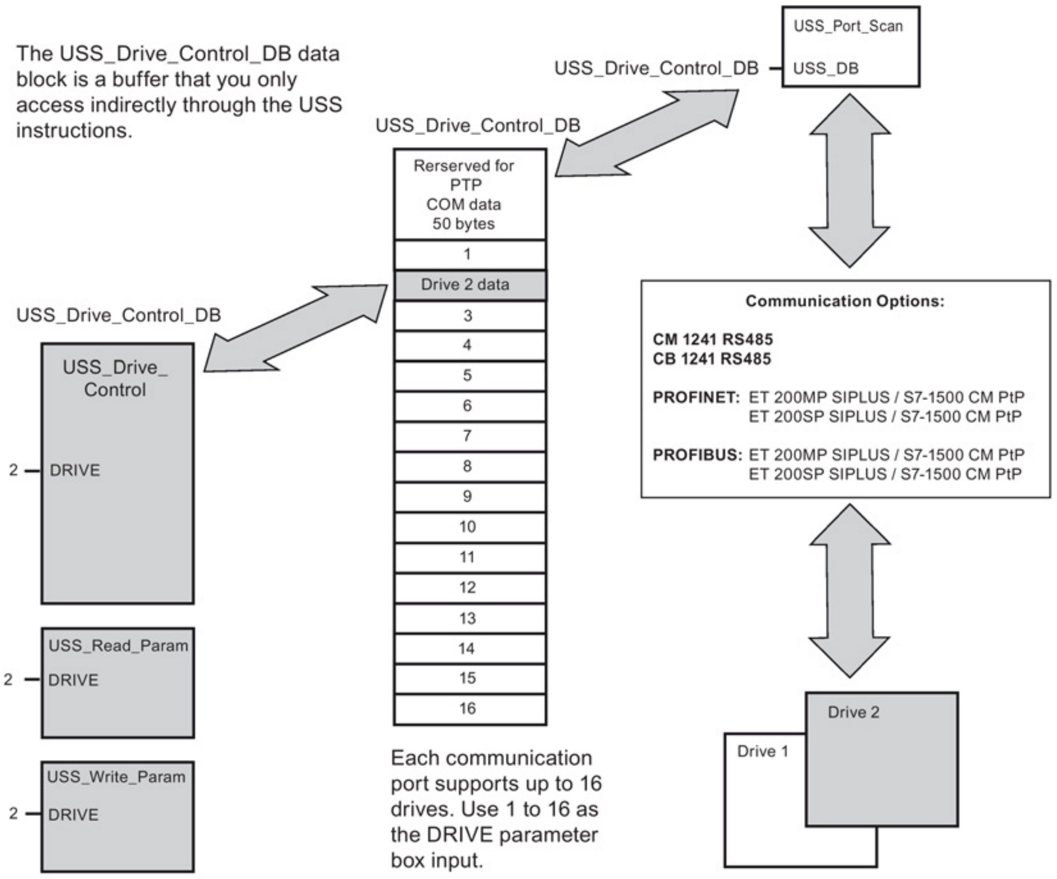
To change the version of the USS instructions, select the version from the drop-down list. You can select the group or individual instructions.

When you use the instruction tree to place a USS instruction in your program, a new FB or FC instance, depending on the USS instruction selected, is created in the project tree. You can see new FB or FC instance in the project tree under PLC_x > Program blocks > System blocks > Program resources.

To verify the version of a USS instruction in a program, you must inspect project tree properties and not the properties of a box displayed in the program editor. Select a project tree USS FB or FC instance, right-click, select "Properties", and select the "Information" page to see the USS instruction version number.

13.4.3 Requirements for using the USS protocol

The four USS instructions use two function blocks (FBs) and two functions (FCs) to support the USS protocol. One USS_Port_Scan instance data block (DB) is used for each USS network. The USS_Port_Scan instance data block contains temporary storage and buffers for all drives on that USS network. The USS instructions share the information in this data block.


The USS_Drive_Control_DB data block is a buffer that you only access indirectly through the USS instructions.
USS_Drive_Control_DB
USS_Port_Scan
USS_DB
USS_Drive_Control_DB
Rerserved for PTP COM data 50 bytes
1
Drive 2 data
3
4
5
6
7
8
9
10
11
12
13
14
15
16
USS_Drive_Control_DB
USS_Drive_ Control
2
DRIVE
USS_Read_Param
2
DRIVE
USS_Write_Param
2
DRIVE
Communication Options:
CM 1241 RS485
CB 1241 RS485
PROFINET: ET 200MP SIPLUS / S7-1500 CM PtP
ET 200SP SIPLUS / S7-1500 CM PtP
PROFIBUS: ET 200MP SIPLUS / S7-1500 CM PtP
ET 200SP SIPLUS / S7-1500 CM PtP
Each communication port supports up to 16 drives. Use 1 to 16 as the DRIVE parameter box input.
Drive 1
Drive 2

All drives (up to 16) connected to a single RS485 port are part of the same USS network. All drives connected to a different RS485 port are part of a different USS network. Each USS network is managed using a unique data block. All instructions associated with a single USS network must share this data block. This includes all USS_Drive_Control, USS_Port_Scan, USS_Read_Param, and USS_Write_Param instructions used to control all drives on a single USS network.

The USS_Drive_Control instruction is a Function Block (FB). When you place the USS_Drive_Control instruction into the program editor, you will be prompted by the "Call options" dialog to assign a DB for this FB. If this is the first USS_Drive_Control instruction in this program for this USS network, then you can accept the default DB assignment (or change the name if you wish) and the new DB is created for you. If however this is not the first USS_Drive_Control instruction for this channel, then you must use the drop-down list in the "Call options" dialog to select the DB name that was previously assigned for this USS network.

The USS_Port_Scan instruction is a Function Block (FB) and handles the actual communication between the CPU and the drives through the Point-to-Point (PtP) RS485 communication port. Each call to this FB handles one communication with one drive. Your program must call this FB fast enough to prevent a communication timeout by the drives. You may call this function FB in a main program cycle OB or any interrupt OB.

The USS_Read_Param, and USS_Write_Param instructions are both Functions (FCs). No DB is assigned when you place these FCs in the editor. Instead, you must assign the appropriate DB reference to the "USS_DB" input of these instructions. Double-click on the parameter field and then click on the parameter helper icon to see the available DB names).

Typically, you should call the USS_Port_Scan FB from a cyclic interrupt OB. The cycle time of the cyclic interrupt OB should be set to about half of the minimum call interval (As an example, 1200 baud communication should use a cyclic time of 350 ms or less).

The USS_Drive_Control FB provides your program access to a specified drive on the USS network. Its inputs and outputs are the status and controls for the drive. If there are 16 drives on the network, your program must have at least 16 USS_Drive_Control calls, one for each drive. These blocks should be called at the rate that is required to control the operation of the drive.

You may only call the USS_Drive_Control FB from a main program cycle OB.

CAUTION

Considerations in calling USS instructions from OBs

Only call USS_Drive_Control, USS_Read_Param, and USS_Write_Param from a main program cycle OB. The USS_Port_Scan FB can be called from any OB, usually from a cyclic interrupt OB.

Do not use instructions USS_Drive_Control, USS_Read_Param, and USS_Write_Param in a higher priority OB than the corresponding USS_Port_Scan instruction. For example, do not place the USS_Port_Scan in the main OB and a USS_Read_Param in a cyclic interrupt OB. Failure to prevent interruption of USS_Port_Scan execution can produce unexpected errors, which could result in personal injury.

The USS_Read_Param, and USS_Write_Param FCs read and write the remote drive operating parameters. These parameters control the internal operation of the drive. See the drive manual for the definition of these parameters. Your program can contain as many of these functions as necessary, but only one read or write request can be active per drive, at any given time. You may only call the USS_Read_Param, and USS_Write_Param FCs from a main program cycle OB.

Calculating the time required for communicating with the drive

Communications with the drive are asynchronous to the S7-1200 scan cycle. The S7-1200 typically completes several scans before one drive communications transaction is completed.

The USS_Port_Scan interval is the time required for one drive transaction. The table below shows the minimum USS_Port_Scan interval for each communication baud rate. Calling the USS_Port_Scan FB more frequently than the USS_Port_Scan interval will not increase the number of transactions. The drive timeout interval is the amount of time that might be taken for a transaction, if communications errors caused 3 tries to complete the transaction. By default, the USS protocol library automatically does up to 2 retries on each transaction.

Table 13- 38 Calculating the time requirements

Baud rate	Calculated minimum USS_Port_Scan call Interval (milliseconds)	Drive message interval timeout per drive (milliseconds)
1200	790	2370
2400	405	1215
4800	212.5	638
9600	116.3	349
19200	68.2	205
38400	44.1	133
57600	36.1	109
115200	28.1	85

13.4.4 USS instructions

13.4.4.1 USS_Port_Scan (Edit communication using USS network) instruction

Table 13- 39 USS_Port_Scan instruction

LAD / FBD	SCL	Description

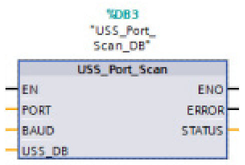
 %DB3
"USS_Port_Scan_DB"
USS_Port_Scan
EN
ENO
PORT
ERROR
BAUD
STATUS
USS_DB
	`USS_Port_Scan(`
	`    PORT:=_uint_in_,`
	`    BAUD:=_dint_in_,`
	`    ERROR=>_bool_out_,`
	`    STATUS=>_word_out_,`
	`    USS_DB:=_fbtref_inout_);`
		The USS_Port_Scan instruction handles communication over a USS network.

Table 13- 40 Data types for the parameters

Parameter and type		Data type	Description
PORT	IN	Port	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table.
BAUD	IN	DInt	The baud rate used for USS communication.
USS_DB	INOUT	USS_BASE	The name of the instance DB that is created and initialized when a USS_Drive_Control instruction is placed in your program.
ERROR	OUT	Bool	When true, this output indicates that an error has occurred and the STATUS output is valid.
STATUS	OUT	Word	The status value of the request indicates the result of the scan or initialization. Additional information is available in the "USS_Extended_Error" variable for some status codes.

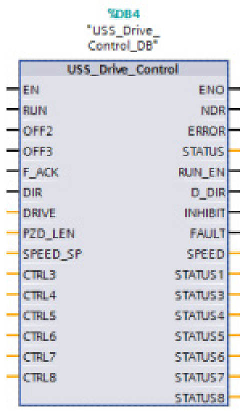
Typically, there is only one USS_Port_Scan instruction per PtP communication port in the program, and each call of this Function Block (FB) handles a transmission to or from a single drive. All USS functions associated with one USS network and PtP communication port must use the same instance DB.

Your program must execute the USS_Port_Scan instruction often enough to prevent drive timeouts. USS_Port_Scan is usually called from a cyclic interrupt OB to prevent drive timeouts and keep the most recent USS data updates available for USS_Drive_Control calls.

13.4.4.2 USS_Drive_Control (Swap data with drive) instruction

Table 13- 41 USS_Drive_Control instruction

LAD / FBD	SCL	Description

 %DB4
"USS_Drive_Control_DB"
USS_Drive_Control
EN
RUN
OFF2
OFF3
F_ACK
DIR
DRIVE
PZD_LEN
SPEED_SP
CTRL3
CTRL4
CTRL5
CTRL6
CTRL7
CTRL8
ENO
NDR
ERROR
STATUS
RUN_EN
D_DIR
INHIBIT
FAULT
SPEED
STATUS1
STATUS3
STATUS4
STATUS5
STATUS6
STATUS7
STATUS8
	`"USS_Drive_Control_DB"(`
	`RUN:=_bool_in_,`
	`OFF2:=_bool_in_,`
	`OFF3:=_bool_in_,`
	`F_ACK:=_bool_in_,`
	`DIR:=_bool_in_,`
	`DRIVE:=_usint_in_,`
	`PZD_LEN:=_usint_in_,`
	`SPEED_SP:=_real_in_,`
	`CTRL3:=_word_in_,`
	`CTRL4:=_word_in_,`
	`CTRL5:=_word_in_,`
	`CTRL6:=_word_in_,`
	`CTRL7:=_word_in_,`
	`CTRL8:=_word_in_,`
	`NDR=>_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_,`
	`RUN_EN=>_bool_out_,`
	`D_DIR=>_bool_out_,`
	`INHIBIT=>_bool_out_,`
	`FAULT=>_bool_out_,`
	`SPEED=>_real_out_,`
	`STATUS1=>_word_out_,`
	`STATUS3=>_word_out_,`
	`STATUS4=>_word_out_,`
	`STATUS5=>_word_out_,`
	`STATUS6=>_word_out_,`
	`STATUS7=>_word_out_,`
	`STATUS8=>_word_out_);`
		The USS_Drive_Control instruction exchanges data with a drive by creating request messages and interpreting the drive response messages. A separate function block should be used for each drive, but all USS functions associated with one USS network and PtP communication port must use the same instance data block. You must create the DB name when you place the first USS_Drive_Control instruction and then reference the DB that was created by the initial instruction usage.
		STEP 7 automatically creates the DB when you insert the instruction.

1 LAD and FBD: Expand the box to reveal all the parameters by clicking the bottom of the box. The parameter pins that are grayed are optional and parameter assignment is not required.

Table 13- 42 Data types for the parameters

Parameter and type		Data type	Description
RUN	IN	Bool	Drive start bit: When true, this input enables the drive to run at the preset speed. When RUN goes to false while a drive is running, the motor will be ramped down to a stop. This behavior differs from the dropping power (OFF2) or braking the motor (OFF3).
OFF2	IN	Bool	Electrical stop bit: When false, this bit cause the drive to coast to a stop with no braking.
OFF3	IN	Bool	Fast stop bit: When false, this bit causes a fast stop by braking the drive rather than just allowing the drive to coast to a stop.
F_ACK	IN	Bool	Fault acknowledge bit: This bit is set to reset the fault bit on a drive. The bit is set after the fault is cleared to indicate to the drive it no longer needs to indicate the previous fault.
DIR	IN	Bool	Drive direction control: This bit is set to indicate that the direction is forward (for positive SPEED_SP).
DRIVE	IN	USInt	Drive address: This input is the address of the USS drive. The valid range is drive 1 to drive 16.
PZD_LEN	IN	USInt	Word length: This is the number of words of PZD data. The valid values are 2, 4, 6, or 8 words. The default value is 2.
SPEED_SP	IN	Real	Speed set point: This is the speed of the drive as a percentage of configured frequency. A positive value specifies forward direction (when DIR is true). Valid range is 200.00 to -200.00.
CTRL3	IN	Word	Control word 3: A value written to a user-configurable parameter on the drive. You must configure this on the drive. (optional parameter)
CTRL4	IN	Word	Control word 4: A value written to a user-configurable parameter on the drive. You must configure this on the drive. (optional parameter)
CTRL5	IN	Word	Control word 5: A value written to a user-configurable parameter on the drive. You must configure this on the drive. (optional parameter)
CTRL6	IN	Word	Control word 6: A value written to a user-configurable parameter on the drive. You must configure this on the drive. (optional parameter)
CTRL7	IN	Word	Control word 7: A value written to a user-configurable parameter on the drive. You must configure this on the drive. (optional parameter)
CTRL8	IN	Word	Control word 8: A value written to a user-configurable parameter on the drive. You must configure this on the drive. (optional parameter)
NDR	OUT	Bool	New data ready: When true, the bit indicates that the outputs contain data from a new communication request.
ERROR	OUT	Bool	Error occurred: When true, this indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported on the USS_Port_Scan instruction ERROR and STATUS outputs.
STATUS	OUT	Word	The status value of the request indicates the result of the scan. This is not a status word returned from the drive.
RUN_EN	OUT	Bool	Run enabled: This bit indicates whether the drive is running.
D_DIR	OUT	Bool	Drive direction: This bit indicates whether the drive is running forward.
INHIBIT	OUT	Bool	Drive inhibited: This bit indicates the state of the inhibit bit on the drive.
FAULT	OUT	Bool	Drive fault: This bit indicates that the drive has registered a fault. You must fix the problem and then set the F_ACK bit to clear this bit when set.

Parameter and type		Data type	Description
SPEED	OUT	Real	Drive Current Speed (scaled value of drive status word 2): The value of the speed of the drive as a percentage of configured speed.
STATUS1	OUT	Word	Drive Status Word 1: This value contains fixed status bits of a drive.
STATUS3	OUT	Word	Drive Status Word 3: This value contains a user-configurable status word on the drive.
STATUS4	OUT	Word	Drive Status Word 4: This value contains a user-configurable status word on the drive.
STATUS5	OUT	Word	Drive Status Word 5: This value contains a user-configurable status word on the drive.
STATUS6	OUT	Word	Drive Status Word 6: This value contains a user-configurable status word on the drive.
STATUS7	OUT	Word	Drive Status Word 7: This value contains a user-configurable status word on the drive.
STATUS8	OUT	Word	Drive Status Word 8: This value contains a user-configurable status word on the drive.

When the initial USS_Drive_Control execution occurs, the drive indicated by the USS address (parameter DRIVE) is initialized in the Instance DB. After this initialization, subsequent executions of USS_Port_Scan can begin communication to the drive at this drive number.

Changing the drive number requires a CPU STOP-to-RUN mode transition that initializes the instance DB. Input parameters are configured into the USS TX message buffer and outputs are read from a "previous" valid response buffer if any exists. There is no data transmission during USS_Drive_Control execution. Drives communicate when USS_Port_Scan is executed. USS_Drive_Control only configures the messages to be sent and interprets data that might have been received from a previous request.

You can control the drive direction of rotation using either the DIR input (Bool) or using the sign (positive or negative) with the SPEED_SP input (Real). The following table indicates how these inputs work together to determine the drive direction, assuming the motor is wired for forward rotation.

Table 13- 43 Interaction of the SPEED_SP and DIR parameters

SPEED_SP	DIR	Drive rotation direction
Value > 0	0	Reverse
Value > 0	1	Forward
Value < 0	0	Forward
Value < 0	1	Reverse

13.4.4.3 USS_Read_Param (Readout parameters from the drive) instruction

Table 13- 44 USS_Read_Param instruction

LAD / FBD	SCL	Description

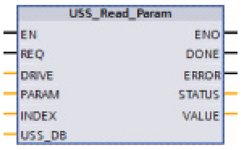
 USS_Read_Param
EN
ENO
REQ
DONE
DRIVE
ERROR
PARAM
STATUS
INDEX
VALUE
USS_DB
	`USS_Read_Param(REQ:=_bool_in_,`
	`DRIVE:=_usint_in_,`
	`PARAM:=_uint_in_,`
	`INDEX:=_uint_in_,`
	`DONE=>_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_,`
	`VALUE=>_variant_out_,`
	`USS_DB:=_fbtref_inout_);`
		The USS_Read_Param instruction reads a parameter from a drive. All USS functions associated with one USS network and PtP communication port must use the same data block. USS_Read_Param must be called from a main program cycle OB.

Table 13- 45 Data types for the parameters

Parameter type		Data type	Description
REQ	IN	Bool	Send request: When true, REQ indicates that a new read request is desired. This is ignored if the request for this parameter is already pending.
DRIVE	IN	USInt	Drive address: DRIVE is the address of the USS drive. The valid range is drive 1 to drive 16.
PARAM	IN	UInt	Parameter number: PARAM designates which drive parameter is written. The range of this parameter is 0 to 2047. On some drives, the most significant byte can access PARAM values greater than 2047. See your drive manual for details on how to access an extended range.
INDEX	IN	UInt	Parameter index: INDEX designates which Drive Parameter index is to be written. A 16-bit value where the Least Significant Byte is the actual index value with a range of (0 to 255). The Most Significant Byte may also be used by the drive and is drive-specific. See your drive manual for details.
USS_DB	INOUT	USS_BASE	The name of the instance DB that is created and initialized when a USS_Drive_Control instruction is placed in your program.
VALUE	IN	Word, Int, UInt, DWord, DInt, UDInt, Real	This is the value of the parameter that was read and is valid only when the DONE bit is true.
DONE[1]	OUT	Bool	When true, indicates that the VALUE output holds the previously requested read parameter value. This bit is set when USS_Drive_Control sees the read response data from the drive. This bit is reset when either: you request the response data using another USS_Read_Param poll, or on the second of the next two calls to USS_Drive_Control.

Parameter type		Data type	Description
ERROR	OUT	Bool	Error occurred: When true, ERROR indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported on the USS_Port_Scan instruction ERROR and STATUS outputs.
STATUS	OUT	Word	STATUS indicates the result of the read request. Additional information is available in the "USS_Extended_Error" variable for some status codes.

1 The DONE bit indicates that valid data has been read from the referenced motor drive and delivered to the CPU. It does not indicate that the USS library is capable of immediately reading another parameter. A blank PKW request must be sent to the motor drive and must also be acknowledged by the instruction before the parameter channel for the specific drive becomes available for use. Immediately calling a USS_Read_Param or USS_Write_Param FC for the specified motor drive will result in a "0x818A" error.

13.4.4.4 USS_Write_Param (Change parameters in the drive) instruction

Note

EEPROM write operations (for the EEPROM inside a USS drive)

Do not overuse the EEPROM permanent write operation. Minimize the number of EEPROM write operations to extend the EEPROM life.

Table 13- 46 USS_Write_Param instruction

LAD / FBD	SCL	Description

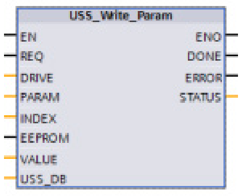
 USS_Write_Param
EN
ENO
REQ
DONE
DRIVE
ERROR
PARAM
STATUS
INDEX
EEPROM
VALUE
USS_DB
	`USS_Write_Param(REQ:=_bool_in_,`
	`DRIVE:=_usint_in_,`
	`PARAM:=_uint_in_,`
	`INDEX:=_uint_in_,`
	`EEPROM:=_bool_in_,`
	`VALUE:=_variant_in_,`
	`DONE=>_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_,`
	`USS_DB:=_fbtref_inout_);`
		The USS_Write_Param instruction modifies a parameter in the drive. All USS functions associated with one USS network and PtP communication port must use the same data block.
		USS_Write_Param must be called from a main program cycle OB.

Table 13- 47 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	Send request: When true, REQ indicates that a new write request is desired. This is ignored if the request for this parameter is already pending.
DRIVE	IN	USInt	Drive address: DRIVE is the address of the USS drive. The valid range is drive 1 to drive 16.
PARAM	IN	UInt	Parameter number: PARAM designates which drive parameter is written. The range of this parameter is 0 to 2047. On some drives, the most significant byte can access PARAM values greater than 2047. See your drive manual for details on how to access an extended range.
INDEX	IN	UInt	Parameter index: INDEX designates which Drive Parameter index is to be written. A 16-bit value where the least significant byte is the actual index value with a range of (0 to 255). The most significant byte may also be used by the drive and is drive-specific. See your drive manual for details.
EEPROM	IN	Bool	Store To Drive EEPROM: When true, a write drive parameter transaction will be stored in the drive EEPROM. If false, the write is temporary and will not be retained if the drive is power cycled.
VALUE	IN	Word, Int, UInt, DWord, DInt, UDInt, Real	The value of the parameter that is to be written. It must be valid on the transition of REQ.
USS_DB	INOUT	USS_BASE	The name of the instance DB that is created and initialized when a USS_Drive_Control instruction is placed in your program.
DONE[1]	OUT	Bool	When true, DONE indicates that the input VALUE has been written to the drive. This bit is set when USS_Drive_Control sees the write response data from the drive. This bit is reset when either you request the response data using another USS_Drive_Control poll, or on the second of the next two calls to USS_Drive_Control.
ERROR	OUT	Bool	When true, ERROR indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported on the USS_Port_Scan instruction ERROR and STATUS outputs.
STATUS	OUT	Word	STATUS indicates the result of the write request. Additional information is available in the "USS_Extended_Error" variable for some status codes.

[1] The DONE bit indicates that valid data has been read from the referenced motor drive and delivered to the CPU. It does not indicate that the USS library is capable of immediately reading another parameter. A blank PKW request must be sent to the motor drive and must also be acknowledged by the instruction before the parameter channel for the specific drive becomes available for use. Immediately calling a USS_Read_Param or USS_Write_Param FC for the specified motor drive will result in a "0x818A" error.

13.4.5 USS status codes

USS instruction status codes are returned at the STATUS output of the USS functions.

Table 13- 48 STATUS codes [1]

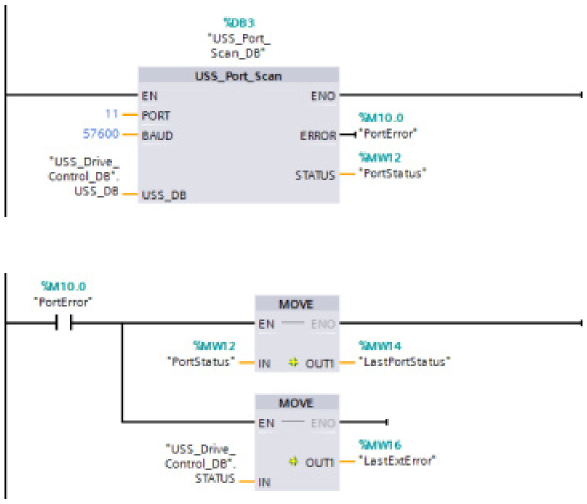
STATUS (W#16#....)	Description
0000	No error
8180	The length of the drive response did not match the characters received from the drive. The drive number where the error occurred is returned in the "USS_Extended_Error" variable. See the extended error description below this table.
8181	VALUE parameter was not a Word, Real or DWord data type.
8182	The user supplied a Word for a parameter value and received a DWord or Real from the drive in the response.
8183	The user supplied a DWord or Real for a parameter value and received a Word from the drive in the response.
8184	The response telegram from drive had a bad checksum. The drive number where the error occurred is returned in the "USS_Extended_Error" variable. See the extended error description below this table.
8185	Illegal drive address (valid drive address range: 1 to16)
8186	The speed set point is out of the valid range (valid speed SP range: -200% to 200%).
8187	The wrong drive number responded to the request sent. The drive number where the error occurred is returned in the "USS_Extended_Error" variable. See the extended error description below this table.
8188	Illegal PZD word length specified (valid range = 2, 4, 6 or 8 words)
8189	Illegal Baud Rate was specified.
818A	The parameter request channel is in use by another request for this drive.
818B	The drive has not responded to requests and retries. The drive number where the error occurred is returned in the "USS_Extended_Error" variable. See the extended error description below this table.
818C	The drive returned an extended error on a parameter request operation. See the extended error description below this table.
818D	The drive returned an illegal access error on a parameter request operation. See your drive manual for information of why parameter access may be limited.
818E	The drive has not been initialized. This error code is returned to USS_Read_Param or USS_Write_Param when USS_Drive_Control, for that drive, has not been called at least once. This keeps the initialization on first scan of USS_Drive_Control from overwriting a pending parameter read or write request, since it initializes the drive as a new entry. To fix this error, call USS_Drive_Control for this drive number.
80Ax-80Fx	Specific errors returned from PtP communication FBs called by the USS Library - These error code values are not modified by the USS library and are defined in the PtP instruction descriptions.

[1] In addition to the USS instruction errors listed above, errors can be returned from the underlying PtP communication instructions (Page 877).

For several STATUS codes, additional information is provided in the "USS_Extended_Error" variable of the USS_Drive_Control Instance DB. For STATUS codes hexadecimal 8180, 8184, 8187, and 818B, USS_Extended_Error contains the drive number where the communication error occurred. For STATUS code hexadecimal 818C, USS_Extended_Error contains a drive error code returned from the drive when using a USS_Read_Param or USS_Write_Param instruction.

Example: Communication errors reporting

Communication errors (STATUS = 16#818B) are only reported on the USS_Port_Scan instruction and not on the USS_Drive_Control instruction. For example, if the network is not properly terminated, then it is possible for a drive to go to RUN but the USS_Drive_Control instruction will show all "0's' for the output parameters. In this case, you can only detect the communication error on the USS_Port_Scan instruction. Since this error is only visible for one scan, you will need to add some capture logic as illustrated in the following example. In this example, when the error bit of the USS_Port_Scan instruction is TRUE, then the STATUS and the USS_Extended_Error values are saved into M memory. The drive number is placed in the USS_Extended_Error variable when the STATUS code value is hexadecimal 8180, 8184, 8187, or 818B.



Network 1"PortStatus"port status and"USS_Drive_Control_DB".USS_Extended_Error
extended error code values are only valid for
one program scan. The values must be
captured for later processing.

Network 2 The "PortError" contact triggers the storage of the "PortStatus" value in "LastPortStatus" and the"USS_Drive_Control_DB".USS_Extended_Error
value in "LastExtError".

Read and write access to a drive's internal parameters

USS drives support read and write access to a drive's internal parameters. This feature allows remote control and configuration of the drive. Drive parameter access operations can fail due to errors such as values out of range or illegal requests for a drive's current mode. The drive generates an error code value that is returned in the "USS_Extended_Error" variable. This error code value is only valid for the last execution of a USS_Read_Param or USS_Write_Param instruction. The drive error code is put into USS_Extended_Error variable when the STATUS code value is hexadecimal 818C. The error code value of USS_Extended_Error depends on the drive model. See the drive's manual for a description of the extended error codes for read and write parameter operations.

13.4.6 USS general drive setup requirements

USS general drive setup requirements consist of the following points:

- The drives must be set to use 4 PKW words.
- The drives can be configured for 2, 4, 6, or 8 PZD words.
- The number of PZD word's in the drive must match PZD_LEN input on the USS_Drive_Control instruction for that drive.
- The baud rate in all the drives must match the BAUD input on the USS_Port_Scan instruction.
- The drive must be set for remote control.
- The drive must be set for frequency set-point to USS on COM Link.
- The drive address must be set to 1 to 16 and match the DRIVE input on the USS_Drive_Control block for that drive.
- The drive direction control must be set to use the polarity of the drive set-point.
- The RS485 network must be terminated properly.

13.4.7 Example: USS general drive connection and setup

Connecting a MicroMaster drive

This information about SIEMENS MicroMaster drives is provided as an example. For other drives, refer to the drive's manual for setup instructions.

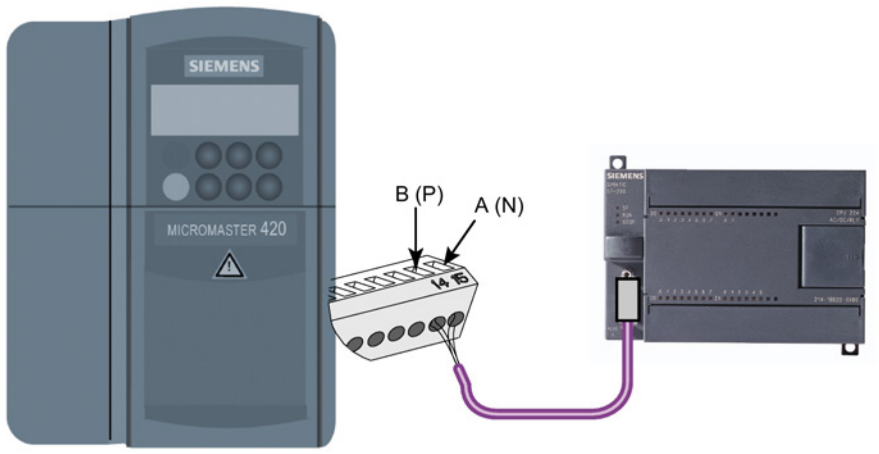
To make the connection to a MicroMaster Series 4 (MM4) drive, insert the ends of the RS485 cable into the two caged-clamp, screw-less terminals provided for USS operation. Standard PROFIBUS cable and connectors can be used to connect the S7-1200.

CAUTION

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. 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. The shield must be tied to chassis ground or pin 1 on the 9-pin connector. It is recommended that you tie wiring terminal 2--0 V on the MicroMaster drive to chassis ground.

The two wires at the opposite end of the RS485 cable must be inserted into the MM4 drive terminal blocks. To make the cable connection on a MM4 drive, remove the drive cover(s) to access the terminal blocks. See the MM4 user manual for details about how to remove the covers(s) of your specific drive.

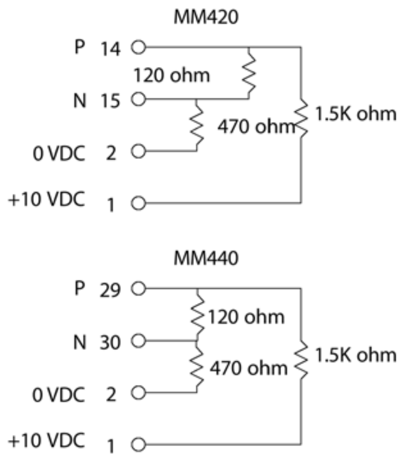


The terminal block connections are labeled numerically. Using a PROFIBUS connector on the S7-1200 side, connect the A terminal of the cable to the drive terminal 15 (for an MM420) or terminal 30 (MM440). Connect the B terminal of B (P) A (N) the cable connector to terminal 14 (MM420) or terminal 29 (MM440).

If the S7-1200 is a terminating node in the network, or if the connection is point-to-point, it is necessary to use terminals A1 and B1 (not A2 and B2) of the connector since they allow the termination settings to be set (for example, with DP connector type 6ES7 972--0BA40--0X40).

CAUTION
Replace drive covers properly before supplying power
Make sure the drive covers are replaced properly before supplying power to the unit.

If the drive is configured as the terminating node in the network, then termination and bias resistors must also be wired to the appropriate terminal connections. This diagram shows examples of the MM4 drive connections necessary for termination and bias.



Setting up the MM4 drive

Before you connect a drive to the S7-1200, you must ensure that the drive has the following system parameters. Use the keypad on the drive to set the parameters:

1. Reset the drive to factory settings (optional).	P0010=30
	P0970=1

If you skip step 1, then ensure that these parameters are set to the indicated values.	USS PZD length = P2012 Index 0=(2, 4, 6, or 8) USS PKW length = P2013 Index 0=4
2. Enable the read/write access to all parameters (Expert mode).	P0003=3

3. Check the motor settings for your drive. The settings will vary according to the motor(s) being used.
To set the parameters P304, P305, P307, P310, and P311, you must first set parameter P010 to 1 (quick commissioning mode). When you are finished setting the parameters, set parameter P010 to 0. Parameters P304, P305, P307, P310, and P311 can only be changed in the quick commissioning mode.
	P0304 = Rated motor voltage (V)
	P0305 = Rated motor current (A)
	P0307 = Rated motor power (W)
	P0310 = Rated motor frequency (Hz)
	P0311 = Rated motor speed

4. Set the local/remote control mode.	P0700 Index 0=5
5. Set selection of frequency set-point to USS on COM link.	P1000 Index 0=5
6. Ramp up time (optional) This is the time in seconds that it takes the motor to accelerate to maximum frequency.	P1120=(0 to 650.00)
7. Ramp down time (optional) This the time in seconds that it takes the motor to decelerate to a complete stop.	P1121=(0 to 650.00)
8. Set the serial link reference frequency:	P2000=(1 to 650 Hz)
9. Set the USS normalization:	P2009 Index 0=0

10. Set the baud rate of the RS485 serial interface:	P2010 Index 0= 4 (2400 baud) 5 (4800 baud) 6 (9600 baud) 7 (19200 baud 8 (38400 baud) 9 (57600 baud) 12 (115200 baud)
11. Enter the Slave address. Each drive (a maximum of 31) can be operated over the bus.	P2011 Index 0=(0 to 31)
12. Set the serial link timeout. This is the maximum permissible period between two incoming data telegrams. This feature is used to turn off the inverter in the event of a communications failure. Timing starts after a valid data telegram has been received. If a further data telegram is not received within the specified time period, the inverter will trip and display fault code F0070. Setting the value to zero switches off the control.	P2014 Index 0=(0 to 65,535 ms) 0=timeout disabled
13. Transfer the data from RAM to EEPROM:	P0971=1 (Start transfer) Save the changes to the parameter settings to EEPROM

13.5 Modbus communication

13.5.1 Overview of Modbus RTU and TCP communication Modbus TCP instructions V13

Modbus function codes

- A CPU operating as a Modbus RTU master (or Modbus TCP client) can read/write both data and I/O states in a remote Modbus RTU slave (or Modbus TCP server). Remote data can be read and then processed in your program logic.
- A CPU operating as a Modbus RTU slave (or Modbus TCP server) allows a supervisory device to read/write both data and I/O states in CPU memory. A RTU master (or Modbus TCP client) can write new values into slave/server CPU memory that is available for your program logic.

WARNING

If an attacker can physically access your networks, the attacker can possibly read and write data.

The TIA Portal, the CPU, and HMIs (except HMIs using GET/PUT) use secure communication that protects against replay and "man-in-the-middle" attacks. Once communication is enabled, the exchange of signed messages takes place in clear text which allows an attacker to read data, but protects against unauthorized writing of data. The TIA Portal, not the communication process, encrypts the data of know-how protected blocks.

All other forms of communication (I/O exchange through PROFIBUS, PROFINET, AS-i, or other I/O bus, GET/PUT, T-Block, and communication modules (CM)) have no security features. You must protect these forms of communication by limiting physical access. If an attacker can physically access your networks utilizing these forms of communication, the attacker can possibly read and write data.

For security information and recommendations, please see our "Operational Guidelines for Industrial Security" (http://www.industry.siemens.com/topics/global/en/industrial-security/Documents/operational_guidelines_industrial_security_en.pdf) in the Siemens Service and Support site.

Table 13- 49 Read data functions: Read remote I/O and program data

Modbus function code	Read slave (server) functions - standard addressing
01	Read output bits: 1 to 2000 bits per request
02	Read input bits: 1 to 2000 bits per request
03	Read Holding registers: 1 to 125 words per request
04	Read input words: 1 to 125 words per request

Table 13- 50 Write data functions: Write remote I/O and modify program data

Modbus function code	Write slave (server) functions - standard addressing
05	Write one output bit: 1 bit per request
06	Write one holding register: 1 word per request
15	Write one or more output bits: 1 to 1968 bits per request
16	Write one or more holding registers: 1 to 123 words per request

- Modbus function codes 08 and 11 provide slave device communication diagnostic information.
- Modbus function code 0 broadcasts a message to all slaves (with no slave response). The broadcast function is not available for Modbus TCP, because communication is connection based.

Table 13- 51 Modbus network station addresses

Station		Address
RTU station	Standard station address	1 to 247
	Extended station address	1 to 65535
TCP station	Station address	IP address and port number

Modbus memory addresses

The actual number of Modbus memory addresses available depends on the CPU model, how much work memory exists, and how much CPU memory is used by other program data. The table below gives the nominal value of the address range.

Table 13- 52 Modbus memory addresses

Station		Address range
RTU station	Standard memory address	10K
	Extended memory address	64K
TCP station	Standard memory address	10K

Modbus RTU communication

Modbus RTU (Remote Terminal Unit) is a standard network communication protocol that uses the RS232 or RS485 electrical connection for serial data transfer between Modbus network devices. You can add PtP (Point to Point) network ports to a CPU with a RS232 or RS485 CM or a RS485 CB.

Modbus RTU uses a master/slave network where all communications are initiated by a single Master device and slaves can only respond to a master’s request. The master sends a request to one slave address and only that slave address responds to the command.

Modbus TCP communication

Modbus TCP (Transmission Control Protocol) is a standard network communication protocol that uses the PROFINET connector on the CPU for TCP/IP communication. No additional communication hardware module is required.

Modbus TCP uses Open User Communications (OUC) connections as a Modbus communication path. Multiple client-server connections may exist, in addition to the connection between STEP 7 and the CPU. Mixed client and server connections are supported up to the maximum number of connections allowed by the CPU model (Page 613).

Each MB_SERVER connection must use a unique instance DB and IP port number. Only 1 connection per IP port is supported. Each MB_SERVER (with its unique instance DB and IP port) must be executed individually for each connection.

A Modbus TCP client (master) must control the client-server connection with the DISCONNECT parameter. The basic Modbus client actions are shown below.

1. Initiate a connection to a particular server (slave) IP address and IP port number
2. Initiate client transmission of a Modbus messages and receive the server responses
3. When desired, initiate the disconnection of client and server to enable connection with a different server.

Modbus RTU instructions in your program

- Modbus_Comm_Load: One execution of Modbus_Comm_Load is used to set up PtP port parameters like baud rate, parity, and flow control. After a CPU port is configured for the Modbus RTU protocol, it can only be used by either the Modbus_Master or Modbus_Slave instructions.
- Modbus_Master: The Modbus_Master instruction enables the CPU to act as a Modbus RTU master device and communicate with one or more Modbus slave devices.
- Modbus_Slave: The Modbus_Slave instruction enables the CPU to act as a Modbus RTU slave device and communicate with a Modbus master device.

Modbus TCP instructions in your program

- MB_CLIENT: Make client-server TCP connection, send command message, receive response, and control the disconnection from the server
- MB_SERVER: Connect to a Modbus TCP client upon request, receive Modbus message, and send response

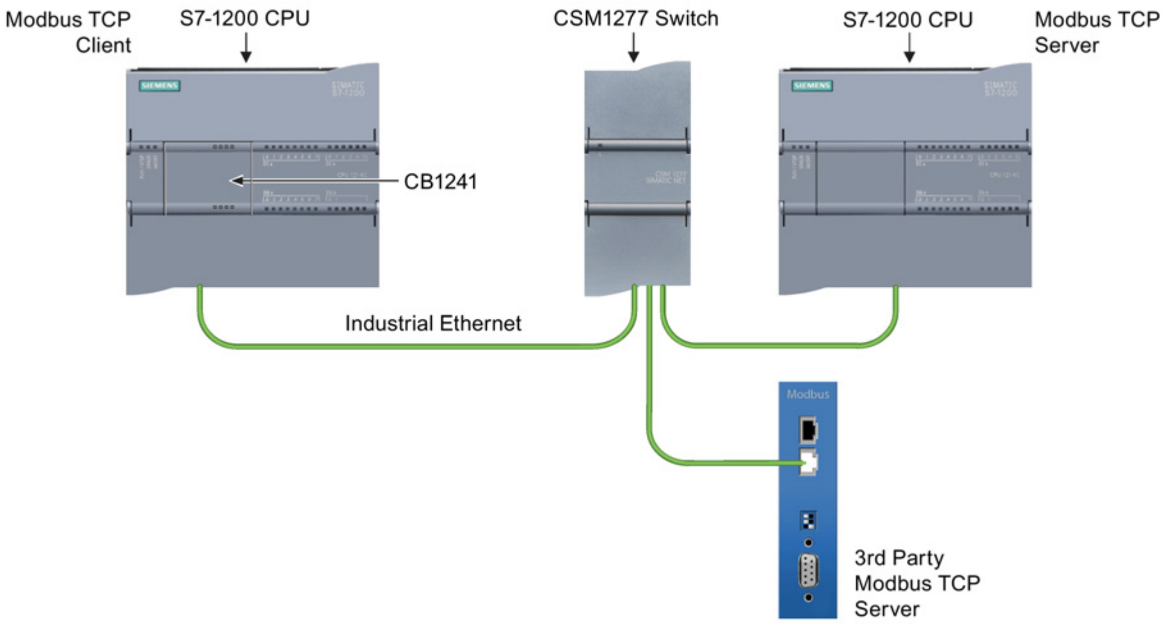
13.5.2 Modbus TCP

13.5.2.1 Overview

Version V4.1 of the S7-1200 CPU together with STEP 7 V13 SP1 extends the capability of Modbus TCP to use enhanced T-block instructions.

For this reason, the S7-1200 supports two sets of PtP instructions:

- Legacy Modbus TCP instructions (Page 1009): These Modbus TCP instructions existed prior to version V4.0 of the S7-1200.
- Modbus TCP instructions (Page 935): These Modbus TCP instructions provide all of the functionality of the legacy instructions.



13.5.2.2 Selecting the version of the Modbus TCP instructions

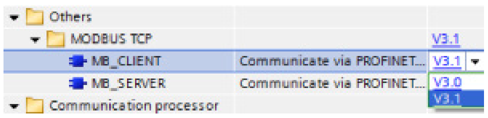
There are two versions of the Modbus TCP instructions available in STEP 7:

- Version 3.0 was initially available in STEP 7 Basic/Professional V13.
- Version 3.1 is available in STEP 7 Basic/Professional V13, SP1.

For compatibility and ease of migration, you can choose which instruction version to insert into your user program.

Do not use both 3.0 and 3.1 instruction versions in the same CPU program. Your program's Modbus TCP instructions must have the same major version number (**1**.x, **2**.y, or **V**.z). The individual instructions within a major version group may have different minor versions (1.**x**).

Click the icon on the instruction tree task card to enable the headers and columns of the instruction tree.



To change the version of the Modbus TCP instructions, select the version from the drop-down list. You can select the group or individual instructions.

When you use the instruction tree to place a Modbus TCP instruction in your program, a new FB instance is created in the project tree. You can see new FB instance in the project tree under PLC_x > Program blocks > System blocks > Program resources.

To verify the version of a Modbus TCP instruction in a program, you must inspect project tree properties and not the properties of a box displayed in the program editor. Select a project tree Modbus TCP FB instance, right-click, select "Properties", and select the "Information" page to see the Modbus TCP instruction version number.

13.5.2.3 Modbus TCP instructions

MB_CLIENT (Communicate using PROFINET as Modbus TCP client) instruction

Table 13- 53 MB_CLIENT instruction

LAD / FBD	SCL	Description

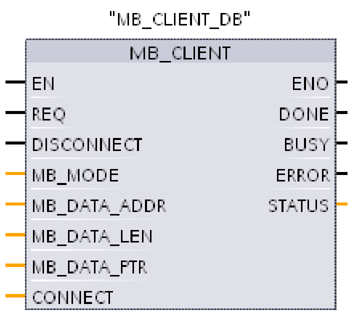
 "MB_CLIENT_DB"
MB_CLIENT
EN
REQ
DISCONNECT
MB_MODE
MB_DATA_ADDR
MB_DATA_LEN
MB_DATA_PTR
CONNECT
ENO
DONE
BUSY
ERROR
STATUS
	`"MB_CLIENT_DB"(`
	`REQ:=_bool_in_,`
	`DISCONNECT:=_bool_in_,`
	`MB_MODE:=_usint_in_,`
	`MB_DATA_ADDR:=_udint_in_,`
	`MB_DATA_LEN:=_uint_in_,`
	`DONE=>_bool_out_,`
	`BUSY=>_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_,`
	`MB_DATA_PTR:=_variant_inout_,`
	`CONNECT:=_variant_inout_);`
		MB_CLIENT communicates as a Modbus TCP client through the PROFINET port on the S7-1200 CPU. No additional communication hardware module is required.
		MB_CLIENT can make a client-server connection, send a Modbus function request, receive a response, and control the disconnection from a Modbus TCP server.

Table 13- 54 Data types for the parameters

Parameter and type		Data type	Description
REQ	In	Bool	FALSE = No Modbus communication request TRUE = Request to communicate with a Modbus TCP server
DISCONNECT	IN	Bool	The DISCONNECT parameter allows your program to control connection and disconnection with a Modbus server device. If DISCONNECT = 0 and a connection does not exist, then MB_CLIENT attempts to make a connection to the assigned IP address and port number. If DISCONNECT = 1 and a connection exists, then a disconnect operation is attempted. Whenever this input is enabled, no other operation will be attempted.
MB_MODE	IN	USInt	Mode selection: Assigns the type of request (read, write, or diagnostic). See the Modbus functions table below for details.
MB_DATA_ADDR	IN	UDInt	Modbus starting Address: Assigns the starting address of the data to be accessed by MB_CLIENT. See the following Modbus functions table for valid addresses.
MB_DATA_LEN	IN	UInt	Modbus data Length: Assigns the number of bits or words to be accessed in this request. See the following Modbus functions table for valid lengths
MB_DATA_PTR	IN_OUT	Variant	Pointer to the Modbus data register: The register buffers data going to or coming from a Modbus server. The pointer must assign a standard global DB or an M memory address.
CONNECT	IN_OUT	Variant	Reference to a Data block structure that contains connection parameters in the system data type "TCON_IP_v4".
DONE	OUT	Bool	The DONE bit is TRUE for one scan, after the last request was completed with no error.
BUSY	OUT	Bool	• 0 - No MB_CLIENT operation in progress • 1 - MB_CLIENT operation in progress
ERROR	OUT	Bool	The ERROR bit is TRUE for one scan, after the MB_CLIENT execution ended with an error. The error code at the STATUS parameter is valid only during the single cycle where ERROR = TRUE.
STATUS	OUT	Word	Execution condition code

Note

CPU firmware version requirement

The Modbus TCP instructions described in this section of the manual require firmware release V4.1 or later.

REQ parameter

FALSE = No Modbus communication request
TRUE = Request to communicate with a Modbus TCP server

If no instance of MB_CLIENT is active and parameter DISCONNECT=0, when REQ=1 a new Modbus request starts. If the connection is not already established, then a new connection is made.

If the same instance of MB_CLIENT is executed again with DISCONNECT=0 and REQ=1, before the completion of the current request, then no subsequent Modbus transmission will be made. However, as soon as the current request is completed, a new request can be processed if MB_CLIENT is executed with REQ=1.

When the current MB_CLIENT communication request is complete, the DONE bit is TRUE for one cycle. The DONE bit can be used as a time gate to sequence multiple MB_CLIENT requests.

Note

Input data consistency during MB_CLIENT processing

Once a Modbus client initiates a Modbus operation, all the input states are saved internally and are then compared on each successive call. The comparison is used to determine if this particular call was the originator of the active client request. More than one MB_CLIENT call can be performed using a common instance DB.

It is important that the inputs are not changed during the period of time that an MB_CLIENT operation is actively being processed. If this rule is not followed, then an MB_CLIENT cannot determine the active instance.

MB_MODE and MB_DATA_ADDR parameters select the Modbus communication function

The MB_CLIENT instruction uses an MB_MODE input rather than a function code. MB_DATA_ADDR assigns the starting Modbus address of the remote data.

The combination of MB_MODE and MB_DATA_ADDR determines the function code that is used in the actual Modbus message. The following table shows the correspondence between parameter MB_MODE, MB_DATA_ADDR, and Modbus function.

Table 13- 55 Modbus functions

MB_MODE	MB_DATA_ADDR	Data length	Modbus function code activated	Operation and data
0	1 to 9999	1 to 2000	01	Read output bits: 1 to 2000 bits per request
0	10001 to 19999	1 to 2000	02	Read input bits: 1 to 2000 bits per request
0	40001 to 49999 or 400001 to 465535	1 to 125	03	Read Holding registers: 1 to 125 words per request
0	30001 to 39999	1 to 125	04	Read input words: 1 to 125 words per request
1	1 to 9999	1	05	Write one output bit: One bit per request
1	40001 to 49999 or 400001 to 465535	1	06	Write one holding register: 1 word per request
1	1 to 9999	2 to 1968	15	Write multiple output bits: 2 to 1968 bits per request
1	40001 to 49999 or 400001 to 465535	2 to 123	16	Write multiple holding registers: 2 to 123 words per request
2	1 to 9999	1 to 1968	15	Write one or more output bits: 1 to 1968 bits per request
2	40001 to 49999 or 400001 to 465535	1 to 123	16	Write one or more holding registers: 1 to 123 words per request
11		0	11	Read the server communication status word and event counter. The status word indicates busy (0 = not busy, 0xFFFF = busy). The event counter is incremented for each successful completion of a message. Both the MB_DATA_ADDR and MB_DATA_LEN parameters of MB_CLIENT are ignored for this function.
80		1	08	Check server status with diagnostic code 0x0000 (Loopback test, server echoes the re-quest) 1 word per request

MB_MODE	MB_DATA_ADDR	Data length	Modbus function code activated	Operation and data
81		1	08	Reset server event counter with diagnostic code 0x000A 1 word per request
3 to 10, 12 to 79, 82 to 255				Reserved

Note

MB_DATA_PTR assigns a buffer to store data read/written to/from a Modbus TCP server

The data buffer can be located in a standard global DB or M memory address.

For a buffer in M memory, use the Any Pointer format. This is in the format P#"Bit Address" "Data Type" "Length", an example would be P#M1000.0 WORD 500.

MB_DATA_PTR parameter assigns a communication buffer

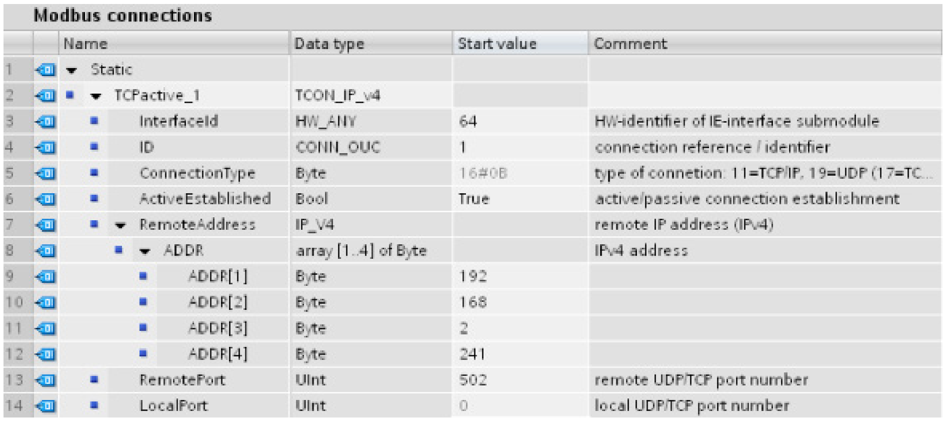
- MB_CLIENT communication functions:
  - Read and write 1-bit data from Modbus server addresses (00001 to 09999)
  - Read 1-bit data from Modbus server addresses (10001 to 19999)
  - Read 16-bit word data from Modbus server addresses (30001 to 39999) and (40001 to 49999)
  - Write 16-bit word data to Modbus server addresses (40001 to 49999)
- Word or bit sized data is transferred to/from the DB or M memory buffer assigned by MB_DATA_PTR.
- If a DB is assigned as the buffer by MB_DATA_PTR, then you must assign data types to all DB data elements.
  - The 1-bit Bool data type represents one Modbus bit address
  - 16-bit single word data types like WORD, UInt, and Int represent one Modbus word address
  - 32-bit double word data types like DWORD, DInt, and Real represent two Modbus word addresses

- Complex DB elements can be assigned by MB_DATA_PTR, such as
  - Arrays
  - Named structures where each element is unique.
  - Named complex structures where each element has a unique name and a 16 or 32 bit data type.
- No requirement that the MB_DATA_PTR data areas be in the same global data block (or M memory area). You can assign one data block for Modbus reads, another data block for Modbus writes, or one data block for each MB_CLIENT.

CONNECT parameter assigns data used to establish a PROFINET connection

You must use a global data block and store the required connection data before you can reference this DB at the CONNECT parameter.

1. Create a new global DB or use an existing global DB to store the CONNECT data. You can use one DB to store multiple TCON_IP_v4 data structures. Each Modbus TCP client or server connection uses a TCON_IP_v4 data structure. You reference the connection data at the CONNECT parameter.
2. Name the DB and a static variable with a helpful name. For example, name the data block "Modbus connections" and a static variable "TCPactive_1" (for Modbus TCP client connection 1).
3. In the DB editor, assign the system data type "TCON_IP_v4" in the Data Type column, for the example static variable "TCPactive_1".
4. Expand the TCON_IP_v4 structure so you can modify the connection parameters, as shown in the following image.
5. Modify data in the TCON_IP_v4 structure for an MB_CLIENT connection.
6. Enter the DB structure reference for the CONNECT parameter of MB_CLIENT. For the example, this would be "Modbus connections".TCPactive_1.



Modbus connections

	Name	Data type	Start value	Comment
1	Static			
2	TCPactive_1	TCON_IP_v4		
3	InterfaceId	HW_ANY	64	HW-identifier of IE-interface submodule
4	ID	CONN_OUC	1	connection reference / identifier
5	ConnectionType	Byte	16#0B	type of connetion: 11=TCP/IP, 19=UDP (17=TC...
6	ActiveEstablished	Bool	True	active/passive connection establishment
7	RemoteAddress	IP_V4		remote IP address (IPv4)
8	ADDR	array [1..4] of Byte		IPv4 address
9	ADDR[1]	Byte	192	
10	ADDR[2]	Byte	168	
11	ADDR[3]	Byte	2	
12	ADDR[4]	Byte	241	
13	RemotePort	UInt	502	remote UDP/TCP port number
14	LocalPort	UInt	0	local UDP/TCP port number

Modify TCP_IP_v4 DB data for each MB_CLIENT connection

- **InterfaceID**: Using the Device configuration window, click on the CPU PROFINET port image. Then click on the General properties tab and use the Hardware identifier that you see there.
- **ID**: Enter a connection ID number between 1 and 4095. Modbus TCP communication is made using underlying TCON, TDISCON, TSEND, and TRCV instructions, for OUC (Open User Communication).
- **ConnectionType**: For TCP/IP, use the default 16#0B (decimal number = 11).
- **ActiveEstablished**: This value is must be 1 or TRUE. The connection is active in that MB_CLIENT initiates Modbus communication.
- **RemoteAddress**: Enter the IP address of the target Modbus TCP server into the four ADDR array elements. For example, enter 192.168.2.241, as in the previous image.
- **RemotePort**: The default is 502. This number is the IP port number of the Modbus server that MB_CLIENT attempts to connect and communicate with. Some third-party Modbus servers require that you use another port number.
- **LocalPort**: This value must be 0, for an MB_CLIENT connection.

Multiple client connections

A Modbus TCP client can support concurrent connections up to the maximum number of Open User Communications connections allowed by the PLC. The total number of connections for a PLC, including Modbus TCP Clients and Servers, must not exceed the maximum number of supported Open User Communications connections (Page 613).

Individual concurrent client connections must follow these rules:

- Each MB_CLIENT connection must use a unique instance DB
- Each MB_CLIENT connection must assign a unique server IP address
- Each MB_CLIENT connection must assign a unique connection ID
- Unique IP port numbers may or may not be required depending upon the server configuration

A different, connection ID must be used with each instance DB. In summary, the instance DB and the connection ID are paired together and must be unique for each connection.

Table 13- 56 MB_CLIENT instance data block: User accessible static variables

Variable	Data type	Default	description
Blocked_Proc_Timeout	Real	3.0	Amount of time (in seconds) to wait upon a blocked Modbus client instance before removing this instance as being ACTIVE. This can occur, for example, when a client request has been issued and then application stops executing the client function before completely finishing the request. The maximum S7-1200 limit is 55 seconds.
MB_Unit_ID	Word	255	Modbus unit identifier: A Modbus TCP server is addressed using its IP address. As a result, the MB_UNIT_ID parameter is not used for Modbus TCP addressing. The MB_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.
RCV_TIMEOUT	Real	2.0	Time in seconds that the MB_CLIENT waits for a server to respond to a request.
Connected	Bool	0	Indicates whether the connection to the assigned server is connected or disconnected: 1=connected, 0=disconnected

Table 13- 57 MB_CLIENT protocol errors

STATUS (W#16#)	Response code to Modbus client (B#16#)	Modbus protocol errors
8381	01	Function code not supported
8382	03	Data length error
8383	02	Data address error or access outside the bounds of the MB_HOLD_REG address area
8384	03	Data value error
8385	03	Data diagnostic code not supported (function code 08)

Table 13- 58 MB_CLIENT execution condition codes [1]

STATUS (W#16#)	MB_CLIENT parameter errors
7001	MB_CLIENT is waiting for a Modbus server response to a connect or disconnect request, on the assigned TCP port. This code is only returned for the first execution of a connect or disconnect operation.
7002	MB_CLIENT is waiting for a Modbus server response to a connect or disconnect request, for the assigned TCP port. This will be returned for any subsequent executions, while waiting for completion of a connect or disconnect operation.
7003	A disconnect operation has successfully completed (Only valid for one PLC scan).

STATUS (W#16#)	MB_CLIENT parameter errors
80C8	The server has not responded in the assigned time. MB_CLIENT must receive a response using the transaction ID that was originally transmitted within the assigned time or this error is returned. Check the connection to the Modbus server device. This error is only returned after retries (if applicable) have been attempted.
8188	Invalid mode
8189	Invalid data address
818A	Invalid data length
818B	Invalid pointer to the DATA_PTR area. This can be the combination of MB_DATA_ADDRESS + MB_DATA_LEN.
818C	Pointer DATA_PTR points to an optimized DB area (must be a standard DB area or M memory area)
8200	The port is busy processing an existing Modbus request.
8380	Received Modbus frame is incorrect or too few bytes have been received.
8387	The assigned Connection ID parameter is different from the ID used for previous requests. There can only be a single Connection ID used within each MB_CLIENT instance DB. This code is also returned as an internal error if the Modbus TCP protocol ID received from a server is not 0.
8388	A Modbus server returned a quantity of data that is different than what was requested. This code applies to Modbus functions 15 or 16 only.

1 In addition to the MB_CLIENT errors listed above, errors can be returned from the underlying T block communication instructions (TCON, TDISCON, TSEND, and TRCV).

MB_SERVER (Communicate using PROFINET as Modbus TCP server) instruction

Table 13- 59 MB_SERVER instruction

LAD / FBD	SCL	Description

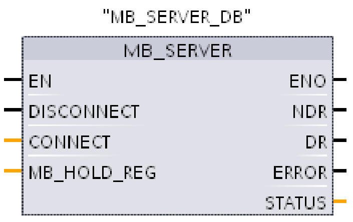
 "MB_SERVER_DB"
MB_SERVER
EN
ENO
DISCONNECT
NDR
CONNECT
DR
MB_HOLD_REG
ERROR
STATUS
	`"MB_SERVER_DB"(`
	`    DISCONNECT:=_bool_in_,`
	`    CONNECT:=_variant_in_,`
	`    NDR=>_bool_out_,`
	`    DR=>_bool_out_,`
	`    ERROR=>_bool_out_,`
	`    STATUS=>_word_out_,`
	
	`MB_HOLD_REG:=_variant_inout_);`
		MB_SERVER communicates as a Modbus TCP server through the PROFINET port on the S7-1200 CPU. No additional communi-cation hardware module is required.
		MB_SERVER can accept a request to con-nect with Modbus TCP client, receive a Modbus function request, and send a re-sponse message.

Table 13- 60 Data types for the parameters

Parameter and type		Data type	Description
DISCONNECT	IN	Bool	MB_SERVER attempts to make a "passive" connection with a partner device. This means that the server is passively listening for a TCP connection request from any requesting IP address. If DISCONNECT = 0 and a connection does not exist, then a passive connection can be initiated. If DISCONNECT = 1 and a connection exists, then a disconnect operation is initiated. This parameter allows your program to control when a connection is accepted. Whenever this input is enabled, no other operation will be attempted.
CONNECT	IN	Variant	Reference to a Data block structure that contains connection parameters in the system data type "TCON_IP_v4".
MB_HOLD_REG	IN_OUT	Variant	Pointer to the MB_SERVER Modbus holding register: The holding register must either be a standard global DB or an M memory address. This memory area is used to hold the data a Modbus client is allowed to access using Modbus register functions 3 (read), 6 (write), and 16 (write).
NDR	OUT	Bool	New Data Ready: 0 = No new data, 1 = Indicates that new data has been written by a Modbus client
DR	OUT	Bool	Data Read: 0 = No data read, 1 = Indicates that data has been read by a Modbus client.
ERROR	OUT	Bool	The ERROR bit is TRUE for one scan, after MB_SERVER execution ended with an error. The error code at the STATUS parameter is valid only during the single cycle where ERROR = TRUE.
STATUS	OUT	Word	Execution condition code

Note

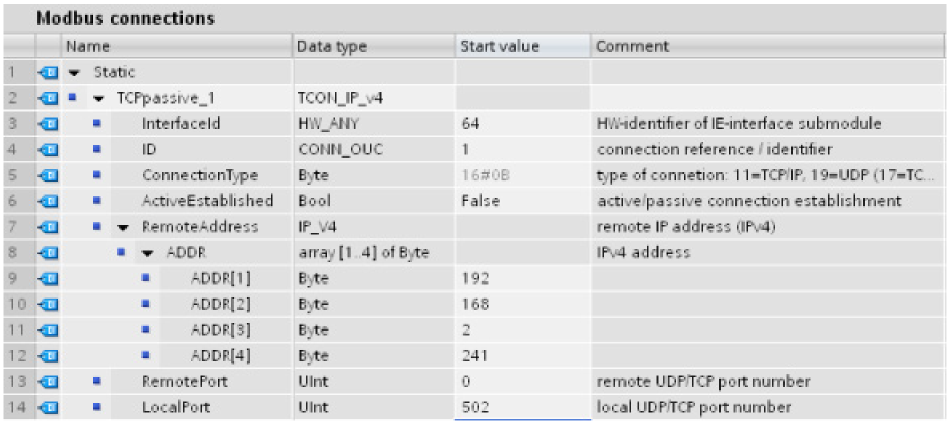
CPU firmware version requirement

The Modbus TCP instructions described in this section of the manual require firmware release V4.1 or later.

CONNECT parameter assigns data used to establish a PROFINET connection

You must use a global data block and store the required connection data before you can reference this DB at the CONNECT parameter.

1. Create a new global DB or use an existing global DB to store the CONNECT data. You can use one DB to store multiple TCON_IP_v4 data structures. Each Modbus TCP client or server connection uses a TCON_IP_v4 data structure. You reference the connection data at the CONNECT parameter.
2. Name the DB and a static variable with a helpful name. For example, name the data block "Modbus connections" and a static variable "TCPpassive_1" (for Modbus TCP server connection 1).
3. In the DB editor, assign the system data type "TCON_IP_v4" in the Data Type column, for the example static variable "TCPactive_1".
4. Expand the TCON_IP_v4 structure so you can modify the connection parameters, as shown in the following image.
5. Modify data in the TCON_IP_v4 structure for an MB_SERVER connection.
6. Enter the DB structure reference for the CONNECT parameter of MB_SEVER. For the example, this would be "Modbus connections".TCPpassive_1.



Modbus connections

	Name	Data type	Start value	Comment
1	Static			
2	TCPpassive_1	TCON_IP_v4		
3	InterfaceId	HW_ANY	64	HW-identifier of IE-interface submodule
4	ID	CONN_OUC	1	connection reference / identifier
5	ConnectionType	Byte	16#0B	type of connetion: 11=TCP/IP, 19=UDP (17=TC...
6	ActiveEstablished	Bool	False	active/passive connection establishment
7	RemoteAddress	IP_V4		remote IP address (IPv4)
8	ADDR	array [1..4] of Byte		IPv4 address
9	ADDR[1]	Byte	192	
10	ADDR[2]	Byte	168	
11	ADDR[3]	Byte	2	
12	ADDR[4]	Byte	241	
13	RemotePort	UInt	0	remote UDP/TCP port number
14	LocalPort	UInt	502	local UDP/TCP port number

Modify TCP_IP_v4 DB data for each MB_SERVER connection

- **InterfaceID**: Using the Device configuration window, click on the CPU PROFINET port image. Then click on the General properties tab and use the Hardware identifier that you see there.
- **ID**: Enter a number between 1 and 4095 that is unique for this connection. Modbus TCP communication is made using underlying TCON, TDISCON, TSEND, and TRCV instructions, for OUC (Open User Communication). Up to eight simultaneous OUC connections are allowed.
- **ConnectionType**: For TCP/IP, use the default 16#0B (decimal value = 11).
- **ActiveEstablished**: This value is must be 0 or FALSE. The connection is passive in that MB_SERVER is waiting for a communication request from a Modbus client.
- **RemoteAddress**: There are two options.
  - Use 0.0.0.0 and MB_CLIENT will respond to a Modbus request from any TCP client
  - Enter the IP address of a target Modbus TCP client and MB_CLIENT only responds to a request originating from this client's IP address. For example, enter 192.168.2.241, as in the previous image.
- **RemotePort**: This value must be 0, for an MB_SERVER connection.
- **LocalPort**: The default is 502. This number is the IP port number of the Modbus client that MB_SERVER attempts to connect and communicate with. Some third-party Modbus clients require another port number.

Modbus and process image addresses

MB_SERVER allows incoming Modbus function codes (1, 2, 4, 5, and 15) to read/write bits/words directly in the input/output process image. For data transfer function codes (3, 6, and 16), the MB_HOLD_REG parameter must be defined as a data type larger than a byte. The following table shows the mapping of Modbus addresses to the process image in the CPU.

Table 13- 61 Mapping of Modbus addresses to the process image

Modbus functions						**S7-1200**	
Codes	Function	Data area	Address range			Data area	CPU address
01	Read bits	Output	1	To	8192	Output Process Image	Q0.0 to Q1023.7
02	Read bits	Input	10001	To	18192	Input Process Image	I0.0 to I1023.7
04	Read words	Input	30001	To	30512	Input Process Image	IW0 to IW1022
05	Write bit	Output	1	To	8192	Output Process Image	Q0.0 to Q1023.7
15	Write bits	Output	1	To	8192	Output Process Image	Q0.0 to Q1023.7

Incoming Modbus message function codes function codes (3, 6, and 16) read/write words in a Modbus holding register, which can be in M memory or a data block. The type of holding register is specified by the MB_HOLD_REG parameter.

Note

MB_HOLD_REG parameter assignment

The Modbus Holding Register can be in a standard global DB or an M memory address.

For A Modbus holding register in M memory, use the Any Pointer format. This is in the format P#"Bit Address" "Data Type" "Length". An example would be P#M1000.0 WORD 500

The following table shows examples of Modbus addresses to holding register mapping used for Modbus function codes 03 (read words), 06 (write word), and 16 (write words). The actual upper limit of DB addresses is determined by the maximum work memory limit and M memory limit, for each CPU model.

Table 13- 62 Mapping examples of Modbus address to CPU memory address

Modbus Address	**MB_HOLD_REG parameter examples**		
	P#M100.0 Word 5	**P#DB10.DBx0.0 Word 5**	**"Recipe".ingredient**
40001	MW100	DB10.DBW0	"Recipe".ingredient[1]
40002	MW102	DB10.DBW2	"Recipe".ingredient[2]
40003	MW104	DB10.DBW4	"Recipe".ingredient[3]
40004	MW106	DB10.DBW6	"Recipe".ingredient[4]
40005	MW108	DB10.DBW8	"Recipe".ingredient[5]

Multiple server connections

Multiple server connections may be created. A single PLC can establish concurrent connections to multiple Modbus TCP clients.

A Modbus TCP server can support concurrent connections up to the maximum number of Open User Communications connections allowed by the PLC. The total number of connections for a PLC, including Modbus TCP Clients and Servers, must not exceed the maximum number of supported Open User Communications connections (Page 613). The Modbus TCP connections may be shared between Client and Server type connections.

Individual concurrent server connection must follow these rules:

- Each MB_SERVER connection must use a unique instance DB.
- Each MB_SERVER connection must assign a unique IP port number. Only 1 connection per port is supported.
- Each MB_SERVER connection must assign a unique connection ID.
- The MB_SERVER must be called individually for each connection (with its respective instance DB).

The connection ID must be unique for each individual connection. A single, connection ID must be used with each individual instance DB. The instance DB and the connection ID are paired together and must be unique for every connection.

Table 13- 63 Modbus diagnostic function codes

MB_SERVER Modbus diagnostic functions		
Codes	Sub-function	Description
08	0x0000	Return query data echo test: The MB_SERVER will echo back to a Modbus client a data word that is received.
08	0x000A	Clear communication event counter: The MB_SEVER will clear the communication event counter that is used for Modbus function 11.
11		Get communication event counter: The MB_SERVER uses an internal communication event counter for recording the number of successful Modbus read and write requests that are sent to the Modbus server. The counter does not increment on any request for Function 8, Function 11, or any request that results in a communication error. The broadcast function is not available for Modbus TCP, because only one client-server connection exists at any one time.

MB_SERVER variables

This table shows the public static variables that are stored in the MB_SERVER instance data block and can be used in your program

Table 13- 64 MB_SERVER public static variables

Variable	**Data type**	**Default**	**Description**
HR_Start_Offset	Word	0	Assigns the starting address of the Modbus Holding register
Request_Count	Word	0	The number of all requests received by this server.
Server_Message_Count	Word	0	The number of requests received for this specific server.
Xmt_Rcv_Count	Word	0	The number of transmissions or receptions that have encountered an error. Also, incremented if a message is received that is an invalid Modbus message.
Exception_Count	Word	0	Modbus specific errors that require a returned exception
Success_Count	Word	0	The number of requests received for this specific server that ha no protocol errors.
Connected	Bool	0	Indicates whether the connection to the assigned client is connected or disconnected: 1=connected, 0=disconnected

Your program can write data to the HR_Start_Offset and control Modbus server operations. The other variables can be read to monitor Modbus status.

HR_Start_Offset

Modbus holding register addresses begin at 40001 . These addresses correspond to the beginning PLC memory address of the holding register. However, you can use the "HR_Start_Offset" variable to start the beginning Modbus holding register address at another number instead of 40001.

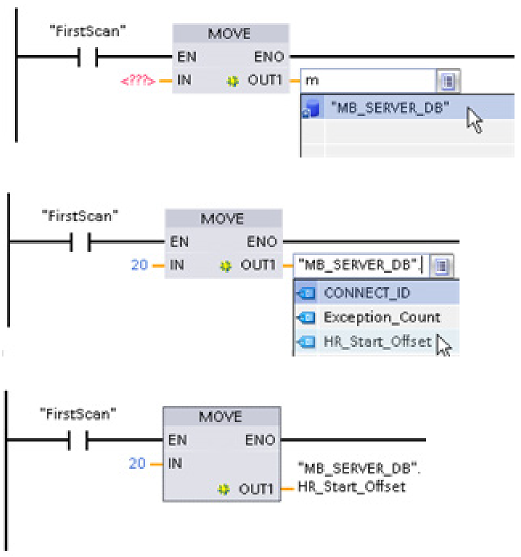
For example, if the holding register starts at MW100 and is 100 words long. An offset of 20 specifies a beginning holding register address of 40021 instead of 40001. Any address less than 40021 or greater than 40119 results in an addressing error.

Table 13- 65 Example of Modbus holding register addressing

HR_Start_Offset	Address	Minimum	Maximum
0	Modbus address (Word)	40001	40099
	S7-1200 address	MW100	MW298
20	Modbus address (Word)	40021	40119
	S7-1200 address	MW100	MW298

HR_Start_Offset is word data in the MB_SERVER instance data block that assigns the starting address of the Modbus holding register. You can set this public static variable by using the parameter helper drop list, after MB_SERVER is placed in your program.

For example, after you place MB_SERVER in a LAD network, you can go to a previous network and assign HR_Start_Offset. The start address must be assigned prior to execution of MB_SERVER.



Entering a Modbus server variable using the default DB name:

1. Set the cursor in the parameter field and type an m character.
2. Select "MB_SERVER_DB" from the drop list of DB names.
3. Select "MB_SERVER_DB.HR_Start_Offset" from the drop list of DB variables.

Table 13- 66 MB_SERVER execution condition codes [1]

STATUS (W#16#)	Response code to Modbus server (B#16#)	Modbus protocol errors
7001		MB_SERVER is waiting for a Modbus client to connect to the assigned TCP port. This code is returned on the first execution of a connect or disconnect operation.
7002		MB_SERVER is waiting for a Modbus client to connect to the assigned TCP port. This code is returned for any subsequent executions, while waiting for completion of a connect or disconnect operation.
7003		A disconnect operation has successfully completed (Only valid for one PLC scan).
8187		Invalid pointer to MB_HOLD_REG: area is too small
818C		Pointer MB_HOLD_REG points to an optimized DB area (must be a standard global DB area or M memory area) or Blocked process timeout exceeds the limit of 55 seconds. (S7-1200 specific)
8381	01	Function code not supported
8382	03	Data length error
8383	02	Data address error or access outside the bounds of the MB_HOLD_REG address area
8384	03	Data value error
8385	03	Data diagnostic code not supported (function code 08)

[1] In addition to the MB_SERVER errors listed above, errors can be returned from the underlying T block communication instructions (TCON, TDISCON, TSEND, and TRCV).

13.5.2.4 Modbus TCP examples

Example: MB_SERVER Multiple TCP connections

You can have multiple Modbus TCP server connections. To accomplish this, MB_SERVER must be independently executed for each connection. Each connection must use an independent instance DB, connection ID, and IP port. The S7-1200 allows only one connection per IP port.

For best performance, MB_SERVER should be executed every program cycle, for each connection.

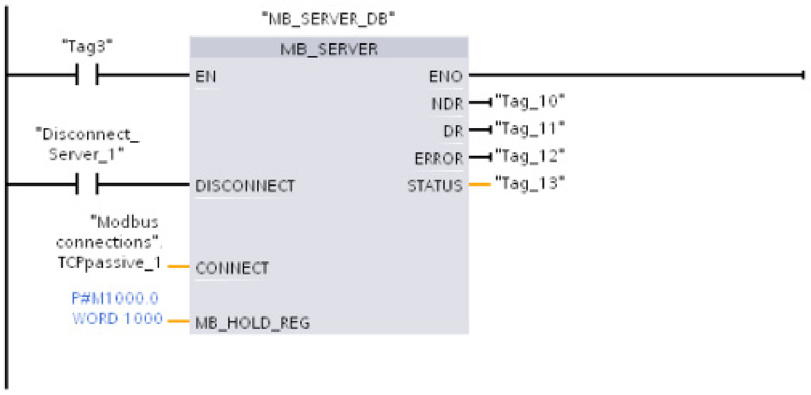
The CONNECT parameter uses system data type TCP_IP_v4.For the example, these data structures are in a DB named "Modbus connections". The "Modbus connections" DB contains two TCP_IP_v4 structures "TCPpassive_1" (for connection 1) and "TCP_passive_2" (for connection 2).The connection properties ID and LocalPort described in the network comments are data elements stored in the CONNECT data structure.

The TCP_IP_v4 CONNECT data also contains an IP address in the RemoteAddress ADDR array. IP address assignments within TCPpassive_1 and TCP_passive_2 do not affect the establishment of TCP server connections, but determine which Modbus TCP clients are allowed to communicate though the connections to each MB_SERVER. MB_SERVER passively listens for a modbus client message and compares the incoming message IP address with the IP address stored in the corresponding RemoteAddress ADDR array.

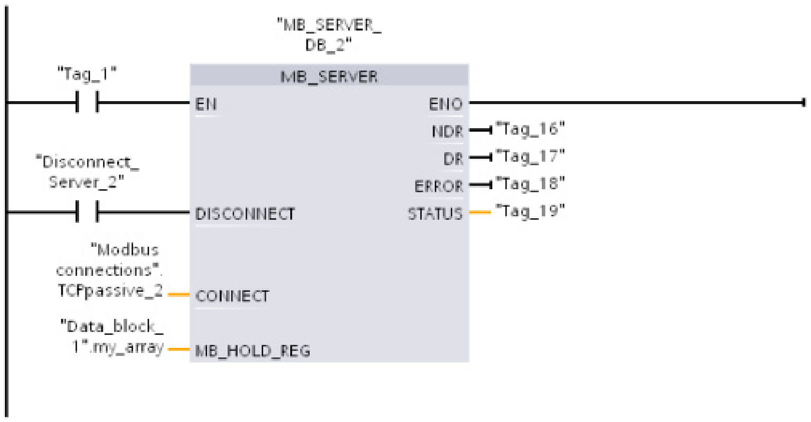
Three MB_SERVER IP address variations are possible for the two MB_SERVER instructions:

- **IP address = 0.0.0.0**
  Each MB_SERVER will respond to all Modbus TCP clients using any IP address.
- **IP address = Same IP address in TCPpassive_1 and TCPpassive_2**
  Both MB_SERVER connections only respond to Modbus clients originating from this IP address.
- **IP address = Different IP number in TCP_passive_1 and TCP_passive_2**
  Each MB_SERVER only responds to Modbus clients that originate from the IP address stored in their TCP_IP_v4 data.

Network 1: Connection #1, Instance DB = "MB_SERVER_DB", within "Modbus connections.TCPpassive_1" (ID = 1 and LocalPort = 502)



Network 2: Connection #2, Instance DB = "MB_SERVER_DB_1, within "Modbus connections.TCPpassive_2" (ID = 2 and LocalPort = 503)



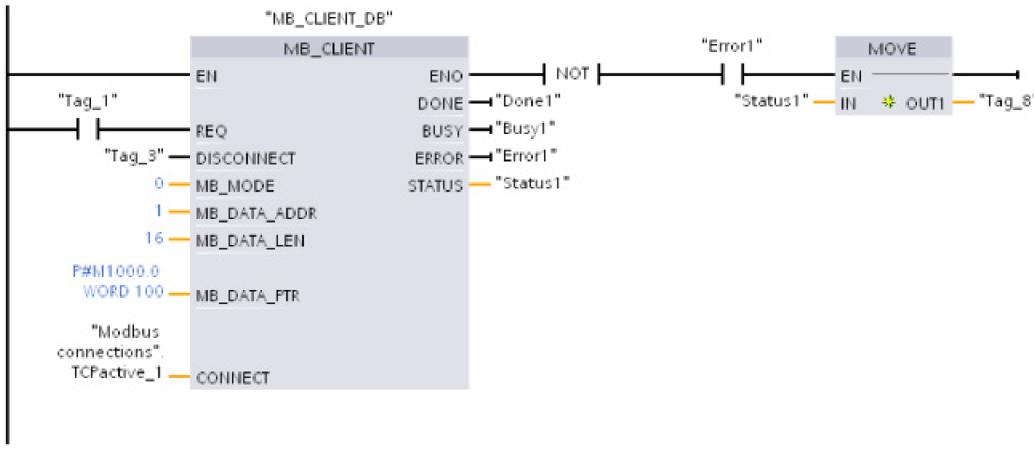
Example: MB_CLIENT 1: Multiple requests with common TCP connection

Multiple Modbus client requests can be sent over the same connection. To accomplish this, use the same instance DB, connection ID, and port number.

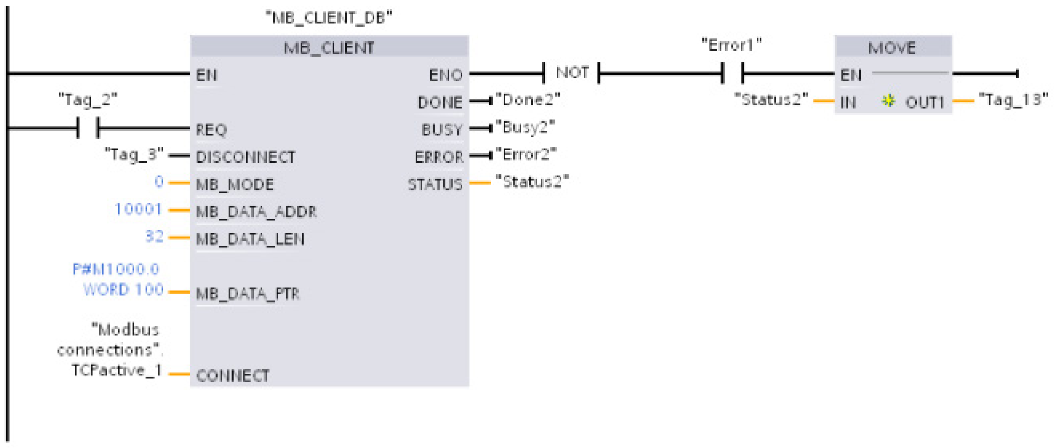
Because both MB_CLIENT boxes use the same CONNECT parameter TCON_IP_v4 data structure ("Modbus_connections".TCPactive_1), the connection ID, port number, and IP address are identical. The CONNECT IP address data assigns the IP address, of the target Modbus TCP server.

Only one MB_CLIENT can be active at any given time. Once a client completes its execution, the next client can begin execution. Your program logic is responsible for the execution sequence logic. The example shows both clients reading remote data from a single Modbus client and transferring the data to the Modbus client's CPU (M memory starting at M1000.0). A returned error is captured, which is optional.

Network 1: Modbus function 1 - Read 16 output bits from a Modbus TCP server with the IP address assigned in "Modbus connections".TCPactive_1.



Network 2: Modbus function 2 - Read 32 input bits from a Modbus TCP server with the IP address assigned in "Modbus connections".TCPactive_1.



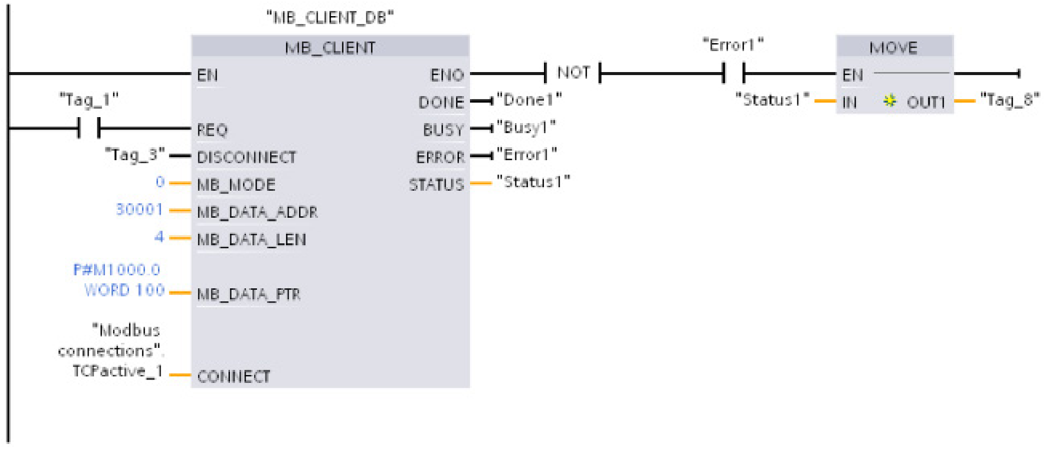
Example: MB_CLIENT 2: Multiple requests with different TCP connections

Modbus TCP client requests can be sent over different connections. To accomplish this, different instance DBs and connection IDs must be used.

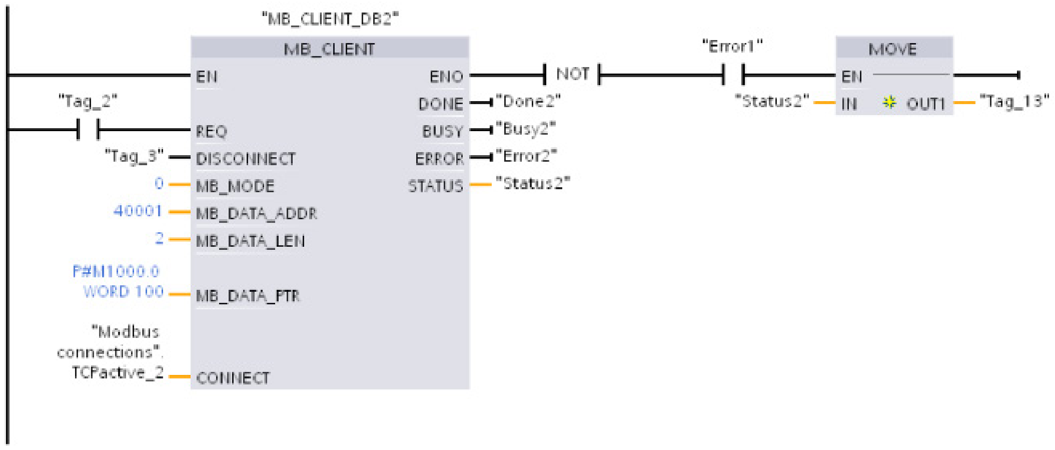
The RemotePort (IP port) number must be different, if the connections are established to the same Modbus server. If the connections are on different servers, there is no IP port number restriction.

The example shows two Modbus TCP clients transferring remote data from two different Modbus TCP servers to the same local CPU memory area, starting at address M1000.0. Also, a returned error is captured which is optional.

Network 1: Modbus function 4 - Read input process image words from a Modbus TCP server CONNECT parameter = "Modbus connections".TCPactive_1: Connection **ID** = 1, **RemoteAddress** = 192.168.2.241, **RemotePort** = 502



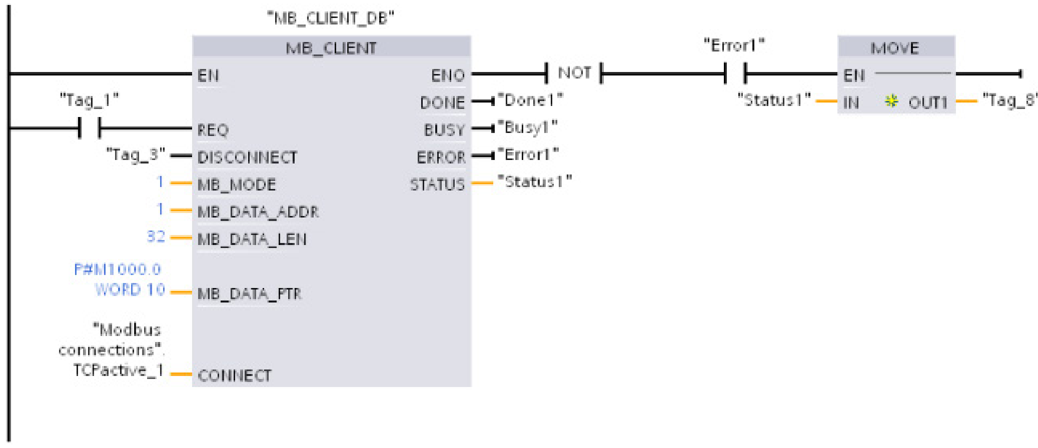
Network 2: Modbus function 3 - Read holding register words from a Modbus TCP server CONNECT parameter = "Modbus connections".TCPactive_2: Connection **ID** = 2, **RemoteAddress** = 192.168.2.242, **RemotePort** = 502



Example: MB_CLIENT 3: Output image write request

This example shows a Modbus client request that transfers bit data from local CPU memory (starting at M1000.0) to a remote Modbus TCP server.

Network 1: Modbus function 15 - Write output bits in a Modbus server

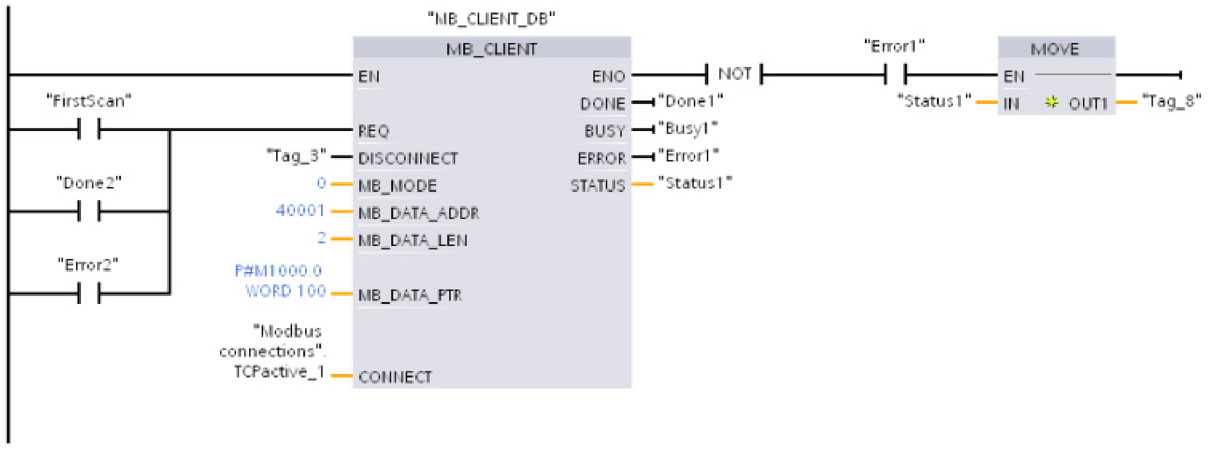


Example: MB_CLIENT 4: Coordinating multiple requests

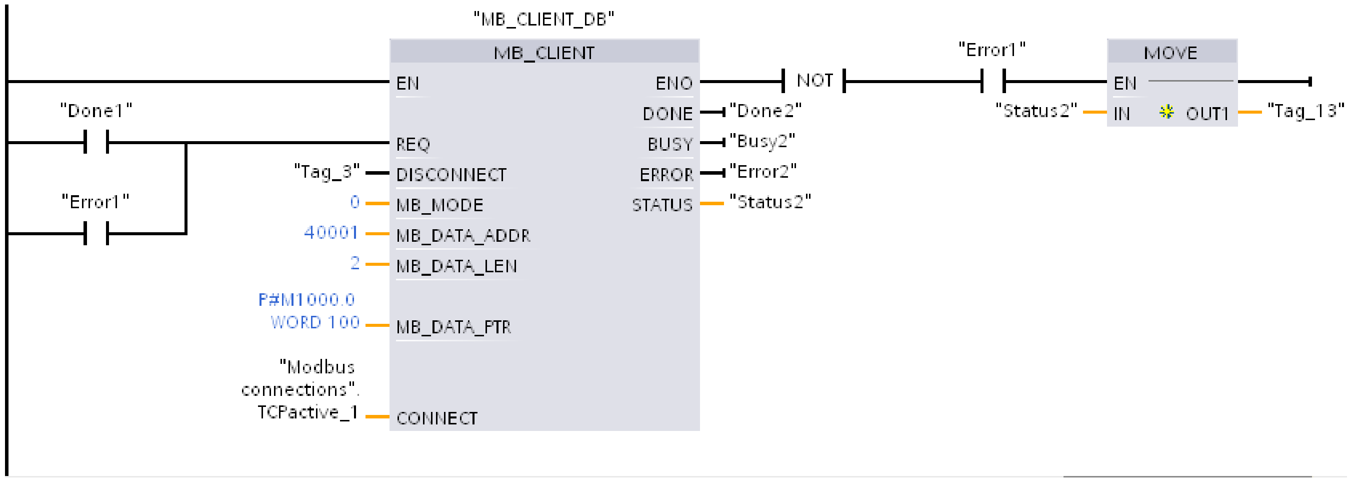
You must ensure that each individual Modbus TCP request finishes execution. The execution sequence must be controlled by your program logic. The example below shows how the outputs of the first and second client requests can control the execution sequence.

The example shows both clients using the same CONNECT connection data (used at different times). The clients transfer holding register data from the same remote Modbus TCP server to the same local CPU memory M address. Also, a returned error is captured which is optional.

Network 1: Modbus function 3 - Read Modbus TCP server holding register words



Network 2: Modbus function 3 - Read Modbus TCP server holding register words



13.5.3 Modbus RTU

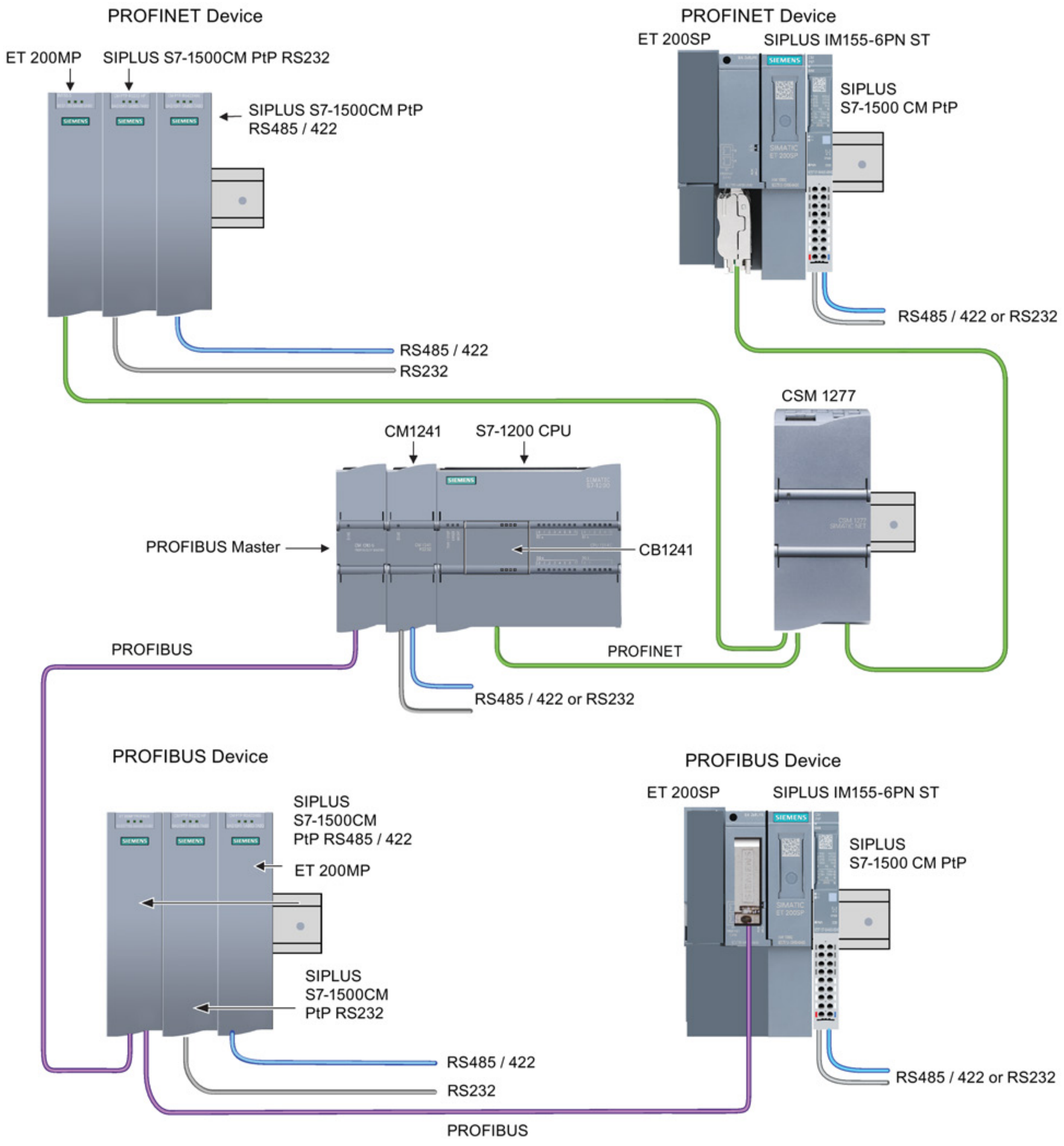
13.5.3.1 Overview

Version V4.1 of the S7-1200 CPU together with STEP 7 V13 SP1 extends the capability of Modbus RTU to use a PROFINET or PROFIBUS distributed I/O rack to communicate to various devices (RFID readers, GPS device, and others):

- PROFINET (Page 616): You connect the Ethernet interface of the S7-1200 CPU to a PROFINET interface module. PtP communication modules in the rack with the interface module can then provide serial communications to the PtP devices.
- PROFIBUS (Page 752): You insert a PROFIBUS communication module in the left side of the rack with the S7-1200 CPU. You connect the PROFIBUS communication module to a rack containing a PROFIBUS interface module. PtP communication modules in the rack with the interface module can then provide serial communications to the PtP devices.

For this reason, the S7-1200 supports two sets of PtP instructions:

- Legacy Modbus RTU instructions (Page 1027): These Modbus RTU instructions existed prior to version V4.0 of the S7-1200 and only work with serial communications using a CM 1241 communication module or CB 1241 communication board.
- Modbus RTU instructions (Page 958): These Modbus RTU instructions provide all of the functionality of the legacy instructions, plus the ability to connect to PROFINET and PROFIBUS distributed I/O. The point-to-point instructions allow you to configure the communications between the PtP communication modules in the distributed I/O rack and the PtP devices.



Note

With version V4.1 of the S7-1200, you can use the point-to-point instructions for all types of point-to-point communication: serial, serial over PROFINET, and serial over PROFIBUS. STEP 7 provides the legacy point-to-point instructions only to support existing programs. The legacy instructions still function, however, with V4.1 CPUs as well as V4.0 and earlier CPUs. You do not have to convert prior programs from one set of instructions to the other.

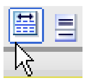
13.5.3.2 Selecting the version of the Modbus RTU instructions

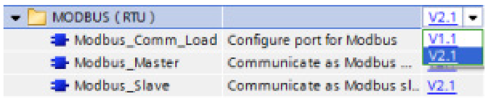
There are two versions of the Modbus RTU instructions available in STEP 7:

- Version 1.1 was initially available in STEP 7 Basic/Professional V13.
- Version 2.1 is available in STEP 7 Basic/Professional V13, SP1.

For compatibility and ease of migration, you can choose which instruction version to insert into your user program.

You cannot use both versions of the instructions with the same module, but two different modules can use different versions of the instructions. Do not use both 1.x and 2.y instruction versions in the same CPU program. Your program's Modbus RTU instructions must have the same major version number (**1**.x, **2**.y, or **V**.z). The individual instructions within a major version group may have different minor versions (1.**x**).


Click the icon on the instruction tree task card to enable the headers and columns of the instruction tree.



```
▼ MODBUS (RTU)                                        V2.1 ▼
   Modbus_Comm_Load   Configure port for Modbus       V1.1
   Modbus_Master      Communicate as Modbus ...       V2.1
   Modbus_Slave       Communicate as Modbus sl...     V2.1
```

To change the version of the Modbus RTU instructions, select the version from the drop-down list. You can select the group or individual instructions.

When you use the instruction tree to place a Modbus RTU instruction in your program, a new FB instance is created in the project tree. You can see new FB instance in the project tree under PLC_x > Program blocks > System blocks > Program resources.

To verify the version of a Modbus RTU instruction in a program, you must inspect project tree properties and not the properties of a box displayed in the program editor. Select a project tree Modbus RTU FB instance, right-click, select "Properties", and select the "Information" page to see the Modbus RTU instruction version number.

13.5.3.3 Modbus RTU instructions

Modbus_Comm_Load (Configure SIPLUS I/O or port on the PtP module for Modbus RTU) instruction

Table 13- 67 Modbus_Comm_Load instruction

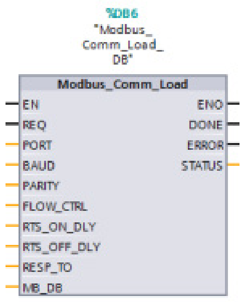
LAD / FBD	SCL	Description
 %DB6 "Modbus_Comm_Load_DB" Modbus_Comm_Load EN REQ PORT BAUD PARITY FLOW_CTRL RTS_ON_DLY RTS_OFF_DLY RESP_TO MB_DB ENO DONE ERROR STATUS	`"Modbus_Comm_Load_DB"(` `REQ:=_bool_in,` `PORT:=_uint_in_,` `BAUD:=_udint_in_,` `PARITY:=_uint_in_,` `FLOW_CTRL:=_uint_in_,` `RTS_ON_DLY:=_uint_in_,` `RTS_OFF_DLY:=_uint_in_,` `RESP_TO:=_uint_in_,` `DONE=>_bool_out,` `ERROR=>_bool_out_,` `STATUS=>_word_out_,` `MB_DB:=_fbtref_inout_);`	The Modbus_Comm_Load instruction configures SIPLUS I/O or a PtP port for Modbus RTU protocol communications. Modbus RTU port hardware options: Install up to three CMs (RS485 or RS232), plus one CB (R4845). Modbus RTU SIPLUS I/O options: Install ET 200MP S7-1500CM PtP (RS485 / 422 or RS232) or ET 200SP S7-1500 CM PtP (RS485 / 422 or RS232) An instance data block is assigned automatically when you place the Modbus_Comm_Load instruction in your program.

Table 13- 68 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	A low to high (positive edge) signal starts the operation. (Version 2.0 only)
PORT	IN	Port	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table.
BAUD	IN	UDInt	Baud rate selection: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200, all other values are invalid
PARITY	IN	UInt	Parity selection: • 0 – None • 1 – Odd • 2 – Even
FLOW_CTRL [1]	IN	UInt	Flow control selection: • 0 – (default) no flow control • 1 – Hardware flow control with RTS always ON (does not apply to RS485 ports) • 2 – Hardware flow control with RTS switched

Parameter and type		Data type	Description
RTS_ON_DLY [1]	IN	UInt	RTS ON delay selection: • 0 – (default) No delay from RTS active until the first character of the message is transmitted • 1 to 65535 – Delay in milliseconds from RTS active until the first character of the message is transmitted (does not apply to RS485 ports). RTS delays shall be applied independent of the FLOW_CTRL selection.
RTS_OFF_DLY [1]	IN	UInt	RTS OFF delay selection: • 0 – (default) No delay from the last character transmitted until RTS goes inactive • 1 to 65535 – Delay in milliseconds from the last character transmitted until RTS goes inactive (does not apply to RS485 ports). RTS delays shall be applied independent of the FLOW_CTRL selection.
RESP_TO [1]	IN	UInt	Response timeout: Time in milliseconds allowed by the Modbus_Master for the slave to respond. If the slave does not respond in this time period, the Modbus_Master will retry the request or terminate the request with an error when the specified number of retries has been sent. 5 ms to 65535 ms (default value = 1000 ms).
MB_DB	IN	Variant	A reference to the instance data block used by the Modbus_Master or Modbus_Slave instructions. After Modbus_Master or Modbus_Slave is placed in your program, the DB identifier appears in the parameter helper drop-list available at the MB_DB box connection.
DONE	OUT	Bool	The DONE bit is TRUE for one scan, after the last request was completed with no error. (Version 2.0 only)
ERROR	OUT	Bool	The ERROR bit is TRUE for one scan, after the last request was terminated with an error. The error code value at the STATUS parameter is valid only during the single scan where ERROR = TRUE.
STATUS	OUT	Word	Execution condition code

[1] Optional parameters for Modbus_Comm_Load (V 2.x or later). Click the arrow at the bottom of a LAD/FBD box to expand the box and include these parameters.

Modbus_Comm_Load is executed to configure a port for the Modbus RTU protocol. Once a port is configured for the Modbus RTU protocol, it can only be used by either the Modbus_Master or Modbus_Slave instructions.

One execution of Modbus_Comm_Load must be used to configure each communication port that is used for Modbus communication. Assign a unique Modbus_Comm_Load instance DB for each port that you use. You can install up to three communication modules (RS232 or RS485) and one communication board (RS485) in the CPU. Call Modbus_Comm_Load from a startup OB and execute it one time or use the first scan system flag (Page 103) to initiate the call to execute it one time. Only execute Modbus_Comm_Load again if communication parameters like baud rate or parity must change.

If you use the Modbus library with a module in a distributed rack, the Modbus_Comm_Load instruction must be executed in a cyclical interrupt routine (for example, once per second or once every 10 seconds). If power is lost to the distributed rack or the module is pulled, upon restoration of module operation, only the HWConfig parameter set is sent to the PtP module. All requests initiated by the Modbus_Master timeout, and the Modbus_Slave goes silent (no response to any message). Cyclic execution of the Modbus_Comm_Load instruction resolves these issues.

An instance data block is assigned for Modbus_Master or Modbus_Slave when you place these instructions in your program. This instance data block is referenced when you specify the MB_DB parameter for the Modbus_Comm_Load instruction.

Modbus_Comm_Load data block variables

The following table shows the public static variables stored in the instance DB for the Modbus_Comm_Load that can be used in your program.

Table 13- 69 Static variables in the instance DB

Variable	Data type	Description
ICHAR_GAP	UInt	Delay for Inter-character gap between characters. This parameter is specified in milliseconds and is used to increase the expected amount of time between received characters. The corresponding number of bit times for this parameter is added to the Modbus default of 35 bit times (3.5 character times).
RETRIES	UInt	Number of retries that the master will attempt before returning the no response error code "0x80C8".
STOP_BITS	USInt	Number of stop bits used in framing each character. Valid values are 1 and 2.

Table 13- 70 Modbus_Comm_Load execution condition codes [1]

STATUS (W#16#)	Description
0000	No error
8180	Invalid port ID value (wrong port/hardware identifier for communication module)
8181	Invalid baud rate value
8182	Invalid parity value
8183	Invalid flow control value
8184	Invalid response timeout value (response timeout less than the 5 ms minimum)
8185	MB_DB parameter is not an instance data block of a Modbus_Master or Modbus_Slave instruction.

[1] In addition to the Modbus_Comm_Load errors listed above, errors can be returned from the underlying PtP communication instructions.

Modbus_Master (Communicate using SIPLUS I/O or the PtP port as Modbus RTU master) instruction

Table 13- 71 Modbus_Master instruction

LAD / FBD	SCL	Description

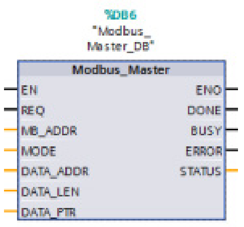
 %DB6
"Modbus_Master_DB"
Modbus_Master
EN
REQ
MB_ADDR
MODE
DATA_ADDR
DATA_LEN
DATA_PTR
ENO
DONE
BUSY
ERROR
STATUS
	`"Modbus_Master_DB"(`
	`    REQ:=_bool_in_,`
	`    MB_ADDR:=_uint_in_,`
	`    MODE:=_usint_in_,`
	`    DATA_ADDR:=_udint_in_,`
	`    DATA_LEN:=_uint_in_,`
	`    DONE=>_bool_out_,`
	`    BUSY=>_bool_out_,`
	`    ERROR=>_bool_out_,`
	`    STATUS=>_word_out_,`
	`    DATA_PTR:=_variant_inout_);`
		The Modbus_Master instruction communicates as a Modbus master using a port that was configured by a previous execution of the Modbus_Comm_Load instruction. An instance data block is assigned automatically when you place the Modbus_Master instruction in your program. This Modbus_Master instance data block is used when you specify the MB_DB parameter for the Modbus_Comm_Load instruction.

Table 13- 72 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	0=No request 1= Request to transmit data to Modbus slave
MB_ADDR	IN	V1.0: USInt V2.0: UInt	Modbus RTU station address: Standard addressing range (1 to 247) Extended addressing range (1 to 65535) The value of 0 is reserved for broadcasting a message to all Modbus slaves. Modbus function codes 05, 06, 15 and 16 are the only function codes supported for broadcast.
MODE	IN	USInt	Mode Selection: Specifies the type of request (read, write, or diagnostic). See the Modbus functions table below for details.
DATA_ADDR	IN	UDInt	Starting Address in the slave: Specifies the starting address of the data to be accessed in the Modbus slave. See the Modbus functions table below for valid addresses.
DATA_LEN	IN	UInt	Data Length: Specifies the number of bits or words to be accessed in this request. See the Modbus functions table below for valid lengths.
DATA_PTR	IN	Variant	Data Pointer: Points to the M or DB address (Standard DB type) for the data being written or read.
DONE	OUT	Bool	The DONE bit is TRUE for one scan, after the last request was completed with no error.
BUSY	OUT	Bool	• 0 – No Modbus_Master operation in progress • 1 – Modbus_Master operation in progress
ERROR	OUT	Bool	The ERROR bit is TRUE for one scan, after the last request was terminated with an error. The error code value at the STATUS parameter is valid only during the single scan where ERROR = TRUE.
STATUS	OUT	Word	Execution condition code

Modbus_Master communication rules

- MB_COMM_LOAD must be executed to configure a port before a Modbus_Master instruction can communicate with that port.
- If a port is to be used to initiate Modbus master requests, that port should not be used by MB_SLAVE. One or more instances of Modbus_Master execution can be used with that port, but all Modbus_Master execution must use the same Modbus_Master instance DB for that port.
- The Modbus instructions do not use communication interrupt events to control the communication process. Your program must poll the Modbus_Master instruction for transmit and receive complete conditions.
- It is recommended that you call all Modbus_Master execution for a given port from a program cycle OB. Modbus_Master instructions may execute in only one of the program cycle or cyclic/time delay execution levels. They must not execute in both execution priority levels. Pre-emption of a Modbus_Master instruction by another Modbus_Master instruction in a higher priority execution priority level will result in improper operation. Modbus_Master instructions must not execute in the startup, diagnostic or time error execution priority levels.
- Once a Modbus_Master instruction initiates a transmission, this instance must be continually executed with the EN input enabled until a DONE=1 state or ERROR=1 state is returned. A particular Modbus_Master instance is considered active until one of these two events occurs. While the original instance is active, any call to any other instance with the REQ input enabled will result in an error. If the continuous execution of the original instance stops, the request state remains active for a period of time specified by the static variable "Blocked_Proc_Timeout". Once this period of time expires, the next Modbus_Master instruction called with an enabled REQ input will become the active instance. This prevents a single Modbus_Master instance from monopolizing or locking access to a port. If the original active instance is not enabled within the period of time specified by the static variable "Blocked_Proc_Timeout", then the next execution by this instance (with REQ not set) will clear the active state. If (REQ is set), then this execution initiates a new Modbus_Master request as if no other instance was active.

REQ parameter

0 = No request; 1 = Request to transmit data to Modbus Slave

You may control this input either through the use of a level or edge triggered contact. Whenever this input is enabled, a state machine is started to ensure that no other Modbus_Master using the same instance DB is allowed to issue a request, until the current request is completed. All other input states are captured and held internally for the current request, until the response is received or an error detected.

If the same instance of Modbus_Master is executed again with REQ input = 1 before the completion of the current request, then no subsequent transmissions are made. However, when the request is completed, a new request is issued whenever a Modbus_Master is executed again with REQ input = 1.

DATA_ADDR and MODE parameters select the Modbus function type

DATA_ADDR (starting Modbus address in the slave): Specifies the starting address of the data to be accessed in the Modbus slave.

The Modbus_Master instruction uses a MODE input rather than a Function Code input. The combination of MODE and Modbus address determine the Function Code that is used in the actual Modbus message. The following table shows the correspondence between parameter MODE, Modbus function code, and Modbus address range.

Table 13- 73 Modbus functions

MODE	Modbus function	Data length	Operation and data	Modbus address
0	01	1 to 2000 1 to 1992 [1]	Read output bits: 1 to (1992 or 2000) bits per request	1 to 9999
0	02	1 to 2000 1 to 1992 [1]	Read input bits: 1 to (1992 or 2000) bits per request	10001 to 19999
0	03	1 to 125 1 to 124 [1]	Read Holding registers: 1 to (124 or 125) words per request	40001 to 49999 or 400001 to 465535
0	04	1 to 125 1 to 124 [1]	Read input words: 1 to (124 or 125) words per request	30001 to 39999
104	04	1 to 125 1 to 124 1	Read input words: 1 to (124 or 125) words per request	00000 to 65535
1	05	1	Write one output bit: One bit per request	1 to 9999
1	06	1	Write one holding register: 1 word per request	40001 to 49999 or 400001 to 465535
1	15	2 to 1968 2 to 1960 [1]	Write multiple output bits: 2 to (1960 or 1968) bits per request	1 to 9999
1	16	2 to 123 2 to 122 [1]	Write multiple holding registers: 2 to (122 or 123) words per request	40001 to 49999 or 400001 to 465535
2	15	1 to 1968 2 to 1960 [1]	Write one or more output bits: 1 to (1960 or 1968) bits per request	1 to 9999
2	16	1 to 123 1 to 122 [1]	Write one or more holding registers: 1 to (122 or 123) words per request	40001 to 49999 or 400001 to 465535
11	11	0	Read the slave communication status word and event counter. The status word indicates busy (0 – not busy, 0xFFFF - busy). The event counter is incremented for each successful completion of a message. Both the DATA_ADDR and DATA_LEN operands of the Modbus_Master instruction are ignored for this function.	
80	08	1	Check slave status using data diagnostic code 0x0000 (Loopback test – slave echoes the request) 1 word per request	

MODE	Modbus function	Data length	Operation and data	Modbus address
81	08	1	Reset slave event counter using data diagnostic code 0x000A 1 word per request	
3 to 10, 12 to 79, 82 to 255			Reserved	

1 For "Extended Addressing" mode the maximum data lengths are reduced by 1 byte or 1 word depending upon the data type used by the function.

DATA_PTR parameter

The DATA_PTR parameter points to the DB or M address that is written to or read from. If you use a data block, then you must create a global data block that provides data storage for reads and writes to Modbus slaves.

Note

The DATA_PTR data block type must allow direct addressing

The data block must allow both direct (absolute) and symbolic addressing. When you create the data block the "Standard" access attribute must be selected.

Data block structures for the DATA_PTR parameter

- These data types are valid for **word reads** of Modbus addresses 30001 to 39999, 40001 to 49999, and 400001 to 465536 and also for **word writes** to Modbus addresses 40001 to 49999 and 400001 to 465536.
  - Standard array of WORD, UINT, or INT data types
  - Named WORD, UINT, or INT structure where each element has a unique name and 16 bit data type.
  - Named complex structure where each element has a unique name and a 16 or 32 bit data type.
- For **bit reads** and writes of Modbus addresses 00001 to 09999 and bit reads of 10001 to 19999.
  - Standard array of Boolean data types.
  - Named Boolean structure of uniquely named Boolean variables.
- Although not required, it is recommended that each Modbus_Master instruction have its own separate memory area. The reason for this recommendation is that there is a greater possibility of data corruption if multiple Modbus_Master instructions are reading and writing to the same memory area.
- There is no requirement that the DATA_PTR data areas be in the same global data block. You can create one data block with multiple areas for Modbus reads, one data block for Modbus writes, or one data block for each slave station.

Modbus master data block variables

The following table shows the public static tags in the instance DB of the Modbus_Master that you can use in your program.

Table 13- 74 Static tags in the instance DB

Tag	Data type	Standard	Description
Blocked_Proc_Timeout	Real	3.0	Amount of time (in seconds) to wait for a blocked Modbus_Master instance before removing this instance as being ACTIVE. This can occur, for example, if a Modbus_Master request is issued and then the program stops to call the Modbus_Master function before it has completely finished the request. The time value must be greater than 0 and less than 55 seconds, or an error occurs.
Extended_Addressing	Bool	FALSE	Configures single or double-byte slave station addressing: • FALSE = One-byte address, 0 to 247 • TRUE = Two-byte address (corresponds to exteneded addressing), 0 to 65535
MD_DB	MB_BASE	-	The MB_DB parameter of the Modbus_Comm_Load instruction must be connected to the MB_DB parameter of the Modnbus_Master instruction.

Your program can write values to the Blocked_Proc_Timeout and Extended_Addressing variables to control the Modbus_Master operations. See the MB_SLAVE topic description of HR_Start_Offset and Extended_Addressing for an example of how to use these variables in the program editor and details about Modbus extended addressing (Page 967).

Condition codes

Table 13- 75 Modbus_Master execution condition codes (communication and configuration errors) [1]

STATUS (W#16#)	Description
0000	No error
80C8	Slave timeout. The specified slave did not respond in the specified time. Please check the baud rate, parity, and wiring of the slave device. This error is only reported after any configured retries have been attempted.
80C9	The Modbus_Master instruction has timed out for one of the following reasons: • The instruction is waiting for a response from the module that is being used for communications. • The Blocked_Proc_Timeout value is set too small. This error is reported if a PROFIBUS or PROFINET distributed I/O device returns from one of the following: • An interruption to power or communication • A communication module pull/plug event In these instances, the hardware configuration from the PLC is reloaded, and Modbus_Comm_Load must be executed again to properly configure the communication module.
80D1	The receiver issued a flow control request to suspend an active transmission and never re-enabled the transmission during the specified wait time. This error is also generated during hardware flow control when the receiver does not assert CTS within the specified wait time.

STATUS (W#16#)	Description
80D2	The transmit request was aborted because no DSR signal is received from the DCE.
80E0	The message was terminated because the receive buffer is full.
80E1	The message was terminated as a result of a parity error.
80E2	The message was terminated as a result of a framing error.
80E3	The message was terminated as a result of an overrun error.
80E4	The message was terminated as a result of the specified length exceeding the total buffer size.
8180	Invalid port ID value or error with Modbus_Comm_Load instruction
8186	Invalid Modbus station address
8188	Invalid Mode specified for broadcast request
8189	Invalid Data Address value
818A	Invalid Data Length value
818B	Invalid pointer to the local data source/destination: Size not correct
818C	Invalid pointer for DATA_PTR or invalid Blocked_Proc_Timeout. The data area must be one of the following: • Classic DB • Array of elemental data types in a symbolic or retentive DB • M memory
8200	Port is busy processing a transmit request.
8280	Negative acknowledgement when reading module. Check the input at the PORT parameter. This can be caused by the loss of a PROFIBUS or PROFINET distributed I/O module, either by a station power or communication loss or a module pull.
8281	Negative acknowledgement when writing to module. Check the input at the PORT parameter. This can be caused by the loss of a PROFIBUS or PROFINET distributed I/O module, either by a station power or communication loss or a module pull.

Table 13- 76 Modbus_Master execution condition codes (Modbus protocol errors) [1]

STATUS (W#16#)	Response code from slave	Modbus protocol errors
8380	-	CRC error
8381	01	Function code not supported
8382	03	Data length error
8383	02	Data address error or address outside the valid range of the DATA_PTR area
8384	Greater than 03	Data value error
8385	03	Data diagnostic code value not supported (function code 08)
8386	-	Function code in the response does not match the code in the request.
8387	-	Wrong slave responded
8388	-	The slave response to a write request is incorrect. The write request returned by the slave does not match what the master actually sent.

[1] In addition to the Modbus_Master errors listed above, errors can be returned from the underlying PtP communication instructions.

Modbus_Slave (Communicate using SIPLUS I/O or the PtP port as Modubus RTU slave) instruction

Table 13- 77 Modbus_Slave instruction

LAD / FBD	SCL	Description

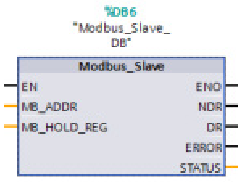
 %DB6
"Modbus_Slave_DB"
Modbus_Slave
EN
ENO
MB_ADDR
NDR
MB_HOLD_REG
DR
ERROR
STATUS
	`"Modbus_Slave_DB"(`
	`    MB_ADDR:=_uint_in_,`
	`    NDR=>_bool_out_,`
	`    DR=>_bool_out_,`
	`    ERROR=>_bool_out_,`
	`    STATUS=>_word_out_,`
	`    MB_HOLD_REG:=_variant_inout_);`
		The Modbus_Slave instruction allows your program to communicate in one of two ways:
		• As a Modbus RTU slave through a PtP port on the CM (RS485 or RS232) and CB (RS485)
		• As a Modbus RTU slave through Modbus RTU SIPLUS I/O options:
		– Install ET 200MP S7-1500CM PtP (RS485 / 422 or RS232).
		– Install ET 200SP S7-1500 CM PtP (RS485 / 422 or RS232).
		When a remote Modbus RTU master issues a request, your user program responds to the request by Modbus_Slave execution. STEP 7 automatically creates an instance DB when you insert the instruction. Use this Modbus_Slave_DB name when you specify the MB_DB parameter for the Modbus_Comm_Load instruction.

Table 13- 78 Data types for the parameters

Parameter and type		Data type	Description
MB_ADDR	IN	V1.0: USInt V2.0: UInt	The station address of the Modbus slave: Standard addressing range (1 to 247) Extended addressing range (0 to 65535)
MB_HOLD_REG	IN	Variant	Pointer to the Modbus Holding Register DB: The Modbus holding register can be M memory or a data block.
NDR	OUT	Bool	New Data Ready: • 0 – No new data • 1 – Indicates that new data has been written by the Modbus master
DR	OUT	Bool	Data Read: • 0 – No data read • 1 – Indicates that data has been read by the Modbus master
ERROR	OUT	Bool	The ERROR bit is TRUE for one scan, after the last request was terminated with an error. If execution is terminated with an error, then the error code value at the STATUS parameter is valid only during the single scan where ERROR = TRUE.
STATUS	OUT	Word	Execution error code

Modbus communication function codes (1, 2, 4, 5, and 15) can read and write bits and words directly in the input process image and output process image of the CPU. For these function codes, the MB_HOLD_REG parameter must be defined as a data type larger than a byte. The following table shows the example mapping of Modbus addresses to the process image in the CPU.

Table 13- 79 Mapping of Modbus addresses to the process image

Modbus functions						**S7-1200**	
Codes	Function	Data area	Address range			Data area	CPU address
01	Read bits	Output	1	to	8192	Output Process Image	Q0.0 to Q1023.7
02	Read bits	Input	10001	to	18192	Input Process Image	I0.0 to I1023.7
04	Read words	Input	30001	to	30512	Input Process Image	IW0 to IW1022
05	Write bit	Output	1	to	8192	Output Process Image	Q0.0 to Q1023.7
15	Write bits	Output	1	to	8192	Output Process Image	Q0.0 to Q1023.7

Modbus communication function codes (3, 6, 16) use a Modbus holding register which can be an M memory address range or a data block. The type of holding register is specified by the MB_HOLD_REG parameter on the Modbus_Slave instruction.

Note

MB_HOLD_REG data block type

A Modbus holding register data block must allow both direct (absolute) and symbolic addressing. When you create the data block the "Standard" access attribute must be selected.

The following table shows examples of Modbus address to holding register mapping that is used for Modbus function codes 03 (read words), 06 (write word), and 16 (write words). The actual upper limit of DB addresses is determined by the maximum work memory limit and M memory limit, for each CPU model.

Table 13- 80 Mapping of Modbus addresses to CPU memory

Modbus master address	**MB_HOLD_REG parameter examples**				
	MW100	**DB10.DBw0**	**MW120**	**DB10.DBW50**	**"Recipe".ingredient**
40001	MW100	DB10.DBW0	MW120	DB10.DBW50	"Recipe".ingredient[1]
40002	MW102	DB10.DBW2	MW122	DB10.DBW52	"Recipe".ingredient[2]
40003	MW104	DB10.DBW4	MW124	DB10.DBW54	"Recipe".ingredient[3]
40004	MW106	DB10.DBW6	MW126	DB10.DBW56	"Recipe".ingredient[4]
40005	MW108	DB10.DBW8	MW128	DB10.DBW58	"Recipe".ingredient[5]

Table 13- 81 Diagnostic functions

S7-1200 Modbus_Slave Modbus diagnostic functions		
Codes	Sub-function	Description
08	0000H	Return query data echo test: The Modbus_Slave will echo back to a Modbus master a word of data that is received.
08	000AH	Clear communication event counter: The Modbus_Slave will clear out the communication event counter that is used for Modbus function 11.
11		Get communication event counter: The Modbus_Slave uses an internal communication event counter for recording the number of successful Modbus read and write requests that are sent to the Modbus_Slave. The counter does not increment on any Function 8, Function 11, or broadcast requests. It is also not incremented on any requests that result in a communication error (for example, parity or CRC errors).

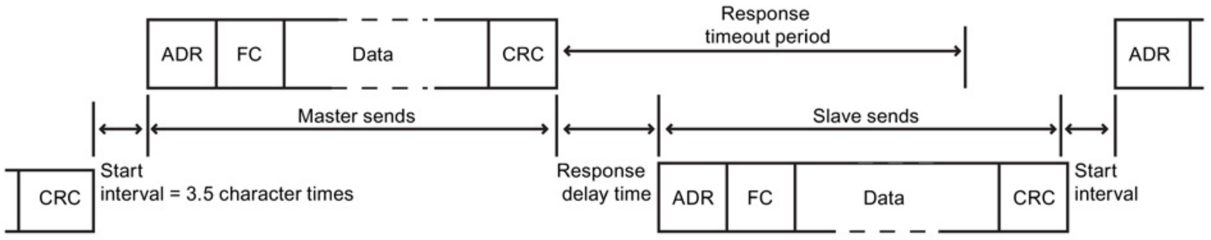
The Modbus_Slave instruction supports broadcast write requests from any Modbus master as long as the request is for accessing valid addresses. Modbus_Slave will produce error code "0x8188" for function codes not supported in broadcast.

Modbus_Slave communication rules

- Modbus_Comm_Load must be executed to configure a port, before a Modbus_Slave instruction can communicate through that port.
- If a port is to respond as a slave to a Modbus_Master, then do not program that port with the Modbus_Master instruction.
- Only one instance of Modbus_Slave can be used with a given port, otherwise erratic behavior may occur.
- The Modbus instructions do not use communication interrupt events to control the communication process. Your program must control the communication process by polling the Modbus_Slave instruction for transmit and receive complete conditions.
- The Modbus_Slaveinstruction must execute periodically at a rate that allows it to make a timely response to incoming requests from a Modbus_Master. It is recommended that you execute Modbus_Slave every scan from a program cycle OB. Executing Modbus_Slave from a cyclic interrupt OB is possible, but is not recommended because of the potential for excessive time delays in the interrupt routine to temporarily block the execution of other interrupt routines.

Modbus signal timing

Modbus_Slave must be executed periodically to receive each request from the Modbus_Master and then respond as required. The frequency of execution for Modbus_Slave is dependent upon the response timeout period of the Modbus_Master. This is illustrated in the following diagram.



The response timeout period RESP_TO is the amount of time a Modbus_Master waits for the start of a response from a Modbus_Slave. This time period is not defined by the Modbus protocol, but is a parameter of each Modbus_Master. The frequency of execution (the time between one execution and the next execution) of Modbus_Slave must be based upon the particular parameters of your Modbus_Master. At a minimum, you should execute Modbus_Slave twice within the response timeout period of the Modbus_Master.

Modbus_Slave variables

This table shows the public static variables stored in the Modbus_Slave instance data block that can be used in your program

Table 13- 82 Modbus_Slave variables

Variable	Data type	Description
Request_Count	Word	The number of all requests received by this slave
Slave_Message_Count	Word	The number of requests received for this specific slave
Bad_CRC_Count	Word	The number of requests received that have a CRC error
Broadcast_Count	Word	The number of broadcast requests received
Exception_Count	Word	Modbus specific errors that require a returned exception
Success_Count	Word	The number of requests received for this specific slave that have no protocol errors
HR_Start_Offset	Word	Specifies the starting address of the Modbus Holding register (default = 0)
Extended_Addressing	Bool	Configures single or double-byte slave addressing (0=single byte address, 1=double-byte address, default = 0)

Your program can write values to the HR_Start_Offset and Extended_Addressing variables and control Modbus slave operations. The other variables can be read to monitor Modbus status.

HR_Start_Offset

Modbus holding register addresses begin at 40001 or 400001. These addresses correspond to the beginning PLC memory address of the holding register. However, you can configure the "HR_Start_Offset" variable to start the beginning Modbus holding register address at another value instead of 40001 or 400001.

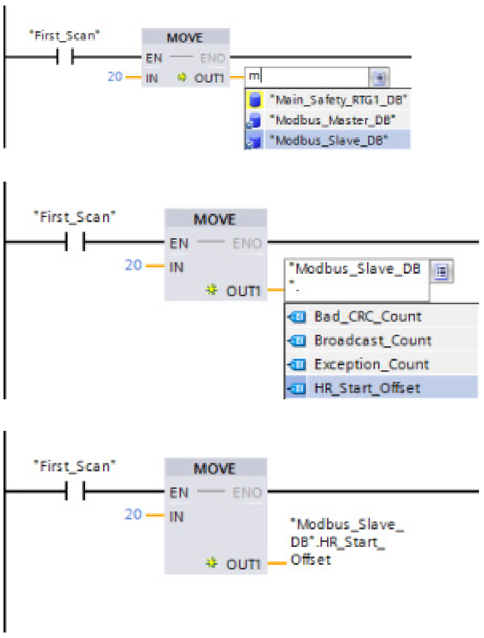
For example, if the holding register is configured to start at MW100 and is 100 words long. An offset of 20 specifies a beginning holding register address of 40021 instead of 40001. Any address below 40021 and above 400119 will result in an addressing error.

Table 13- 83 Example of Modbus holding register addressing

HR_Start_Offset	Address	Minimum	Maximum
0	Modbus address (Word)	40001	40099
	S7-1200 address	MW100	MW298
20	Modbus address (Word)	40021	40119
	S7-1200 address	MW100	MW298

HR_Start_Offset is a word value that specifies the starting address of the Modbus holding register and is stored in the Modbus_Slave instance data block. You can set this public static variable value by using the parameter helper drop-list, after Modbus_Slave is placed in your program.

For example, after Modbus_Slave is placed in a LAD network, you can go to a previous network and assign the HR_Start_Offset value. The value must be assigned prior to execution of Modbus_Slave.



Entering a Modbus slave variable using the default DB name:

1. Set the cursor in the parameter field and type an m character.
2. Select "Modbus_Slave_DB" from the drop-list.
3. Set the cursor at the right side of the DB name (after the quote character) and enter a period character.
4. Select "Modbus_Slave_DB.HR_Start_Offset" from the drop list.

Extended_Addressing

The Extended_Addressing variable is accessed in a similar way as the HR_Start_Offset reference discussed above except that the Extended_Addressing variable is a Boolean value. The Boolean value must be written by an output coil and not a move box.

Modbus slave addressing can be configured to be either a single byte (which is the Modbus standard) or double byte. Extended addressing is used to address more than 247 devices within a single network. Selecting extended addressing allows you to address a maximum of 64000 addresses. A Modbus function 1 frame is shown below as an example.

Table 13- 84 Single-byte slave address (byte 0)

Function 1	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	
Request	Slave addr.	F code	Start address		Length of coils		
Valid Response	Slave addr.	F code	Length	Coil data			
Error response	Slave addr.	0x81	E code				

Table 13- 85 Double-byte slave address (byte 0 and byte 1)

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Request	Slave address		F code	Start address		Length of coils	
Valid Response	Slave address		F code	Length	Coil data		
Error response	Slave address		0x81	E code			

Condition codes

Table 13- 86 Modbus_Slave execution condition codes (communication and configuration errors) [1]

STATUS (W#16#)	Description
80D1	The receiver issued a flow control request to suspend an active transmission and never re-enabled the transmission during the specified wait time. This error is also generated during hardware flow control when the receiver does not assert CTS within the specified wait time.
80D2	The transmit request was aborted because no DSR signal is received from the DCE.
80E0	The message was terminated because the receive buffer is full.
80E1	The message was terminated as a result of a parity error.
80E2	The message was terminated as a result of a framing error.
80E3	The message was terminated as a result of an overrun error.
80E4	The message was terminated as a result of the specified length exceeding the total buffer size.
8180	Invalid port ID value or error with Modbus_Comm_Load instruction
8186	Invalid Modbus station address

STATUS (W#16#)	Description
8187	Invalid pointer to MB_HOLD_REG DB: Area is too small
818C	Invalid MB_HOLD_REG pointer. The data area must be one of the following: • Classic DB • Array of elemental data types in a symbolic or retentive DB • M memory

Table 13- 87 Modbus_Slave execution condition codes (Modbus protocol errors) [1]

STATUS (W#16#)	Response code from slave	Modbus protocol errors
8380	No response	CRC error
8381	01	Function code not supported or not supported within broadcasts
8382	03	Data length error
8383	02	Data address error or address outside the valid range of the DATA_PTR area
8384	03	Data value error
8385	03	Data diagnostic code value not supported (function code 08)

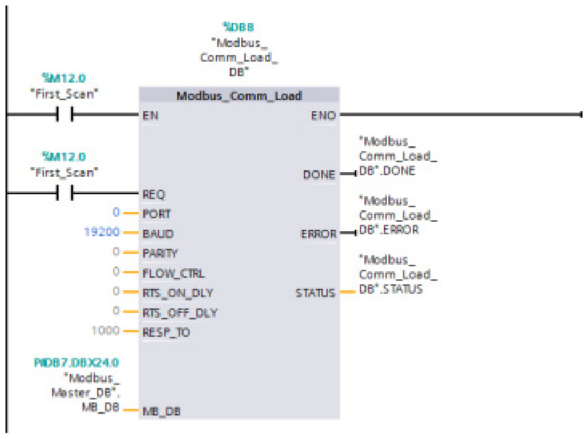
[1] In addition to the Modbus_Slave errors listed above, errors can be returned from the underlying PtP communication instructions.

13.5.3.4 Modbus RTU examples

Example: Modbus RTU master program

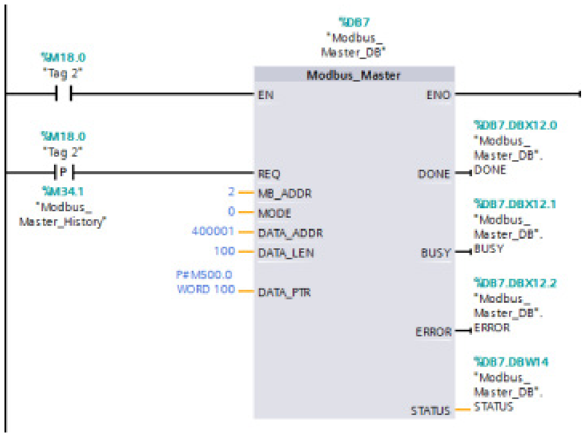
Modbus_Comm_Load is initialized during start-up by using the first scan flag. Execution of Modbus_Comm_Load in this manner should only be done when the serial port configuration will not change at runtime.

Network 1: Initialize the RS485 module parameters only once during the first scan.

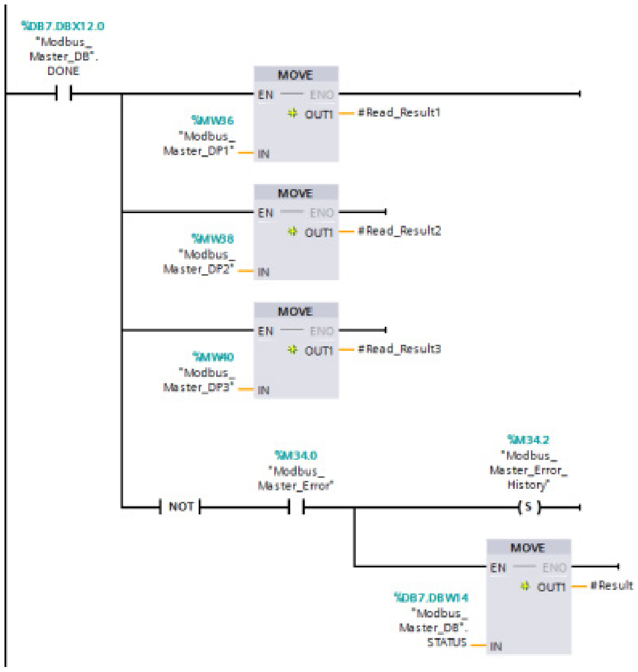


One Modbus_Master instruction is used in the program cycle OB to communicate with a single slave. Additional Modbus_Master instructions can be used in the program cycle OB to communicate with other slaves, or one Modbus_Master FB could be re-used to communicate with additional slaves.

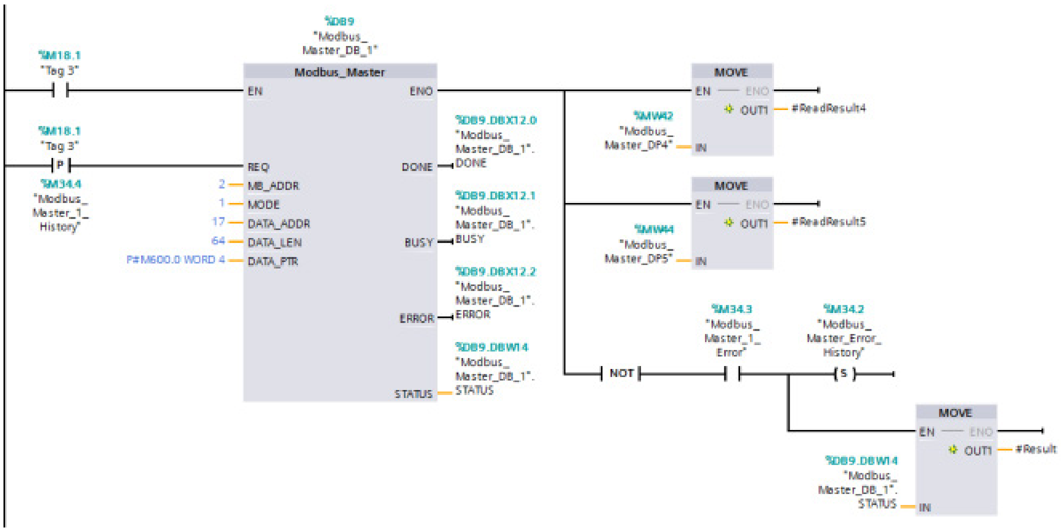
Network 2: Read 100 words from the slave holding register.



Network 3: This is an optional network that just shows the values of the first 3 words once the read operation is done.



Network 4: Write 64 bits to the output image register starting at slave address Q2.0.

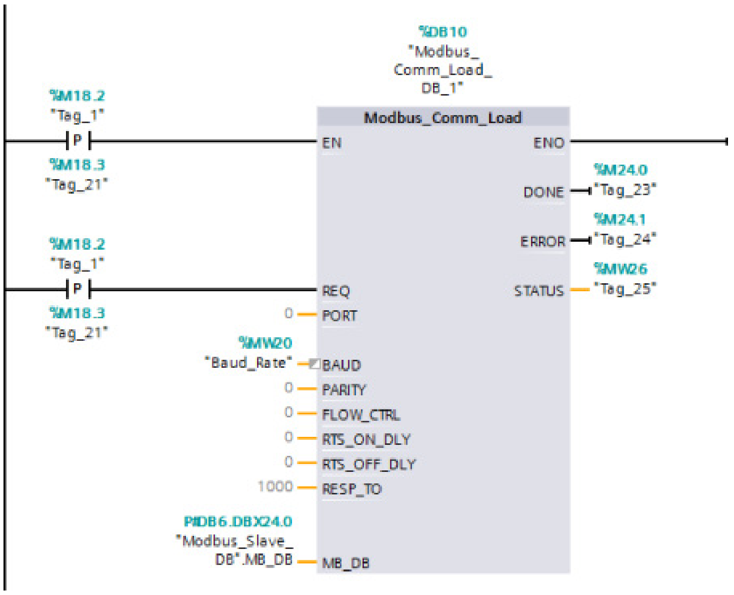


Example: Modbus RTU slave program

MB_COMM_LOAD shown below is initialized each time "Tag_1" is enabled.

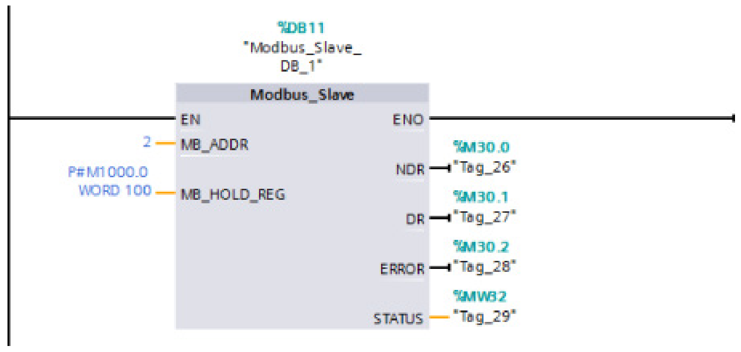
Execution of MB_COMM_LOAD in this manner should only be done when the serial port configuration will change at runtime, as a result of HMI configuration.

Network 1: Initialize the RS485 module parameters each time they are changed by an HMI device.



MB_SLAVE shown below is placed in a cyclic OB that is executed every 10 ms. While this does not give the absolute fastest response by the slave, it does provide good performance at 9600 baud for short messages (20 bytes or less in the request).

Network 2: Check for Modbus master requests during each scan. The Modbus holding register is configured for 100 words starting at MW1000.



13.6 Legacy PtP communication (CM/CB 1241 only)

Prior to the release of STEP 7 V13 SP1 and the S7-1200 V4.1 CPUs, the point-to-point communication instructions existed with different names, and in some cases, slightly different interfaces. The general concepts about point-to-point communication (Page 864), as well as port (Page 865) and parameter configuration (Page 877) apply to both sets of instructions. Refer to the individual legacy point-to-point instructions for programming information.

Table 13- 88 Common error classes

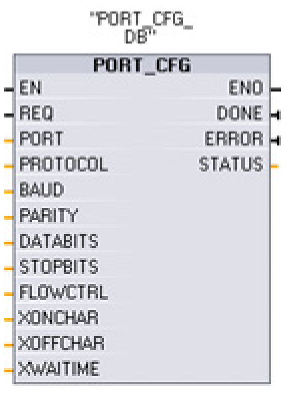
Class description	Error classes	Description
Port configuration	80Ax	Used to define common port configuration errors
Transmit configuration	80Bx	Used to define common transmit configuration errors
Receive configuration	80Cx	Used to define common receive configuration errors
Transmission runtime	80Dx	Used to define common transmission runtime errors
Reception runtime	80Ex	Used to define common reception runtime errors
Signal handling	80Fx	Used to define common errors associated with all signal handling

13.6.1 Legacy point-to-point instructions

13.6.1.1 PORT_CFG (Configure communication parameters dynamically) instruction

Table 13- 89 PORT_CFG (Port Configuration) instruction

LAD / FBD	SCL	Description

 "PORT_CFG_DB"
PORT_CFG
EN
REQ
PORT
PROTOCOL
BAUD
PARITY
DATABITS
STOPBITS
FLOWCTRL
XONCHAR
XOFFCHAR
XWAITIME
ENO
DONE
ERROR
STATUS
	`"PORT_CFG_DB"(`
	`    REQ:=_bool_in_,`
	`    PORT:=_uint_in_,`
	`    PROTOCOL:=_uint_in_,`
	`    BAUD:=_uint_in_,`
	`    PARITY:=_uint_in_,`
	`    DATABITS:=_uint_in_,`
	`    STOPBITS:=_uint_in_,`
	`    FLOWCTRL:=_uint_in_,`
	`    XONCHAR:=_char_in_,`
	`    XOFFCHAR:=_char_in_,`
	`    WAITTIME:=_uint_in_,`
	`    DONE=>_bool_out_,`
	`    ERROR=>_bool_out_,`
	`    STATUS=>_word_out_);`
		PORT_CFG allows you to change port parameters such as baud rate from your program.
		You can set up the initial static configuration of the port in the device configuration properties, or just use the default values. You can execute the PORT_CFG instruction in your program to change the configuration.

1 STEP 7 automatically creates the DB when you insert the instruction.

The PORT_CFG configuration changes are not permanently stored in the CPU. The parameters configured in the device configuration are restored when the CPU transitions from RUN to STOP mode and after a power cycle. See Configuring the communication ports (Page 865) and Managing flow control (Page 867) for more information.

Table 13- 90 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	Activate the configuration change on rising edge of this input. (Default value: False)
PORT	IN	PORT	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table. (Default value: 0)
PROTOCOL	IN	UInt	0 - Point-to-Point communication protocol (Default value) 1..n - future definition for specific protocols
BAUD	IN	UInt	Port baud rate (Default value: 6): 1 = 300 baud, 2 = 600 baud, 3 = 1200 baud, 4 = 2400 baud, 5 = 4800 baud, 6 = 9600 baud, 7 = 19200 baud, 8 = 38400 baud, 9 = 57600 baud, 10 = 76800 baud, 11 = 115200 baud
PARITY	IN	UInt	Port parity (Default value: 1): 1 = No parity, 2 = Even parity, 3 = Odd parity, 4 = Mark parity, 5 = Space parity
DATABITS	IN	UInt	Bits per character (Default value:1): 1 = 8 data bits, 2 = 7 data bits
STOPBITS	IN	UInt	Stop bits (Default value: 1): 1 = 1 stop bit, 2 = 2 stop bits
FLOWCTRL	IN	UInt	Flow control (Default value: 1): 1 = No flow control, 2 = XON/XOFF, 3 = Hardware RTS always ON, 4 = Hardware RTS switched
XONCHAR	IN	Char	Specifies the character that is used as the XON character. This is typically a DC1 character (16#11). This parameter is only evaluated if flow control is enabled. (Default value: 16#11)
XOFFCHAR	IN	Char	Specifies the character that is used as the XOFF character. This is typically a DC3 character (116#3). This parameter is only evaluated if flow control is enabled. (Default value: 16#13)
XWAITIME	IN	UInt	Specifies how long to wait for a XON character after receiving a XOFF character, or how long to wait for the CTS signal after enabling RTS (0 to 65535 ms). This parameter is only evaluated if flow control is enabled. (Default value: 2000)
DONE	OUT	Bool	TRUE for one execution after the last request was completed with no error
ERROR	OUT	Bool	TRUE for one execution after the last request was completed with an error
STATUS	OUT	Word	Execution condition code (Default value: 0)

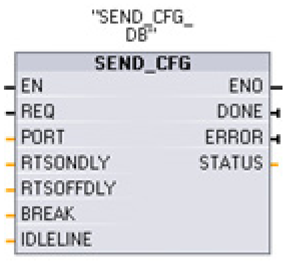
Table 13- 91 Condition codes

STATUS (W#16#....)	Description
80A0	Specific protocol does not exist.
80A1	Specific baud rate does not exist.
80A2	Specific parity option does not exist.
80A3	Specific number of data bits does not exist.
80A4	Specific number of stop bits does not exist.
80A5	Specific type of flow control does not exist.
80A6	Wait time is 0 and flow control enabled
80A7	XON and XOFF are illegal values (for example, the same value)

13.6.1.2 SEND_CFG (Configure serial transmission parameters dynamically) instruction

Table 13- 92 SEND_CFG (Send Configuration) instruction

LAD / FBD	SCL	Description

 "SEND_CFG_DB"[1]
SEND_CFG
EN
ENO
REQ
DONE
PORT
ERROR
RTSONDLY
STATUS
RTSOFFDLY
BREAK
IDLELINE
	`"SEND_CFG_DB"(`
	`REQ:=_bool_in_,`
	`PORT:=_uint_in_,`
	`RTSONDLY:=_uint_in_,`
	`RTSOFFDLY:=_uint_in_,`
	`BREAK:=_uint_in_,`
	`IDLELINE:=_uint_in_,`
	`DONE=>_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_);`
		SEND_CFG allows the dynamic configuration of serial transmission parameters for a PtP communication port. Any queued messages within a CM or CB are discarded when SEND_CFG is executed.

[1] STEP 7 automatically creates the DB when you insert the instruction.

You can set up the initial static configuration of the port in the device configuration properties, or just use the default values. You can execute the SEND_CFG instruction in your program to change the configuration.

The SEND_CFG configuration changes are not permanently stored in the CPU. The parameters configured in the device configuration are restored when the CPU transitions from RUN to STOP mode and after a power cycle. See Configuring transmit (send) parameters (Page 868).

Table 13- 93 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	Activate the configuration change on the rising edge of this input.. (Default value: False)
PORT	IN	PORT	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table. (Default value: 0)
RTSONDLY	IN	UInt	Number of milliseconds to wait after enabling RTS before any Tx data transmission occurs. This parameter is only valid when hardware flow control is enabled. The valid range is 0 - 65535 ms. A value of 0 disables the feature. (Default value: 0)
RTSOFFDLY	IN	UInt	Number of milliseconds to wait after the Tx data transmission occurs before RTS is disabled: This parameter is only valid when hardware flow control is enabled. The valid range is 0 - 65535 ms. A value of 0 disables the feature. (Default value: 0)
BREAK	IN	UInt	This parameter specifies that a break will be sent upon the start of each message for the specified number of bit times. The maximum is 65535 bit times up to an eight second maximum. A value of 0 disables the feature. (Default value: 12)
IDLELINE	IN	UInt	This parameter specifies that the line will remain idle for the specified number of bit times before the start of each message. The maximum is 65535 bit times up to an eight second maximum. A value of 0 disables the feature. (Default value: 12)
DONE	OUT	Bool	TRUE for one execution after the last request was completed with no error
ERROR	OUT	Bool	TRUE for one execution after the last request was completed with an error
STATUS	OUT	Word	Execution condition code (Default value: 0)

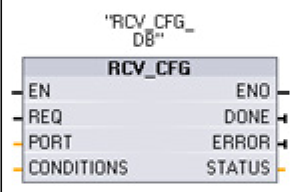
Table 13- 94 Condition codes

STATUS (W#16#....)	Description
80B0	Transmit interrupt configuration is not allowed.
80B1	Break time is greater than the maximum allowed value.
80B2	Idle time is greater than the maximum allowed value.

13.6.1.3 RCV_CFG (Configure serial receive parameters dynamically) instruction

Table 13- 95 RCV_CFG (Receive Configuration) instruction

LAD / FBD	SCL	Description

 "RCV_CFG_DB"
RCV_CFG
EN
ENO
REQ
DONE
PORT
ERROR
CONDITIONS
STATUS
	`"RCV_CFG_DB"(`
	`    REQ:=_bool_in_,`
	`    PORT:=_uint_in_,`
	`    CONDITIONS:=_struct_in_,`
	`    DONE=>_bool_out_,`
	`    ERROR=>_bool_out_,`
	`    STATUS=>_word_out_);`
		RCV_CFG performs dynamic configuration of serial receiver parameters for a PtP communication port. This instruction configures the conditions that signal the start and end of a received message. Any queued messages within a CM or CB are discarded when RCV_CFG is executed.

[1] STEP 7 automatically creates the DB when you insert the instruction.

You can set up the initial static configuration of the communication port in the device configuration properties, or just use the default values. You can execute the RCV_CFG instruction in your program to change the configuration.

The RCV_CFG configuration changes are not permanently stored in the CPU. The parameters configured in the device configuration are restored when the CPU transitions from RUN to STOP mode and after a power cycle. See Configuring receive parameters (Page 869) for more information.

Table 13- 96 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	Activate the configuration change on the rising edge of this input. (Default value: False)
PORT	IN	PORT	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table. (Default value: 0)
CONDITIONS	IN	CONDITIONS	The Conditions data structure specifies the starting and ending message conditions as described below.
DONE	OUT	Bool	TRUE for one scan, after the last request was completed with no error
ERROR	OUT	Bool	TRUE for one scan, after the last request was completed with an error
STATUS	OUT	Word	Execution condition code (Default value: 0)

Start conditions for the RCV_PTP instruction

The RCV_PTP instruction uses the configuration specified by the RCV_CFG instruction to determine the beginning and ending of point-to-point communication messages. The start of a message is determined by the start conditions. The start of a message can be determined by one or a combination of start conditions. If more than one start condition is specified, all the conditions must be satisfied before the message is started.

See the topic "Configuring receive parameters (Page 869)" for a description of the message start conditions.

Parameter CONDITIONS data type structure part 1 (start conditions)

Table 13- 97 CONDITIONS structure for START conditions

Parameter and type		Data type	Description
STARTCOND	IN	UInt	Specifies the start condition (Default value: 1) • 01H - Start Char • 02H - Any Char • 04H - Line Break • 08H - Idle Line • 10H - Sequence 1 • 20H - Sequence 2 • 40H - Sequence 3 • 80H - Sequence 4
IDLETIME	IN	UInt	The number of bit times required for idle line timeout. (Default value: 40). Only used with an idle line condition. 0 to 65535
STARTCHAR	IN	Byte	The start character used with the start character condition. (Default value: B#16#2)
SEQ[1].CTL	IN	Byte	Sequence 1 ignore/compare control for each character: (Default value: B#16#0) These are the enabling bits for each character in start sequence • 01H - Character 1 • 02H - Character 2 • 04H - Character 3 • 08H - Character 4 • 10H - Character 5 Disabling the bit associated with a character means any character will match, in this sequence position.
SEQ[1].STR	IN	Char[5]	Sequence 1 start characters (5 characters). Default value: 0
SEQ[2].CTL	IN	Byte	Sequence 2 ignore/compare control for each character. Default value: B#16#0)
SEQ[2].STR	IN	Char[5]	Sequence 2 start characters (5 characters). Default value: 0
SEQ[3].CTL	IN	Byte	Sequence 3 ignore/compare control for each character. Default value: B#16#0
SEQ[3].STR	IN	Char[5]	Sequence 3 start characters (5 characters). Default value: 0

Parameter and type		Data type	Description
SEQ[4].CTL	IN	Byte	Sequence 4 ignore/compare control for each character. Default value: B#16#0
SEQ[4].STR	IN	Char[5]	Sequence 4 start characters (5 characters), Default value: 0

Example

Consider the following received hexadecimal coded message: "**68** 10 aa **68** bb 10 aa 16" and the configured start sequences shown in the table below. Start sequences begin to be evaluated when the first 68H character is successfully received. Upon successfully receiving the fourth character (the second 68H), then start condition 1 is satisfied. Once the start conditions are satisfied, the evaluation of the end conditions begins.

The start sequence processing can be terminated due to various parity, framing, or inter-character timing errors. These errors result in no received message, because the start condition was not satisfied.

Table 13- 98 Start conditions

Start condition	First Character	First Character +1	First Character +2	First Character +3	First Character +4
1	**68**H	xx	xx	**68**H	xx
2	10H	aaH	xx	xx	xx
3	dcH	aaH	xx	xx	xx
4	e5H	xx	xx	xx	xx

End conditions for the RCV_PTP instruction

The end of a message is determined by the specification of end conditions. The end of a message is determined by the first occurrence of one or more configured end conditions. The section "Message end conditions" in the topic "Configuring receive parameters (Page 869)" describes the end conditions that you can configure in the RCV_CFG instruction.

You can configure the end conditions in either the properties of the communication interface in the device configuration, or from the RCV_CFG instruction. Whenever the CPU transitions from STOP to RUN, the receive parameters (both start and end conditions) return to the device configuration settings. If the STEP 7 user program executes RCV_CFG, then the settings are changed to the RCV_CFG conditions.

Parameter CONDITIONS data type structure part 2 (end conditions)

Table 13- 99 CONDITIONS structure for END conditions

Parameter	Parameter type	Data type	Description
ENDCOND	IN	UInt 0	This parameter specifies message end condition: • 01H - Response time • 02H - Message time • 04H - Inter-character gap • 08H - Maximum length • 10H - N + LEN + M • 20H - Sequence
MAXLEN	IN	UInt 1	Maximum message length: Only used when the maximum length end condition is selected. 1 to 1024 bytes
N	IN	UInt 0	Byte position within the message of the length field. Only used with the N + LEN + M end condition. 1 to 1022 bytes
LENGTHSIZE	IN	UInt 0	Size of the length field (1, 2, or 4 bytes). Only used with the N + LEN + M end condition.
LENGTHM	IN	UInt 0	Specify the number of characters following the length field that are not included in the value of the length field. This is only used with the N + LEN + M end condition. 0 to 255 bytes
RCVTIME	IN	UInt 200	Specify how long to wait for the first character to be received. The receive operation will be terminated with an error if a character is not successfully received within the specified time. This is only used with the response time condition. (0 to 65535 bit times with an 8 second maximum) This parameter is not a message end condition since evaluation terminates when the first character of a response is received. It is an end condition only in the sense that it terminates a receiver operation because no response is received when a response is expected. You must select a separate end condition.
MSGTIME	IN	UInt 200	Specify how long to wait for the entire message to be completely received once the first character has been received. This parameter is only used when the message timeout condition is selected. (0 to 65535 milliseconds)
CHARGAP	IN	UInt 12	Specify the number of bit times between characters. If the number of bit times between characters exceeds the specified value, then the end condition will be satisfied. This is only used with the inter-character gap condition. (0 to 65535 bit times up to 8 second maximum)

Parameter	Parameter type	Data type	Description
SEQ.CTL	IN	Byte B#16#0	Sequence 1 ignore/compare control for each character: These are the enabling bits for each character for the end sequence. Character 1 is bit 0, character 2 is bit 1, …, character 5 is bit 4. Disabling the bit associated with a character means any character will match, in this sequence position.
SEQ.STR	IN	Char[5] 0	Sequence 1 start characters (5 characters)

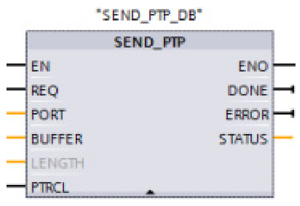
Table 13- 100 Condition codes

STATUS (W#16#....)	Description
80C0	Illegal start condition selected
80C1	Illegal end condition selected, no end condition selected
80C2	Receive interrupt enabled and this is not possible.
80C3	Maximum length end condition is enabled and max length is 0 or > 1024.
80C4	Calculated length is enabled and N is >= 1023.
80C5	Calculated length is enabled and length is not 1, 2 or 4.
80C6	Calculated length is enabled and M value is > 255.
80C7	Calculated length is enabled and calculated length is > 1024.
80C8	Response timeout is enabled and response timeout is zero.
80C9	Inter-character gap timeout is enabled and it is zero.
80CA	Idle line timeout is enabled and it is zero.
80CB	End sequence is enabled but all chars are "don't care".
80CC	Start sequence (any one of 4) is enabled but all characters are "don't care".

13.6.1.4 SEND_PTP (Transmit send buffer data) instruction

Table 13- 101 SEND_PTP (Send Point-to-Point data) instruction

LAD / FBD	SCL	Description

 "SEND_PTP_DB"
SEND_PTP
EN
ENO
REQ
DONE
PORT
ERROR
BUFFER
STATUS
LENGTH
PTRCL
	`"SEND_PTP_DB"(`
	`REQ:=_bool_in_,`
	`PORT:=_uint_in_,`
	
	`BUFFER:=_variant_in_,`
	`LENGTH:=_uint_in_,`
	`PTRCL:=_bool_in_,`
	`DONE=>_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_);`
		SEND_PTP initiates the transmission of the data and transfers the assigned buffer to the communication interface. The CPU program continues while the CM or CB sends the data at the assigned baud rate. Only one send operation can be pending at a given time. The CM or CB returns an error if a second SEND_PTP is executed while the CM or CB is already transmitting a message.

1 STEP 7 automatically creates the DB when you insert the instruction.

Table 13- 102 Data types for the parameters

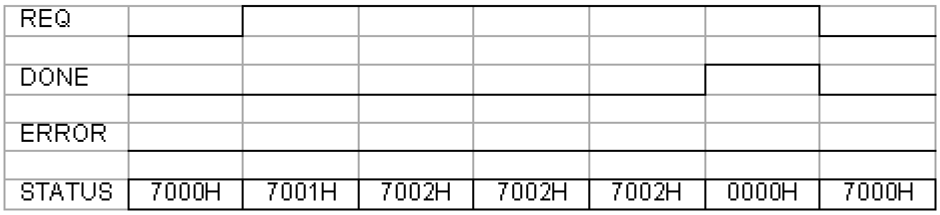
Parameter and type		Data type	Description
REQ	IN	Bool	Activates the requested transmission on the rising edge of this transmission enable input. This initiates transfer of the contents of the buffer to the Point-to-Point communication interface. (Default value: False)
PORT	IN	PORT	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table. (Default value: 0)
BUFFER	IN	Variant	This parameter points to the starting location of the transmit buffer. (Default value: 0) **Note:** Boolean data or Boolean arrays are not supported.
LENGTH [1]	IN	UInt	Transmitted frame length in bytes (Default value: 0) When transmitting a complex structure, always use a length of 0.
PTRCL	IN	Bool	Reserved for future use
DONE	OUT	Bool	TRUE for one scan, after the last request was completed with no error
ERROR	OUT	Bool	TRUE for one scan, after the last request was completed with an error
STATUS	OUT	Word	Execution condition code (Default value: 0)

[1] Optional parameter: Click the arrow at the bottom of a LAD/FBD box to expand the box and include this parameter.

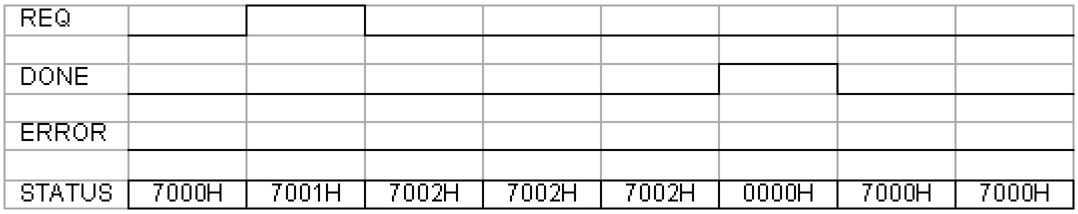
While a transmit operation is in progress, the DONE and ERROR outputs are FALSE. When a transmit operation is complete, either the DONE or the ERROR output will be set TRUE to show the status of the transmit operation. While DONE or ERROR is TRUE, the STATUS output is valid.

The instruction returns a status of 16#7001 if the communication interface accepts the transmit data. Subsequent SEND_PTP executions return 16#7002, if the CM or CB is still busy transmitting. When the transmit operation is complete, the CM or CB returns the status of the transmit operation as 16#0000 (if no errors occurred). Subsequent executions of SEND_PTP with REQ low return a status of 16#7000 (not busy).

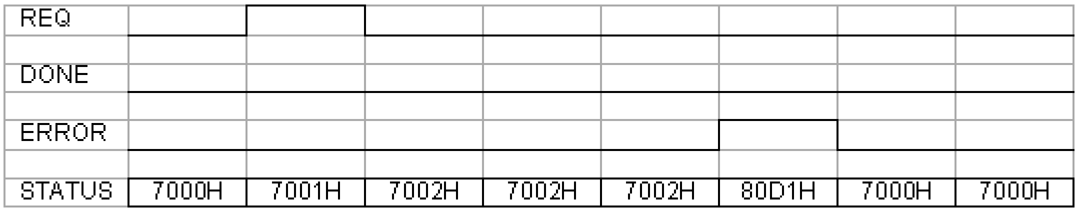
The following diagrams show the relationship of the output values to REQ. This assumes that the instruction is called periodically to check for the status of the transmission process. In the diagram below, it is assumed that the instruction is called every scan (represented by the STATUS values).



The following diagram shows how the DONE and STATUS parameters are valid for only one scan if the REQ line is pulsed (for one scan) to initiate the transmit operation.



The following diagram shows the relationship of DONE, ERROR and STATUS parameters when there is an error.



The DONE, ERROR and STATUS values are only valid until SEND_PTP executes again with the same instance DB.

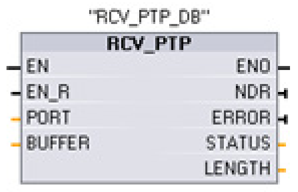
Table 13- 103 Condition codes

STATUS (W#16#....)	Description
80D0	New request while transmitter active
80D1	Transmit aborted because of no CTS within wait time
80D2	Transmit aborted because of no DSR from the DCE device
80D3	Transmit aborted because of queue overflow (transmit more than 1024 bytes)
80D5	Reverse bias signal (wire break condition)
833A	The DB for the BUFFER parameter does not exist.

13.6.1.5 RCV_PTP (Enable receive messages) instruction

Table 13- 104 RCV_PTP (Receive Point-to-Point) instruction

LAD / FBD	SCL	Description

 "RCV_PTP_DB"
RCV_PTP
EN
ENO
EN_R
NDR
PORT
ERROR
BUFFER
STATUS
LENGTH
	`"RCV_PTP_DB"(`
	`EN_R:=_bool_in_,`
	`PORT:=_uint_in_,`
	`BUFFER:=_variant_in_,`
	`NDR=>_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_,`
	`LENGTH=>_uint_out_);`
		RCV_PTP checks for messages that have been received in the CM or CB. If a message is available, it will be transferred from the CM or CB to the CPU. An error returns the appropriate STATUS value.

1 STEP 7 automatically creates the DB when you insert the instruction.

Table 13- 105 Data types for the parameters

Parameter and type		Data type	Description
EN_R	IN	Bool	When this input is TRUE and a message is available, the message is transferred from the CM or CB to the BUFFER. When EN_R is FALSE, the CM or CB is checked for messages and NDR, ERROR and STATUS output are updated, but the message is not transferred to the BUFFER. (Default value: 0)
PORT	IN	PORT	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table. (Default value: 0)
BUFFER	IN	Variant	This parameter points to the starting location of the receive buffer. This buffer should be large enough to receive the maximum length message. Boolean data or Boolean arrays are not supported. (Default value: 0)
NDR	OUT	Bool	TRUE for one execution when new data is ready and operation is complete with no errors.
ERROR	OUT	Bool	TRUE for one execution after the operation was completed with an error.
STATUS	OUT	Word	Execution condition code (Default value: 0)
LENGTH	OUT	UInt	Length of the returned message in bytes (Default value: 0)

Note the following correlation between the EN_R input and the message buffer of the RCV_PTP instruction:

Input EN_R controls the copy of a received message to the BUFFER.

When the EN_R input is TRUE and a message is available, the CPU transfers the message from the CM or CB to the BUFFER and updates the NDR, ERROR, STATUS, and LENGTH outputs.

When EN_R is FALSE, the CPU checks the CM or CB for messages and updates the NDR, ERROR, and STATUS outputs, but does not transfer the message to the BUFFER. (Note that the default value of EN_R is FALSE.)

The recommended practice is to set EN_R to TRUE and control execution of the RCV_PTP instruction with the EN input.

The STATUS value is valid when either NDR or ERROR is TRUE. The STATUS value provides the reason for termination of the receive operation in the CM or CB. This is typically a positive value, indicating that the receive operation was successful and that the receive process terminated normally. If the STATUS value is negative (the Most Significant Bit of the hexadecimal value is set), the receive operation was terminated for an error condition such as parity, framing, or overrun errors.

Each PtP communication interface can buffer up to a maximum of 1024 bytes. This could be one large message or several smaller messages. If more than one message is available in the CM or CB, the RCV_PTP instruction returns the oldest message available. A subsequent RCV_PTP instruction execution returns the next oldest message available.

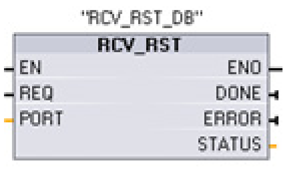
Table 13- 106 Condition codes

STATUS (W#16#...)	Description
0000	No buffer present
0094	Message terminated due to received maximum character length
0095	Message terminated because of message timeout
0096	Message terminated because of inter-character timeout
0097	Message terminated because of response timeout
0098	Message terminated because the "N+LEN+M" length condition was satisfied
0099	Message terminated because of end sequence was satisfied
80E0	Message terminated because the receive buffer is full
80E1	Message terminated due to parity error
80E2	Message terminated due to framing error
80E3	Message terminated due to overrun error
80E4	Message terminated because calculated length exceeds buffer size
80E5	Reverse bias signal (wire break condition)
833A	The DB for the BUFFER parameter does not exist.

13.6.1.6 RCV_RST (Delete receive buffer) instruction

Table 13- 107 RCV_RST (Receiver Reset) instruction

LAD / FBD	SCL	Description

 "RCV_RST_DB"
RCV_RST
EN
ENO
REQ
DONE
PORT
ERROR
STATUS
	`"RCV_RST_DB"(`
	`    REQ:=_bool_in_,`
	`    PORT:=_uint_in_,`
	`    DONE=>_bool_out_,`
	`    ERROR=>_bool_out_,`
	`    STATUS=>_word_out_);`
		RCV_RST clears the receive buffers in the CM or CB.

1 STEP 7 automatically creates the DB when you insert the instruction.

Table 13- 108 Data types for parameters

Parameter and type		Data type	Description
REQ	IN	Bool	Activates the receiver reset on the rising edge of this enable input (Default value: False)
PORT	IN	PORT	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table. (Default value: 0)
DONE	OUT	Bool	When TRUE for one scan, indicates that the last request was completed without errors.

Parameter and type		Data type	Description
ERROR	OUT	Bool	When TRUE, shows that the last request was completed with errors. Also, when this output is TRUE, the STATUS output will contain related error codes.
STATUS	OUT	Word	Error code (Default value: 0) See Common parameters for Point-to-Point instructions (Page 877) for communication status codes.

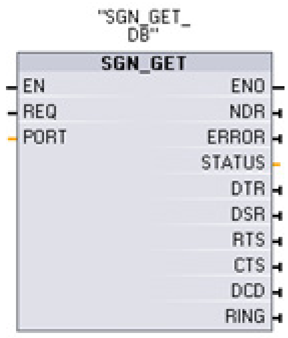
Note

You might want to use the RCV_RST instruction to be sure the message buffers are clear following a communications error, or after changing a communication parameter such as the baud rate. Executing RCV_RST causes the module to clear all of the internal message buffers. After clearing the message buffers, you can be assured that when your program executes a subsequent receive instruction, the messages it returns are new messages and not old messages from some time prior to the RCV_RST call.

13.6.1.7 SGN_GET (Query RS-232 signals) instruction

Table 13- 109 SGN_GET (Get RS232 signals) instruction

LAD / FBD	SCL	Description

 "SGN_GET_DB"
SGN_GET
EN
REQ
PORT
ENO
NDR
ERROR
STATUS
DTR
DSR
RTS
CTS
DCD
RING
	`"SGN_GET_DB"(`
	`REQ:=_bool_in_,`
	`PORT:=_uint_in_,`
	`NDR=>_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_,`
	`DTR=>_bool_out_,`
	`DSR=>_bool_out_,`
	`RTS=>_bool_out_,`
	`CTS=>_bool_out_,`
	`DCD=>_bool_out_,`
	`RING=>_bool_out_);`
		SGN_GET reads the current states of RS232 communication signals.
		This function is valid only for the RS232 CM.

1 STEP 7 automatically creates the DB when you insert the instruction.

Table 13- 110 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	Get RS232 signal state values on the rising edge of this input (Default value: False)
PORT	IN	PORT	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table.
NDR	OUT	Bool	TRUE for one scan, when new data is ready and the operation is complete with no errors
ERROR	OUT	Bool	TRUE for one scan, after the operation was completed with an error
STATUS	OUT	Word	Execution condition code (Default value: 0)
DTR	OUT	Bool	Data terminal ready, module ready (output). Default value: False
DSR	OUT	Bool	Data set ready, communication partner ready (input). Default value: False
RTS	OUT	Bool	Request to send, module ready to send (output). Default value: False
CTS	OUT	Bool	Clear to send, communication partner can receive data (input). Default value: False
DCD	OUT	Bool	Data carrier detect, receive signal level (always False, not supported)
RING	OUT	Bool	Ring indicator, indication of incoming call (always False, not supported)

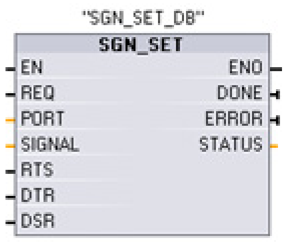
Table 13- 111 Condition codes

STATUS (W#16#....)	Description
80F0	CM or CB is RS485 and no signals are available

13.6.1.8 SGN_SET (Set RS-232 signals) instruction

Table 13- 112 SGN_SET (Set RS232 signals) instruction

LAD / FBD	SCL	Description

 "SGN_SET_DB"
SGN_SET
EN
ENO
REQ
DONE
PORT
ERROR
SIGNAL
STATUS
RTS
DTR
DSR
	`"SGN_SET_DB"(`
	`REQ:=_bool_in_,`
	`PORT:=_uint_in_,`
	`SIGNAL:=_byte_in_,`
	`RTS:=_bool_in_,`
	`DTR:=_bool_in_,`
	`DSR:=_bool_in_,`
	`DONE=>_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_);`
		SGN_SET sets the states of RS232 communication signals.
		This function is valid only for the RS232 CM.

1 STEP 7 automatically creates the DB when you insert the instruction.

Table 13- 113 Data types for parameters

Parameter and type		Data type	Description
REQ	IN	Bool	Start the set RS232 signals operation, on the rising edge of this input (Default value: False)
PORT	IN	PORT	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table. (Default value: 0)
SIGNAL	IN	Byte	Selects which signal to set: (multiple allowed). Default value: 0 • 01H = Set RTS • 02H = Set DTR • 04H = Set DSR
RTS	IN	Bool	Request to send, module ready to send value to set (true or false), Default value: False
DTR	IN	Bool	Data terminal ready, module ready to send value to set (true or false). Default value: False
DSR	IN	Bool	Data set ready (only applies to DCE type interfaces), not used.
DONE	OUT	Bool	TRUE for one execution after the last request was completed with no error
ERROR	OUT	Bool	TRUE for one execution after the last request was completed with an error
STATUS	OUT	Word	Execution condition code (Default value: 0)

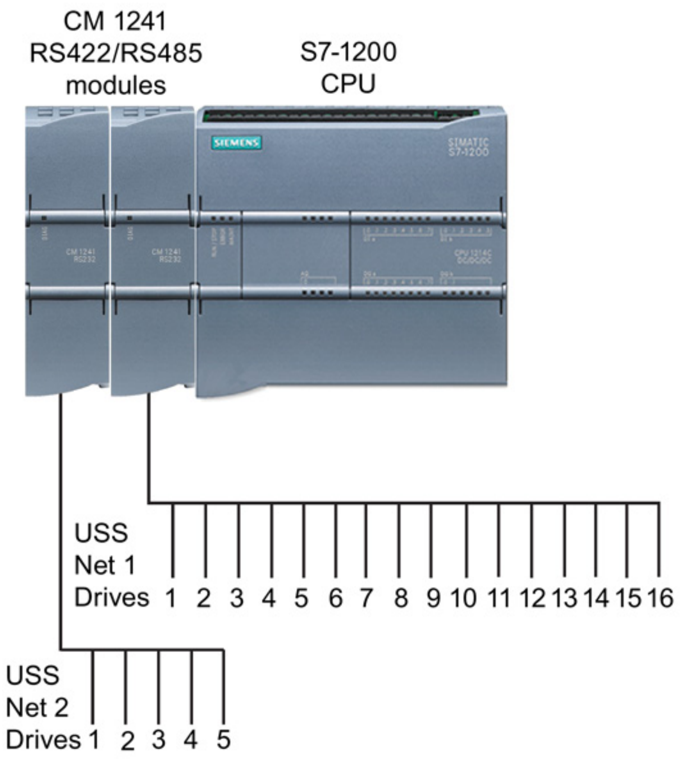
Table 13- 114 Condition codes

STATUS (W#16#....)	Description
80F0	CM or CB is RS485 and no signals can be set
80F1	Signals cannot be set because of Hardware flow control
80F2	Cannot set DSR because module is DTE
80F3	Cannot set DTR because module is DCE

13.7 Legacy USS communication (CM/CB 1241 only)

The USS instructions control the operation of motor drives which support the universal serial interface (USS) protocol. You can use the USS instructions to communicate with multiple drives through RS485 connections to CM 1241 RS485 communication modules or a CB 1241 RS485 communication board. Up to three CM 1241 RS422/RS485 modules and one CB 1241 RS485 board can be installed in a S7-1200 CPU. Each RS485 port can operate up to sixteen drives.

The USS protocol uses a master-slave network for communications over a serial bus. The master uses an address parameter to send a message to a selected slave. A slave itself can never transmit without first receiving a request to do so. Direct message transfer between the individual slaves is not possible. USS communication operates in half-duplex mode. The following USS illustration shows a network diagram for an example drive application.



Prior to the release of STEP 7 V13 SP1 and the S7-1200 V4.1 CPUs, the USS communication instructions existed with different names, and in some cases, slightly different interfaces. The general concepts apply to both sets of instructions. Refer to the individual legacy USS instructions for programming information.

13.7.1 Selecting the version of the USS instructions

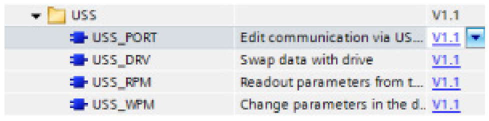
There are two versions of USS instructions available in STEP 7:

- Version 2.0 was initially available in STEP 7 Basic/Professional V13.
- Version 2.1 is available in STEP 7 Basic/Professional V13, SP1.

For compatibility and ease of migration, you can choose which instruction version to insert into your user program.

You cannot use both versions of the instructions with the same module, but two different modules can use different versions of the instructions.

Click the icon on the instruction tree task card to enable the headers and columns of the instruction tree.



USS		V1.1
USS_PORT	Edit communication via US...	V1.1
USS_DRV	Swap data with drive	V1.1
USS_RPM	Readout parameters from t...	V1.1
USS_WPM	Change parameters in the d..	V1.1

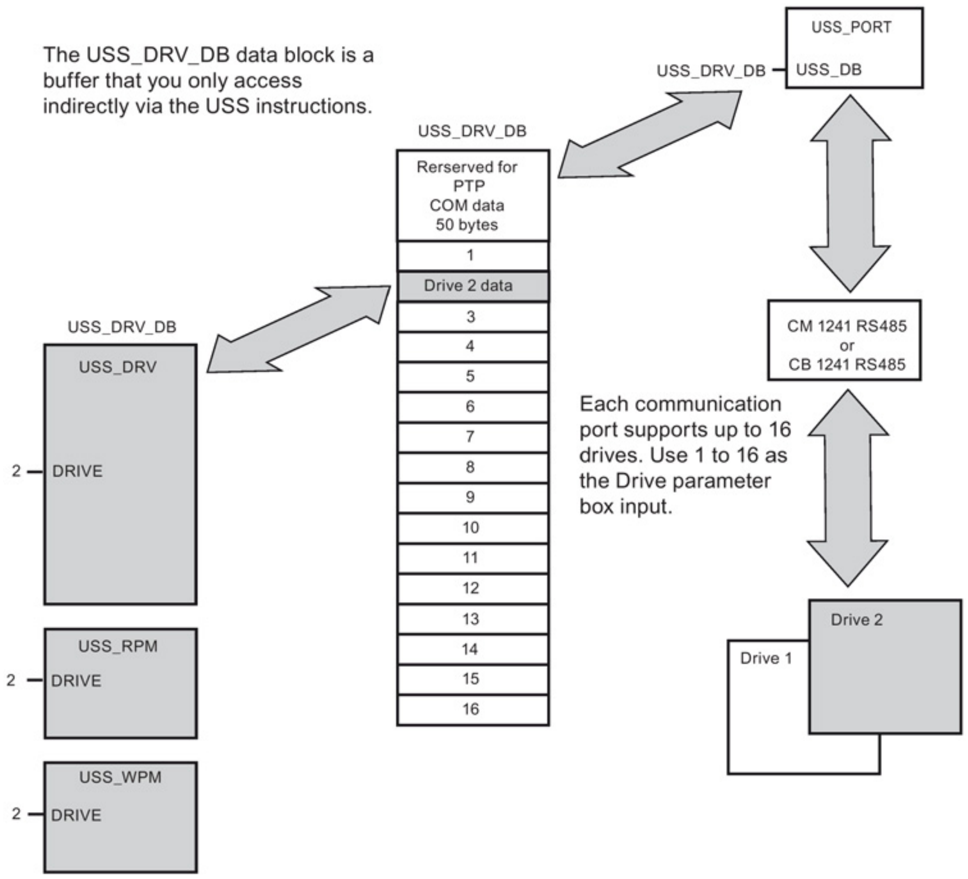
To change the version of the USS instructions, select the version from the drop-down list. You can select the group or individual instructions.

When you use the instruction tree to place a USS instruction in your program, a new FB or FC instance, depending on the USS instruction selected, is created in the project tree. You can see new FB or FC instance in the project tree under PLC_x > Program blocks > System blocks > Program resources.

To verify the version of a USS instruction in a program, you must inspect project tree properties and not the properties of a box displayed in the program editor. Select a project tree USS FB or FC instance, right-click, select "Properties", and select the "Information" page to see the USS instruction version number.

13.7.2 Requirements for using the USS protocol

The four USS instructions use 1 FB and 3 FCs to support the USS protocol. One USS_PORT instance data block (DB) is used for each USS network. The USS_PORT instance data block contains temporary storage and buffers for all drives on that USS network. The USS instructions share the information in this data block.


The USS_DRV_DB data block is a buffer that you only access indirectly via the USS instructions.
USS_PORT
USS_DRV_DB
USS_DB
USS_DRV_DB
Rerserved for PTP COM data 50 bytes
1
Drive 2 data
3
4
5
6
7
8
9
10
11
12
13
14
15
16
USS_DRV_DB
USS_DRV
2
DRIVE
USS_RPM
2
DRIVE
USS_WPM
2
DRIVE
CM 1241 RS485 or CB 1241 RS485
Each communication port supports up to 16 drives. Use 1 to 16 as the Drive parameter box input.
Drive 2
Drive 1

All drives (up to 16) connected to a single RS485 port are part of the same USS network. All drives connected to a different RS485 port are part of a different USS network. Each USS network is managed using a unique data block. All instructions associated with a single USS network must share this data block. This includes all USS_DRV, USS_PORT, USS_RPM, and USS_WPM instructions used to control all drives on a single USS network.

The USS_DRV instruction is a Function Block (FB). When you place the USS_DRV instruction into the program editor, you will be prompted by the "Call options" dialog to assign a DB for this FB. If this is the first USS_DRV instruction in this program for this USS network, then you can accept the default DB assignment (or change the name if you wish) and the new DB is created for you. If however this is not the first USS_DRV instruction for this channel, then you must use the drop-down list in the "Call options" dialog to select the DB name that was previously assigned for this USS network.

Instructions USS_PORT, USS_RPM, and USS_WPM are all Functions (FCs). No DB is assigned when you place these FCs in the editor. Instead, you must assign the appropriate DB reference to the "USS_DB" input of these instructions. Double-click on the parameter field and then click on the parameter helper icon to see the available DB names).

The USS_PORT function handles the actual communication between the CPU and the drives via the Point-to-Point (PtP) RS485 communication port. Each call to this function handles one communication with one drive. Your program must call this function fast enough to prevent a communication timeout by the drives. You may call this function in a main program cycle OB or any interrupt OB.

Typically, you should call the USS_PORT function from a cyclic interrupt OB. The cycle time of the cyclic interrupt OB should be set to about half of the minimum call interval (As an example, 1200 baud communication should use a cyclic time of 350 ms or less).

The USS_DRV function block provides your program access to a specified drive on the USS network. Its inputs and outputs are the status and controls for the drive. If there are 16 drives on the network, your program must have at least 16 USS_DRV calls, one for each drive. These blocks should be called at the rate that is required to control the operation of the drive.

You may only call the USS_DRV function block from a main program cycle OB.

CAUTION

Considerations in calling USS instructions from OBs

Only call USS_DRV, USS_RPM, and USS_WPM from a main program cycle OB. The USS_PORT function can be called from any OB, usually from a cyclic interrupt OB.

Do not use instructions USS_DRV, USS_RPM, or USS_WPM in a higher priority OB than the corresponding USS_PORT instruction. For example, do not place the USS_PORT in the main and a USS_RPM in a cyclic interrupt OB. Failure to prevent interruption of USS_PORT execution can produce unexpected errors, which could result in personal injury.

The USS_RPM and USS_WPM functions read and write the remote drive operating parameters. These parameters control the internal operation of the drive. See the drive manual for the definition of these parameters. Your program can contain as many of these functions as necessary, but only one read or write request can be active per drive, at any given time. You may only call the USS_RPM and USS_WPM functions from a main program cycle OB.

Calculating the time required for communicating with the drive

Communications with the drive are asynchronous to the S7-1200 scan cycle. The S7-1200 typically completes several scans before one drive communications transaction is completed.

The USS_PORT interval is the time required for one drive transaction. The table below shows the minimum USS_PORT interval for each communication baud rate. Calling the USS_PORT function more frequently than the USS_PORT interval will not increase the number of transactions. The drive timeout interval is the amount of time that might be taken for a transaction, if communications errors caused 3 tries to complete the transaction. By default, the USS protocol library automatically does up to 2 retries on each transaction.

Table 13- 115 Calculating the time requirements

Baud rate	Calculated minimum USS_PORT call Interval (milliseconds)	Drive message interval timeout per drive (milliseconds)
1200	790	2370
2400	405	1215
4800	212.5	638
9600	116.3	349
19200	68.2	205
38400	44.1	133
57600	36.1	109
115200	28.1	85

13.7.3 Legacy USS instructions

13.7.3.1 USS_PORT (Edit communication using USS network) instruction

Table 13- 116 USS_PORT instruction

LAD / FBD	SCL	Description

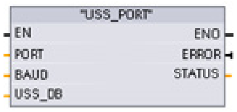
 "USS_PORT"
EN
ENO
PORT
ERROR
BAUD
STATUS
USS_DB
	`USS_PORT(`
	`PORT:=_uint_in_,`
	`BAUD:=_dint_in_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_,`
	`USS_DB:=_fbtref_inout_);`
		The USS_PORT instruction handles communication over a USS network.

Table 13- 117 Data types for the parameters

Parameter and type		Data type	Description
PORT	IN	Port	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table.
BAUD	IN	DInt	The baud rate used for USS communication.
USS_DB	INOUT	USS_BASE	The name of the instance DB that is created and initialized when a USS_DRV instruction is placed in your program.
ERROR	OUT	Bool	When true, this output indicates that an error has occurred and the STATUS output is valid.
STATUS	OUT	Word	The status value of the request indicates the result of the scan or initialization. Additional information is available in the "USS_Extended_Error" variable for some status codes.

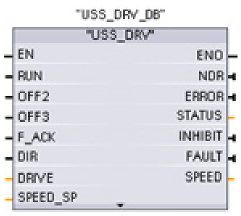
Typically, there is only one USS_PORT instruction per PtP communication port in the program, and each call of this function handles a transmission to or from a single drive. All USS functions associated with one USS network and PtP communication port must use the same instance DB.

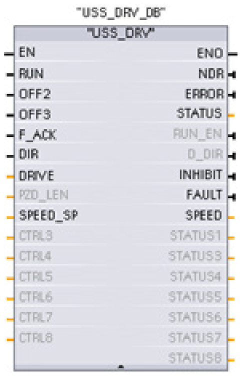
Your program must execute the USS_PORT instruction often enough to prevent drive timeouts. USS_PORT is usually called from a cyclic interrupt OB to prevent drive timeouts and keep the most recent USS data updates available for USS_DRV calls.

13.7.3.2 USS_DRV (Swap data with drive) instruction

Table 13- 118 USS_DRV instruction

LAD / FBD	SCL	Description

Default view
 "USS_DRV_DB"
"USS_DRV"
EN
RUN
OFF2
OFF3
F_ACK
DIR
DRIVE
SPEED_SP
ENO
NDR
ERROR
STATUS
INHIBIT
FAULT
SPEED
	`"USS_DRV_DB"(`
	`RUN:=_bool_in_,`
	`OFF2:=_bool_in_,`
	`OFF3:=_bool_in_,`
	`F_ACK:=_bool_in_,`
	`DIR:=_bool_in_,`
	`DRIVE:=_usint_in_,`
	`PZD_LEN:=_usint_in_,`
	`SPEED_SP:=_real_in_,`
	`CTRL3:=_word_in_,`
	`CTRL4:=_word_in_,`
	`CTRL5:=_word_in_,`
	`CTRL6:=_word_in_,`
	`CTRL7:=_word_in_,`
	`CTRL8:=_word_in_,`
	`NDR=>_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_,`
	`RUN_EN=>_bool_out_,`
	`D_DIR=>_bool_out_,`
	`INHIBIT=>_bool_out_,`
	`FAULT=>_bool_out_,`
	`SPEED=>_real_out_,`
	`STATUS1=>_word_out_,`
	`STATUS3=>_word_out_,`
	`STATUS4=>_word_out_,`
	`STATUS5=>_word_out_,`
	`STATUS6=>_word_out_,`
	`STATUS7=>_word_out_,`
	`STATUS8=>_word_out_);`
		The USS_DRV instruction exchanges data with a drive by creating request messages and interpreting the drive response messages. A separate function block should be used for each drive, but all USS functions associated with one USS network and PtP communication port must use the same instance data block. You must create the DB name when you place the first USS_DRV instruction and then reference the DB that was created by the initial instruction usage.
		STEP 7 automatically creates the DB when you insert the instruction.

Expanded view  "USS_DRV_DB" "USS_DRV" EN RUN OFF2 OFF3 F_ACK DIR DRIVE PZD_LEN SPEED_SP CTRL3 CTRL4 CTRL5 CTRL6 CTRL7 CTRL8 ENO NDR ERROR STATUS RUN_EN D_DIR INHIBIT FAULT SPEED STATUS1 STATUS3 STATUS4 STATUS5 STATUS6 STATUS7 STATUS8		

1 LAD and FBD: Expand the box to reveal all the parameters by clicking the bottom of the box. The parameter pins that are grayed are optional and parameter assignment is not required.

Table 13- 119 Data types for the parameters

Parameter and type		Data type	Description
RUN	IN	Bool	Drive start bit: When true, this input enables the drive to run at the preset speed. When RUN goes to false while a drive is running, the motor will be ramped down to a stop. This behavior differs from the dropping power (OFF2) or braking the motor (OFF3).
OFF2	IN	Bool	Electrical stop bit: When false, this bit cause the drive to coast to a stop with no braking.
OFF3	IN	Bool	Fast stop bit: When false, this bit causes a fast stop by braking the drive rather than just allowing the drive to coast to a stop.
F_ACK	IN	Bool	Fault acknowledge bit: This bit is set to reset the fault bit on a drive. The bit is set after the fault is cleared to indicate to the drive it no longer needs to indicate the previous fault.
DIR	IN	Bool	Drive direction control: This bit is set to indicate that the direction is forward (for positive SPEED_SP).
DRIVE	IN	USInt	Drive address: This input is the address of the USS drive. The valid range is drive 1 to drive 16.
PZD_LEN	IN	USInt	Word length: This is the number of words of PZD data. The valid values are 2, 4, 6, or 8 words. The default value is 2.
SPEED_SP	IN	Real	Speed set point: This is the speed of the drive as a percentage of configured frequency. A positive value specifies forward direction (when DIR is true). Valid range is 200.00 to -200.00.
CTRL3	IN	Word	Control word 3: A value written to a user-configurable parameter on the drive. You must configure this on the drive. (optional parameter)
CTRL4	IN	Word	Control word 4: A value written to a user-configurable parameter on the drive. You must configure this on the drive. (optional parameter)
CTRL5	IN	Word	Control word 5: A value written to a user-configurable parameter on the drive. You must configure this on the drive. (optional parameter)
CTRL6	IN	Word	Control word 6: A value written to a user-configurable parameter on the drive. You must configure this on the drive. (optional parameter)
CTRL7	IN	Word	Control word 7: A value written to a user-configurable parameter on the drive. You must configure this on the drive. (optional parameter)
CTRL8	IN	Word	Control word 8: A value written to a user-configurable parameter on the drive. You must configure this on the drive. (optional parameter)
NDR	OUT	Bool	New data ready: When true, the bit indicates that the outputs contain data from a new communication request.
ERROR	OUT	Bool	Error occurred: When true, this indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported on the USS_PORT instruction ERROR and STATUS outputs.
STATUS	OUT	Word	The status value of the request indicates the result of the scan. This is not a status word returned from the drive.
RUN_EN	OUT	Bool	Run enabled: This bit indicates whether the drive is running.
D_DIR	OUT	Bool	Drive direction: This bit indicates whether the drive is running forward.
INHIBIT	OUT	Bool	Drive inhibited: This bit indicates the state of the inhibit bit on the drive.
FAULT	OUT	Bool	Drive fault: This bit indicates that the drive has registered a fault. You must fix the problem and then set the F_ACK bit to clear this bit when set.

Parameter and type		Data type	Description
SPEED	OUT	Real	Drive Current Speed (scaled value of drive status word 2): The value of the speed of the drive as a percentage of configured speed.
STATUS1	OUT	Word	Drive Status Word 1: This value contains fixed status bits of a drive.
STATUS3	OUT	Word	Drive Status Word 3: This value contains a user-configurable status word on the drive.
STATUS4	OUT	Word	Drive Status Word 4: This value contains a user-configurable status word on the drive.
STATUS5	OUT	Word	Drive Status Word 5: This value contains a user-configurable status word on the drive.
STATUS6	OUT	Word	Drive Status Word 6: This value contains a user-configurable status word on the drive.
STATUS7	OUT	Word	Drive Status Word 7: This value contains a user-configurable status word on the drive.
STATUS8	OUT	Word	Drive Status Word 8: This value contains a user-configurable status word on the drive.

When the initial USS_DRV execution occurs, the drive indicated by the USS address (parameter DRIVE) is initialized in the Instance DB. After this initialization, subsequent executions of USS_PORT can begin communication to the drive at this drive number.

Changing the drive number requires a CPU STOP-to-RUN mode transition that initializes the instance DB. Input parameters are configured into the USS TX message buffer and outputs are read from a "previous" valid response buffer if any exists. There is no data transmission during USS_DRV execution. Drives communicate when USS_PORT is executed. USS_DRV only configures the messages to be sent and interprets data that might have been received from a previous request.

You can control the drive direction of rotation using either the DIR input (Bool) or using the sign (positive or negative) with the SPEED_SP input (Real). The following table indicates how these inputs work together to determine the drive direction, assuming the motor is wired for forward rotation.

Table 13- 120 Interaction of the SPEED_SP and DIR parameters

SPEED_SP	DIR	Drive rotation direction
Value > 0	0	Reverse
Value > 0	1	Forward
Value < 0	0	Forward
Value < 0	1	Reverse

13.7.3.3 USS_RPM (Readout parameters from the drive) instruction

Table 13- 121 USS_RPM instruction

LAD / FBD	SCL	Description

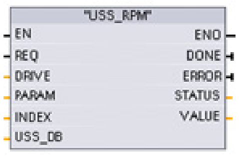
	`USS_RPM(REQ:=_bool_in_,`
	`DRIVE:=_usint_in_,`
	`PARAM:=_uint_in_,`
	`INDEX:=_uint_in_,`
	`DONE=>_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_,`
	`VALUE=>_variant_out_,`
	`USS_DB:=_fbtref_inout_);`
		The USS_RPM instruction reads a parameter from a drive. All USS functions associated with one USS network and PtP communication port must use the same data block. USS_RPM must be called from a main program cycle OB.

Table 13- 122 Data types for the parameters

Parameter type		Data type	Description
REQ	IN	Bool	Send request: When true, REQ indicates that a new read request is desired. This is ignored if the request for this parameter is already pending.
DRIVE	IN	USInt	Drive address: DRIVE is the address of the USS drive. The valid range is drive 1 to drive 16.
PARAM	IN	UInt	Parameter number: PARAM designates which drive parameter is written. The range of this parameter is 0 to 2047. On some drives, the most significant byte can access PARAM values greater than 2047. See your drive manual for details on how to access an extended range.
INDEX	IN	UInt	Parameter index: INDEX designates which Drive Parameter index is to be written. A 16-bit value where the Least Significant Byte is the actual index value with a range of (0 to 255). The Most Significant Byte may also be used by the drive and is drive-specific. See your drive manual for details.
USS_DB	INOUT	USS_BASE	The name of the instance DB that is created and initialized when a USS_DRV instruction is placed in your program.
VALUE	IN	Word, Int, UInt, DWord, DInt, UDInt, Real	This is the value of the parameter that was read and is valid only when the DONE bit is true.
DONE[1]	OUT	Bool	When true, indicates that the VALUE output holds the previously requested read parameter value. This bit is set when USS_DRV sees the read response data from the drive. This bit is reset when either: you request the response data via another USS_RPM poll, or on the second of the next two calls to USS_DRV

Parameter type		Data type	Description
ERROR	OUT	Bool	Error occurred: When true, ERROR indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported on the USS_PORT instruction ERROR and STATUS outputs.
STATUS	OUT	Word	STATUS indicates the result of the read request. Additional information is available in the "USS_Extended_Error" variable for some status codes.

[1] The DONE bit indicates that valid data has been read from the referenced motor drive and delivered to the CPU. It does not indicate that the USS library is capable of immediately reading another parameter. A blank PKW request must be sent to the motor drive and must also be acknowledged by the instruction before the parameter channel for the specific drive becomes available for use. Immediately calling a USS_RPM or USS_WPM FC for the specified motor drive will result in a 0x818A error.

13.7.3.4 USS_WPM (Change parameters in the drive) instruction

Note

EEPROM write operations (for the EEPROM inside a USS drive)

Do not overuse the EEPROM permanent write operation. Minimize the number of EEPROM write operations to extend the EEPROM life.

Table 13- 123 USS_WPM instruction

LAD / FBD	SCL	Description

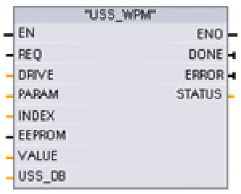
 "USS_WPM"
EN
REQ
DRIVE
PARAM
INDEX
EEPROM
VALUE
USS_DB
ENO
DONE
ERROR
STATUS
	`USS_WPM(REQ:=_bool_in_,`
	`DRIVE:=_usint_in_,`
	`PARAM:=_uint_in_,`
	`INDEX:=_uint_in_,`
	`EEPROM:=_bool_in_,`
	`VALUE:=_variant_in_,`
	`DONE=>_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_,`
	`USS_DB:=_fbtref_inout_);`
		The USS_WPM instruction modifies a parameter in the drive. All USS functions associated with one USS network and PtP communication port must use the same data block.
		USS_WPM must be called from a main program cycle OB.

Table 13- 124 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	Send request: When true, REQ indicates that a new write request is desired. This is ignored if the request for this parameter is already pending.
DRIVE	IN	USInt	Drive address: DRIVE is the address of the USS drive. The valid range is drive 1 to drive 16.
PARAM	IN	UInt	Parameter number: PARAM designates which drive parameter is written. The range of this parameter is 0 to 2047. On some drives, the most significant byte can access PARAM values greater than 2047. See your drive manual for details on how to access an extended range.
INDEX	IN	UInt	Parameter index: INDEX designates which Drive Parameter index is to be written. A 16-bit value where the least significant byte is the actual index value with a range of (0 to 255). The most significant byte may also be used by the drive and is drive-specific. See your drive manual for details.
EEPROM	IN	Bool	Store To Drive EEPROM: When true, a write drive parameter transaction will be stored in the drive EEPROM. If false, the write is temporary and will not be retained if the drive is power cycled.
VALUE	IN	Word, Int, UInt, DWord, DInt, UDInt, Real	The value of the parameter that is to be written. It must be valid on the transition of REQ.
USS_DB	INOUT	USS_BASE	The name of the instance DB that is created and initialized when a USS_DRV instruction is placed in your program.
DONE[1]	OUT	Bool	When true, DONE indicates that the input VALUE has been written to the drive. This bit is set when USS_DRV sees the write response data from the drive. This bit is reset when either you request the response data via another USS_WPM poll, or on the second of the next two calls to USS_DRV
ERROR	OUT	Bool	When true, ERROR indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported on the USS_PORT instruction ERROR and STATUS outputs.
STATUS	OUT	Word	STATUS indicates the result of the write request. Additional information is available in the "USS_Extended_Error" variable for some status codes.

[1] The DONE bit indicates that valid data has been read from the referenced motor drive and delivered to the CPU. It does not indicate that the USS library is capable of immediately reading another parameter. A blank PKW request must be sent to the motor drive and must also be acknowledged by the instruction before the parameter channel for the specific drive becomes available for use. Immediately calling a USS_RPM or USS_WPM FC for the specified motor drive will result in a 0x818A error.

13.7.4 Legacy USS status codes

USS instruction status codes are returned at the STATUS output of the USS functions.

Table 13- 125 STATUS codes [1]

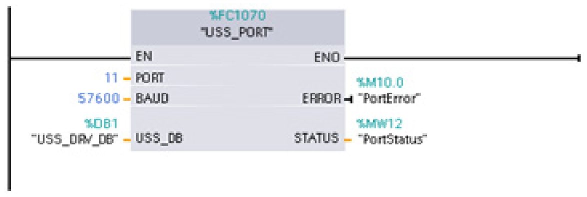
STATUS (W#16#....)	Description
0000	No error
8180	The length of the drive response did not match the characters received from the drive. The drive number where the error occurred is returned in the "USS_Extended_Error" variable. See the extended error description below this table.
8181	VALUE parameter was not a Word, Real or DWord data type.
8182	The user supplied a Word for a parameter value and received a DWord or Real from the drive in the response.
8183	The user supplied a DWord or Real for a parameter value and received a Word from the drive in the response.
8184	The response telegram from drive had a bad checksum. The drive number where the error occurred is returned in the "USS_Extended_Error" variable. See the extended error description below this table.
8185	Illegal drive address (valid drive address range: 1 to16)
8186	The speed set point is out of the valid range (valid speed SP range: -200% to 200%).
8187	The wrong drive number responded to the request sent. The drive number where the error occurred is returned in the "USS_Extended_Error" variable. See the extended error description below this table.
8188	Illegal PZD word length specified (valid range = 2, 4, 6 or 8 words)
8189	Illegal Baud Rate was specified.
818A	The parameter request channel is in use by another request for this drive.
818B	The drive has not responded to requests and retries. The drive number where the error occurred is returned in the "USS_Extended_Error" variable. See the extended error description below this table.
818C	The drive returned an extended error on a parameter request operation. See the extended error description below this table.
818D	The drive returned an illegal access error on a parameter request operation. See your drive manual for information of why parameter access may be limited.
818E	The drive has not been initialized. This error code is returned to USS_RPM or USS_WPM when USS_DRV, for that drive, has not been called at least once. This keeps the initialization on first scan of USS_DRV from overwriting a pending parameter read or write request, since it initializes the drive as a new entry. To fix this error, call USS_DRV for this drive number.
80Ax-80Fx	Specific errors returned from PtP communication FBs called by the USS Library - These error code values are not modified by the USS library and are defined in the PtP instruction descriptions.

[1] In addition to the USS instruction errors listed above, errors can be returned from the underlying PtP communication instructions.

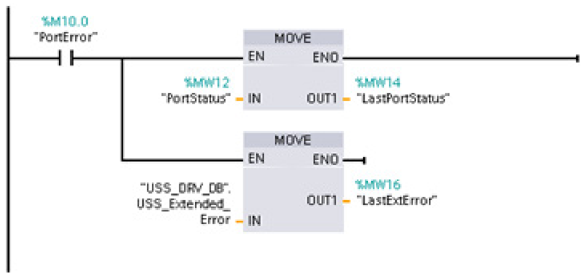
For several STATUS codes, additional information is provided in the "USS_Extended_Error" variable of the USS_DRV Instance DB. For STATUS codes hexadecimal 8180, 8184, 8187, and 818B, USS_Extended_Error contains the drive number where the communication error occurred. For STATUS code hexadecimal 818C, USS_Extended_Error contains a drive error code returned from the drive when using a USS_RPM or USS_WPM instruction.

Example: communication errors reporting

Communication errors (STATUS = 16#818B) are only reported on the USS_PORT instruction and not on the USS_DRV instruction. For example, if the network is not properly terminated then it is possible for a drive to go to RUN but the USS_DRV instruction will show all 0's for the output parameters. In this case, you can only detect the communication error on the USS_PORT instruction. Since this error is only visible for one scan, you will need to add some capture logic as illustrated in the following example. In this example, when the error bit of the USS_PORT instruction is TRUE, then the STATUS and the USS_Extended_Error values are saved into M memory. The drive number is placed in USS_Extended_Error variable when the STATUS code value is hexadecimal 8180, 8184, 8187, or 818B.



Network 1"PortStatus"port status and"USS_DRV_DB".USS_Extended_ Errorextended error code values are only valid for one program scan. The values must be captured for later pro-cessing.



Network 2 The "PortError" contact triggers the storage of the "PortStatus" value in "LastPortStatus" and the"USS_DRV_DB".USS_Extended_E rrorvalue in "LastExtError".

Read and write access to drive internal parameters

USS drives support read and write access to a drive's internal parameters. This feature allows remote control and configuration of the drive. Drive parameter access operations can fail due to errors such as values out of range or illegal requests for a drive's current mode. The drive generates an error code value that is returned in the "USS_Extended_Error" variable. This error code value is only valid for the last execution of a USS_RPM or USS_WPM instruction. The drive error code is put into USS_Extended_Error variable when the STATUS code value is hexadecimal 818C. The error code value of "USS_Extended_Error" depends on the drive model. See the drive's manual for a description of the extended error codes for read and write parameter operations.

13.7.5 Legacy USS general drive setup requirements

Legacy USS general drive setup requirements consist of the following points:

- The drives must be set to use 4 PKW words.
- The drives can be configured for 2, 4, 6, or 8 PZD words.
- The number of PZD word's in the drive must match PZD_LEN input on the USS_DRV instruction for that drive.
- The baud rate in all the drives must match the BAUD input on the USS_PORT instruction.
- The drive must be set for remote control.
- The drive must be set for frequency set-point to USS on COM Link.
- The drive address must be set to 1 to 16 and match the DRIVE input on the USS_DRV block for that drive.
- The drive direction control must be set to use the polarity of the drive set-point.
- The RS485 network must be terminated properly.

USS general drive connection and setup is the same for USS instructions (V4.1) and legacy USS instructions (V4.0 and earlier). Refer to the Example: USS general drive connection and setup (Page 927) for further information.

13.8 Legacy Modbus TCP communication

13.8.1 Overview

Prior to the release of STEP 7 V13 SP1 and the S7-1200 V4.1 CPUs, the Modbus TCP communication instructions existed with different names, and in some cases, slightly different interfaces. The general concepts apply to both sets of instructions. Refer to the individual legacy Modbus TCP instructions for programming information.

13.8.2 Selecting the version of the Modbus TCP instructions

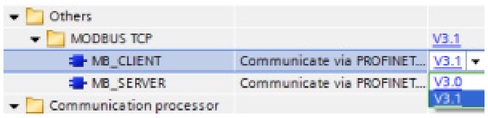
There are two versions of the Modbus TCP instructions available in STEP 7:

- Version 3.0 was initially available in STEP 7 Basic/Professional V13.
- Version 3.1 is available in STEP 7 Basic/Professional V13, SP1.

For compatibility and ease of migration, you can choose which instruction version to insert into your user program.

Do not use both 3.0 and 3.1 instruction versions in the same CPU program. Your program's Modbus TCP instructions must have the same major version number (**1**.x, **2**.y, or **V**.z). The individual instructions within a major version group may have different minor versions (1.**x**).

Click the icon on the instruction tree task card to enable the headers and columns of the instruction tree.



Others		
MODBUS TCP		V3.1
MB_CLIENT	Communicate via PROFINET...	V3.1
MB_SERVER	Communicate via PROFINET...	V3.0
Communication processor		V3.1

To change the version of the Modbus TCP instructions, select the version from the drop-down list. You can select the group or individual instructions.

When you use the instruction tree to place a Modbus TCP instruction in your program, a new FB instance is created in the project tree. You can see new FB instance in the project tree under PLC_x > Program blocks > System blocks > Program resources.

To verify the version of a Modbus TCP instruction in a program, you must inspect project tree properties and not the properties of a box displayed in the program editor. Select a project tree Modbus TCP FB instance, right-click, select "Properties", and select the "Information" page to see the Modbus TCP instruction version number.

13.8.3 Legacy Modbus TCP instructions

13.8.3.1 MB_CLIENT (Communicate using PROFINET as Modbus TCP client) instruction

Table 13- 126 MB_CLIENT instruction

LAD / FBD	SCL	Description

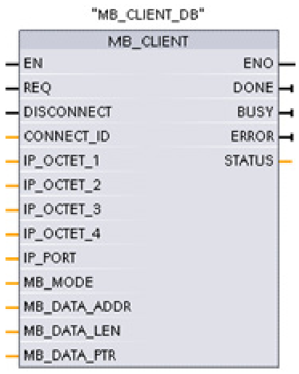
 "MB_CLIENT_DB"
MB_CLIENT
EN
ENO
REQ
DONE
DISCONNECT
BUSY
CONNECT_ID
ERROR
IP_OCTET_1
STATUS
IP_OCTET_2
IP_OCTET_3
IP_OCTET_4
IP_PORT
MB_MODE
MB_DATA_ADDR
MB_DATA_LEN
MB_DATA_PTR
	`"MB_CLIENT_DB"(`
	`    REQ:=_bool_in_,`
	`    DISCONNECT:=_bool_in_,`
	`    CONNECT_ID=_uint_in_,`
	`    IP_OCTET_1:=_byte_in_,`
	`    IP_OCTET_2:=_byte_in_,`
	`    IP_OCTET_3:=_byte_in_,`
	`    IP_OCTET_4:=_byte_in_,`
	`    IP_PORT:=_uint_in_,`
	`    MB_MODE:=_usint_in_,`
	`    MB_DATA_ADDR:=_udint_in_,`
	`    MB_DATA_LEN:=_uint_in_,`
	`    DONE=>_bool_out_,`
	`    BUSY=>_bool_out_,`
	`    ERROR=>_bool_out_,`
	`    STATUS=>_word_out_,`
	`    MB_DATA_PTR:=_variant_inout_);`
		MB_CLIENT communicates as a Modbus TCP client through the PROFINET connector on the S7-1200 CPU. No additional communication hardware module is required.
		MB_CLIENT can make a client-server connection, send a Modbus function request, receive a response, and control the disconnection from a Modbus TCP server.

Table 13- 127 Data types for the parameters

Parameter and type		Data type	Description
REQ	In	Bool	FALSE = No Modbus communication request TRUE = Request to communicate with a Modbus TCP server
DISCONNECT	IN	Bool	The DISCONNECT parameter allows your program to control connection and disconnection with a Modbus server device. If DISCONNECT = 0 and a connection does not exist, then MB_CLIENT attempts to make a connection to the assigned IP address and port number. If DISCONNECT = 1 and a connection exists, then a disconnect operation is attempted. Whenever this input is enabled, no other operation will be attempted.
CONNECT_ID	IN	UInt	The CONNECT_ID parameter must uniquely identify each connection within the PLC. Each unique instance of the MB_CLIENT or MB_SERVER instruction must contain a unique CONNECT_ID parameter.
IP_OCTET_1	IN	USInt	Modbus TCP server IP address: Octet 1 8-bit part of the 32-bit IPv4 IP address of the Modbus TCPserver to which the client will connect and communicate using the Modbus TCP protocol.
IP_OCTET_2	IN	USInt	Modbus TCP server IP address: Octet 2
IP_OCTET_3	IN	USInt	Modbus TCP server IP address: Octet 3
IP_OCTET_4	IN	USInt	Modbus TCP server IP address: Octet 4
IP_PORT	IN	UInt	Default value = 502: The IP port number of the server to which the client will attempt to connect and ultimately communicate using the TCP/IP protocol.

Parameter and type		Data type	Description
MB_MODE	IN	USInt	Mode Selection: Assigns the type of request (read, write, or diagnostic). See the Modbus functions table below for details.
MB_DATA_ADDR	IN	UDInt	Modbus starting Address: Assigns the starting address of the data to be accessed by MB_CLIENT. See the Modbus functions table below for valid addresses.
MB_DATA_LEN	IN	UInt	Modbus data Length: Assigns the number of bits or words to be accessed in this request. See the Modbus functions table below for valid lengths
MB_DATA_PTR	IN_OUT	Variant	Pointer to the Modbus data register: The register buffers data going to or coming from a Modbus server. The pointer must assign a standard global DB or a M memory address.
DONE	OUT	Bool	The DONE bit is TRUE for one scan, after the last request was completed with no error.
BUSY	OUT	Bool	• 0 - No MB_CLIENT operation in progress • 1 - MB_CLIENT operation in progress
ERROR	OUT	Bool	The ERROR bit is TRUE for one scan, after the MB_CLIENT execution was terminated with an error. The error code value at the STATUS parameter is valid only during the single cycle where ERROR = TRUE.
STATUS	OUT	Word	Execution condition code

REQ parameter

FALSE = No Modbus communication request
TRUE = Request to communicate with a Modbus TCP server

If no instance of MB_CLIENT is active and parameter DISCONNECT=0, when REQ=1 a new Modbus request will start. If the connection is not already established then a new connection will be made.

If the same instance of MB_CLIENT is executed again with DISCONNECT=0 and REQ=1, before the completion of the current request, then no subsequent Modbus transmission will be made. However, as soon as the current request is completed, a new request can be processed if MB_CLIENT is executed with REQ=1.

When the current MB_CLIENT communication request is complete, the DONE bit is TRUE for one cycle. The DONE bit can be used as a time gate to sequence multiple MB_CLIENT requests.

Note

Input data consistency during MB_CLIENT processing

Once a Modbus client initiates a Modbus operation, all the input states are saved internally and are then compared on each successive call. The comparison is used to determine if this particular call was the originator of the active client request. More than one MB_CLIENT call can be performed using a common instance DB.

As a result, it is important that the inputs are not changed during the period of time that a MB_CLIENT operation is actively being processed. If this rule is not followed, then a MB_CLIENT cannot determine that it is the active instance.

MB_MODE and MB_DATA_ADDR parameters select the Modbus communication function

MB_DATA_ADDR assigns the starting Modbus address of the data to be accessed. The MB_CLIENT instruction uses a MB_MODE input rather than a function code input.

The combination of MB_MODE and MB_DATA_ADDR values determine the function code that is used in the actual Modbus message. The following table shows the correspondence between parameter MB_MODE, Modbus function, and Modbus address range.

Table 13- 128 Modbus functions

MB_MODE	Modbus function	Data length	Operation and data	MB_DATA_ADDR
0	01	1 to 2000	Read output bits: 1 to 2000 bits per request	1 to 9999
0	02	1 to 2000	Read input bits: 1 to 2000 bits per request	10001 to 19999
0	03	1 to 125	Read Holding registers: 1 to 125 words per request	40001 to 49999 or 400001 to 465535
0	04	1 to 125	Read input words: 1 to 125 words per request	30001 to 39999
1	05	1	Write one output bit: One bit per request	1 to 9999
1	06	1	Write one holding register: 1 word per request	40001 to 49999 or 400001 to 465535
1	15	2 to 1968	Write multiple output bits: 2 to 1968 bits per request	1 to 9999
1	16	2 to 123	Write multiple holding registers: 2 to 123 words per request	40001 to 49999 or 400001 to 465535
2	15	1 to 1968	Write one or more output bits: 1 to 1968 bits per request	1 to 9999
2	16	1 to 123	Write one or more holding registers: 1 to 123 words per request	40001 to 49999 or 400001 to 465535
11	11	0	Read the server communication status word and event counter. The status word indicates busy (0 – not busy, 0xFFFF - busy). The event counter is incremented for each successful completion of a message. Both the MB_DATA_ADDR and MB_DATA_LEN parameters of MB_CLIENT are ignored for this function.	
80	08	1	Check server status using data diagnostic code 0x0000 (Loopback test – server echoes the request) 1 word per request	

MB_MODE	Modbus function	Data length	Operation and data	MB_DATA_ADDR
81	08	1	Reset server event counter using data diagnostic code 0x000A 1 word per request	
3 to 10, 12 to 79, 82 to 255			Reserved	

Note

MB_DATA_PTR assigns a buffer to store data read/written to/from a Modbus TCP server

The data buffer can be in a standard global DB or M memory address.

For a buffer in M memory, use the standard Any Pointer format. This is in the format P#"Bit Address" "Data Type" "Length", an example would be P#M1000.0 WORD 500.

MB_DATA_PTR assigns a communication buffer

- MB_CLIENT communication functions:
  - Read and write 1-bit data from Modbus server addresses (00001 to 09999)
  - Read 1-bit data from Modbus server addresses (10001 to 19999)
  - Read 16-bit word data from Modbus server addresses (30001 to 39999) and (40001 to 49999)
  - Write 16-bit word data to Modbus server addresses (40001 to 49999)
- Word or bit sized data is transferred to/from the DB or M memory buffer assigned by MB_DATA_PTR.
- If a DB is assigned as the buffer by MB_DATA_PTR, then you must assign data types to all DB data elements.
  - The 1-bit Bool data type represents one Modbus bit address
  - 16-bit single word data types like WORD, UInt, and Int represent one Modbus word address
  - 32-bit double word data types like DWORD, DInt, and Real represent two Modbus word addresses

- Complex DB elements can be assigned by MB_DATA_PTR, such as
  - Standard arrays
  - Named structures where each element is unique.
  - Named complex structures where each element has a unique name and a 16 or 32 bit data type.
- There is no requirement that the MB_DATA_PTR data areas be in the same global data block (or M memory area). You can assign one data block for Modbus reads, another data block for Modbus writes, or one data block for each MB_CLIENT station.

Multiple client connections

A Modbus TCP client can support concurrent connections up to the maximum number of Open User Communications connections allowed by the PLC. The total number of connections for a PLC, including Modbus TCP Clients and Servers, must not exceed the maximum number of supported Open User Communications connections (Page 613). The Modbus TCP connections may be shared between Client and/or Server type connections.

Individual client connections must follow these rules:

- Each MB_CLIENT connection must use a distinct instance DB
- Each MB_CLIENT connection must specify a unique server IP address
- Each MB_CLIENT connection must specify a unique connection ID
- Unique IP port numbers may or may not be required depending upon the server configuration

The Connection ID must be unique for each individual connection. This means a single, unique Connection ID must only be used with each individual instance DB. In summary, the instance DB and the Connection ID are paired together and must be unique for every connection.

Table 13- 129 MB_CLIENT instance data block user accessible static variables

Variable	Data type	Default	description
Blocked_Proc_Timeout	Real	3.0	Amount of time (in seconds) to wait upon a blocked Modbus client instance before removing this instance as being ACTIVE. This can occur, for example, when a client request has been issued and then application stops executing the client function before it has completely finished the request. The maximum S7-1200 limit is 55 seconds.
MB_Unit_ID	Word	255	Modbus unit identifier: A Modbus TCP server is addressed using its IP address. As a result, the MB_UNIT_ID parameter is not used for Modbus TCP addressing. The MB_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 initialized within a restricted range of values.
RCV_TIMEOUT	Real	2.0	Time in seconds that the MB_CLIENT waits for a server to respond to a request.
Connected	Bool	0	Indicates whether the connection to the assigned server is connected or disconnected: 1=connected, 0=disconnected

Table 13- 130 MB_CLIENT protocol errors

STATUS (W#16#)	Response code to Modbus client (B#16#)	Modbus protocol errors
8381	01	Function code not supported
8382	03	Data length error
8383	02	Data address error or access outside the bounds of the MB_HOLD_REG address area
8384	03	Data value error
8385	03	Data diagnostic code value not supported (function code 08)

Table 13- 131 MB_CLIENT execution condition codes [1]

STATUS (W#16#)	MB_CLIENT parameter errors
7001	MB_CLIENT is waiting for a Modbus server response to a connect or disconnect request, on the assigned TCP port. This is only reported for the first execution of a connect or disconnect operation.
7002	MB_CLIENT is waiting for a Modbus server response to a connect or disconnect request, for the assigned TCP port. This will be reported for any subsequent executions, while waiting for completion of a connect or disconnect operation.
7003	A disconnect operation has successfully completed (Only valid for one PLC scan).

STATUS (W#16#)	MB_CLIENT parameter errors
80C8	The server did not respond in the assigned time. MB_CLIENT must receive a response using the transaction ID that was originally transmitted within the assigned time or this error is returned. Check the connection to the Modbus server device. This error is only reported after any configured retries (if applicable) have been attempted.
8188	Invalid mode value
8189	Invalid data address value
818A	Invalid data length value
818B	Invalid pointer to the DATA_PTR area. This can be the combination of MB_DATA_ADDRESS + MB_DATA_LEN.
818C	Pointer to a optimized DATA_PTR area (must be a standard DB area or M memory area)
8200	The port is busy processing an existing Modbus request.
8380	Received Modbus frame is malformed or too few bytes have been received.
8387	The assigned Connection ID parameter is different from the ID used for previous requests. There can only be a single Connection ID used within each MB_CLIENT instance DB. This is also used as an internal error if the Modbus TCP protocol ID received from a server is not 0.
8388	A Modbus server returned a quantity of data that is different than what was requested. This applies to Modbus functions 15 or 16 only.

[1] In addition to the MB_CLIENT errors listed above, errors can be returned from the underlying T block communication instructions (TCON, TDISCON, TSEND, and TRCV (Page 664)).

13.8.3.2 MB_SERVER (Communicate using PROFINET as Modbus TCP server) instruction

The "MB_SERVER" instruction communicates as Modbus TCP server through the PROFINET connector on the S7-1200 CPU. The "MB_SERVER" instruction processes connection requests of a Modbus TCP client, receives and processes Modbus requests, and sends responses.

To use the instruction, you do not require an additional hardware module.

NOTICE

Security information

Note that each client of the network is given read and write access to the process image inputs and outputs and to the data block or bit memory area defined by the Modbus holding register.

The option is available to restrict access to an IP address to prevent unauthorized read and write operations. Note, however, that the shared address can also be used for unauthorized access.

Table 13- 132 MB_SERVER instruction

LAD / FBD	SCL	Description

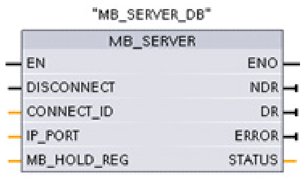
 "MB_SERVER_DB"
MB_SERVER
EN
ENO
DISCONNECT
NDR
CONNECT_ID
DR
IP_PORT
ERROR
MB_HOLD_REG
STATUS
	`"MB_SERVER_DB"(`
	`    DISCONNECT:=_bool_in_,`
	`    CONNECT_ID:=_uint_in_,`
	`    IP_PORT:=_uint_in_,`
	`    NDR=>_bool_out_,`
	`    DR=>_bool_out_,`
	`    ERROR=>_bool_out_,`
	`    STATUS=>_word_out_,`
	`    MB_HOLD_REG:=_variant_inout_);`
		MB_SERVER communicates as a Modbus TCP server through the PROFINET connector on the S7-1200 CPU. No additional communication hardware module is required.
		MB_SERVER can accept a request to connect with Modbus TCP client, receive a Modbus function request, and send a response message.

Table 13- 133 Data types for the parameters

Parameter and type		Data type	Description
DISCONNECT	IN	Bool	MB_SERVER attempts to make a "passive" connection with a partner device. This means that the server is passively listening for a TCP connection request from any requesting IP address. If DISCONNECT = 0 and a connection does not exist, then a passive connection can be initiated. If DISCONNECT = 1 and a connection exists, then a disconnect operation is initiated. This allows your program to control when a connection is accepted. Whenever this input is enabled, no other operation will be attempted.
CONNECT_ID	IN	UInt	CONNECT_ID uniquely identifies each connection within the PLC. Each unique instance of the MB_CLIENT or MB_SERVER instruction must contain a unique CONNECT_ID parameter.
IP_PORT	IN	UInt	Default value = 502: The IP port number that identifies the IP port that will be monitored for a connection request from a Modbus client. These TCP port numbers are not allowed for a MB_SERVER passive connection: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963, and 34964.
MB_HOLD_REG	IN_OUT	Variant	Pointer to the MB_SERVER Modbus holding register: The holding register must either be a standard global DB or a M memory address. This memory area is used to hold the values a Modbus client is allowed to access using Modbus register functions 3 (read), 6 (write), and 16 (write).
NDR	OUT	Bool	New Data Ready: 0 = No new data, 1 = Indicates that new data has been written by a Modbus client
DR	OUT	Bool	Data Read: 0 = No data read, 1 = Indicates that data has been read by a Modbus client.
ERROR	OUT	Bool	The ERROR bit is TRUE for one scan, after MB_SERVER execution was terminated with an error. The error code value at the STATUS parameter is valid only during the single cycle where ERROR = TRUE.
STATUS	OUT	Word	Execution condition code

MB_SERVER allows incoming Modbus function codes (1, 2, 4, 5, and 15) to read or write bits and words directly in the input process image and output process image of the S7-1200 CPU. For data transfer function codes (3, 6, and 16), the MB_HOLD_REG parameter must be defined as a data type larger than a byte. The following table shows the mapping of Modbus addresses to the process image in the CPU.

Table 13- 134 Mapping of Modbus addresses to the process image

Modbus functions						**S7-1200**	
Codes	Function	Data area	Address range			Data area	CPU address
01	Read bits	Output	1	To	8192	Output Process Image	Q0.0 to Q1023.7
02	Read bits	Input	10001	To	18192	Input Process Image	I0.0 to I1023.7
04	Read words	Input	30001	To	30512	Input Process Image	IW0 to IW1022
05	Write bit	Output	1	To	8192	Output Process Image	Q0.0 to Q1023.7
15	Write bits	Output	1	To	8192	Output Process Image	Q0.0 to Q1023.7

Incoming Modbus message function codes function codes (3, 6, and 16) read or write words in a Modbus holding register which can be an M memory address range or a data block. The type of holding register is specified by the MB_HOLD_REG parameter.

Note

MB_HOLD_REG parameter assignment

The Modbus Holding Register can be in a standard global DB or an M memory address.

For A Modbus holding register in M memory, use the standard Any Pointer format. This is in the format P#"Bit Address" "Data Type" "Length". An example would be P#M1000.0 WORD 500

The following table shows examples of Modbus address to holding register mapping used for Modbus function codes 03 (read words), 06 (write word), and 16 (write words). The actual upper limit of DB addresses is determined by the maximum work memory limit and M memory limit, for each CPU model.

Table 13- 135 Mapping examples of Modbus address to CPU memory address

Modbus Address	**MB_HOLD_REG parameter examples**		
	P#M100.0 Word 5	**P#DB10.DBx0.0 Word 5**	**"Recipe".ingredient**
40001	MW100	DB10.DBW0	"Recipe".ingredient[1]
40002	MW102	DB10.DBW2	"Recipe".ingredient[2]
40003	MW104	DB10.DBW4	"Recipe".ingredient[3]
40004	MW106	DB10.DBW6	"Recipe".ingredient[4]
40005	MW108	DB10.DBW8	"Recipe".ingredient[5]

Multiple server connections

Multiple server connections may be created. This permits a single PLC to establish concurrent connections to multiple Modbus TCP clients.

A Modbus TCP server can support concurrent connections up to the maximum number of Open User Communications connections allowed by the PLC. The total number of connections for a PLC, including Modbus TCP Clients and Servers, must not exceed the maximum number of supported Open User Communications connections (Page 613). The Modbus TCP connections may be shared between Client and/or Server type connections.

Individual server connection must follow these rules:

- Each MB_SERVER connection must use a distinct instance DB.
- Each MB_SERVER connection must be established with a unique IP port number. Only 1 connection per port is supported.
- Each MB_SERVER connection must use a unique connection ID.
- The MB_SERVER must be called individually for each connection (with its respective instance DB).

The Connection ID must be unique for each individual connection. This means a single, unique Connection ID must only be used with each individual instance DB. In summary, the instance DB and the Connection ID are paired together and must be unique for every connection.

Table 13- 136 Modbus diagnostic function codes

MB_SERVER Modbus diagnostic functions		
Codes	Sub-function	Description
08	0x0000	Return query data echo test: The MB_SERVER will echo back to a Modbus client a word of data that is received.
08	0x000A	Clear communication event counter: The MB_SEVER will clear out the communication event counter that is used for Modbus function 11.
11		Get communication event counter: The MB_SERVER uses an internal communication event counter for recording the number of successful Modbus read and write requests that are sent to the Modbus server. The counter does not increment on any Function 8 or Function 11 requests. It is also not incremented on any requests that result in a communication error. The broadcast function is not available for Modbus TCP, because only one client-server connection exists at any one time.

MB_SERVER variables

This table shows the public static variables stored in the MB_SERVER instance data block that can be used in your program

Table 13- 137 MB_SERVER public static variables

Variable	Data type	Default value	Description
HR_Start_Offset	Word	0	Assigns the starting address of the Modbus Holding register
Request_Count	Word	0	The number of all requests received by this server.
Server_Message_Count	Word	0	The number of requests received for this specific server.
Xmt_Rcv_Count	Word	0	The number of transmissions or receptions that have encountered an error. Also, incremented if a message is received that is an invalid Modbus message.
Exception_Count	Word	0	Modbus specific errors that require a returned exception
Success_Count	Word	0	The number of requests received for this specific server that ha no protocol errors.
Connected	Bool	0	Indicates whether the connection to the assigned client is connected or disconnected: 1=connected, 0=disconnected

Your program can write values to the HR_Start_Offset and control Modbus server operations. The other variables can be read to monitor Modbus status.

HR_Start_Offset

Modbus holding register addresses begin at 40001. These addresses correspond to the beginning PLC memory address of the holding register. However, you can configure the "HR_Start_Offset" variable to start the beginning Modbus holding register address at another value instead of 40001.

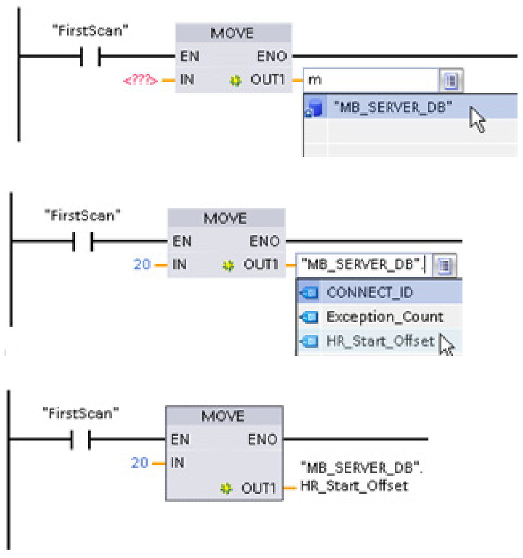
For example, if the holding register is configured to start at MW100 and is 100 words long. An offset of 20 specifies a beginning holding register address of 40021 instead of 40001. Any address below 40021 and above 40119 will result in an addressing error.

Table 13- 138 Example of Modbus holding register addressing

HR_Start_Offset	Address	Minimum	Maximum
0	Modbus address (Word)	40001	40099
	S7-1200 address	MW100	MW298
20	Modbus address (Word)	40021	40119
	S7-1200 address	MW100	MW298

HR_Start_Offset is a word value that specifies the starting address of the Modbus holding register and is stored in the MB_SERVER instance data block. You can set this public static variable value by using the parameter helper drop-list, after MB_SERVER is placed in your program.

For example, after MB_SERVER is placed in a LAD network, you can go to a previous network and assign the HR_Start_Offset value. The value must be assigned prior to execution of MB_SERVER.



Entering a Modbus server variable using the default DB name:

1. Set the cursor in the parameter field and type an m character.
2. Select "MB_SERVER_DB" from the drop-list of DB names.
3. Select "MB_SERVER_DB.HR_Start_Offset" from the drop-list of DB variables.

Table 13- 139 MB_SERVER execution condition codes [1]

STATUS (W#16#)	Response code to Modbus server (B#16#)	Modbus protocol errors
7001		MB_SERVER is waiting for a Modbus client to connect to the assigned TCP port. This code is reported on the first execution of a connect or disconnect operation.
7002		MB_SERVER is waiting for a Modbus client to connect to the assigned TCP port. This code is reported for any subsequent executions, while waiting for completion of a connect or disconnect operation.
7003		A disconnect operation has successfully completed (Only valid for one PLC scan).
8187		Invalid pointer to MB_HOLD_REG: area is too small
818C		Pointer to an optimized MB_HOLD_REG area (must be a standard DB area or M memory area) or Blocked process timeout exceeds the limit of 55 seconds. (S7-1200 specific)
8381	01	Function code not supported
8382	03	Data length error
8383	02	Data address error or access outside the bounds of the MB_HOLD_REG address area
8384	03	Data value error
8385	03	Data diagnostic code value not supported (function code 08)

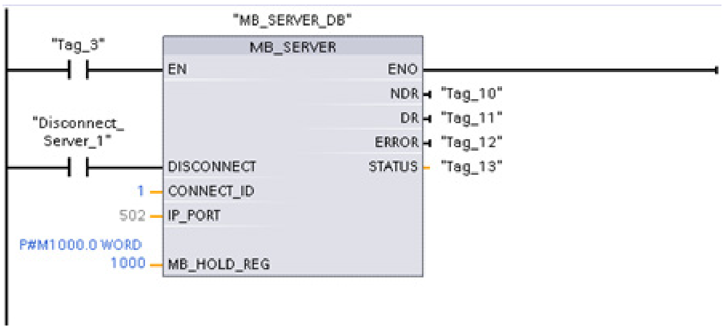
[1] In addition to the MB_SERVER errors listed above, errors can be returned from the underlying T block communication instructions (TCON, TDISCON, TSEND, and TRCV (Page 664)).

13.8.4 Example: Legacy MB_SERVER Multiple TCP connections

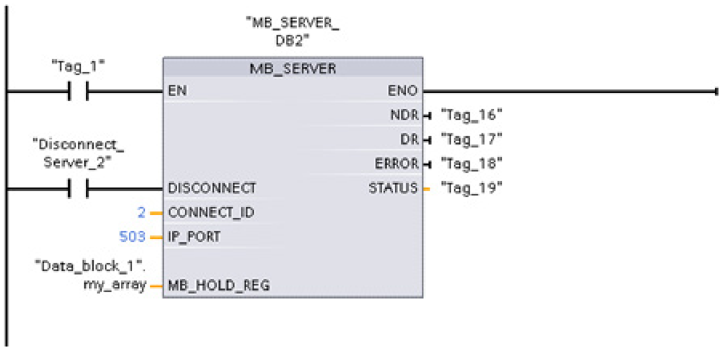
You can have multiple Modbus TCP server connections. To accomplish this, MB_SERVER must be independently executed for each connection. Each connection must use an independent instance DB, connection ID, and IP port. The S7-1200 allows only one connection per IP port.

For best performance, MB_SERVER should be executed every program cycle, for each connection.

Network 1: Connection #1 with independent IP_PORT, connection ID, and instance DB



Network 2: Connection #2 with independent IP_PORT, connection ID, and instance DB



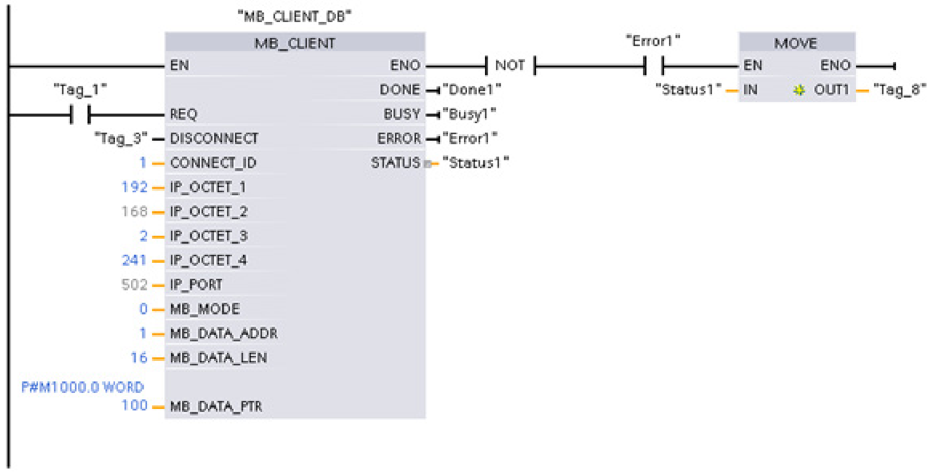
13.8.5 Example: Legacy MB_CLIENT 1: Multiple requests with common TCP connection

Multiple Modbus client requests can be sent over the same connection. To accomplish this, use the same instance DB, connection ID, and port number.

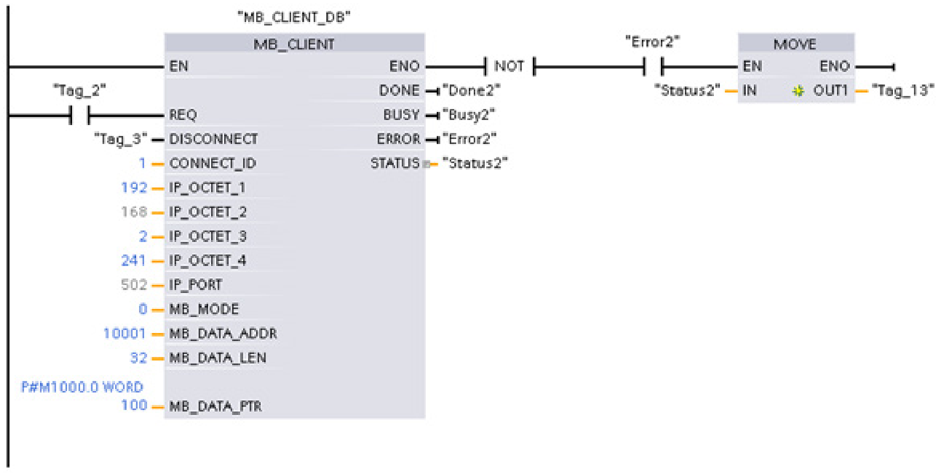
Only 1 client can be active at any given time. Once a client completes its execution, the next client begins execution. Your program is responsible for the order of execution.

The example shows both clients writing to the same memory area. Also, a returned error is captured which is optional.

Network 1: Modbus function 1 - Read 16 output image bits



Network 2: Modbus function 2 - Read 32 input image bits



13.8.6 Example: Legacy MB_CLIENT 2: Multiple requests with different TCP connections

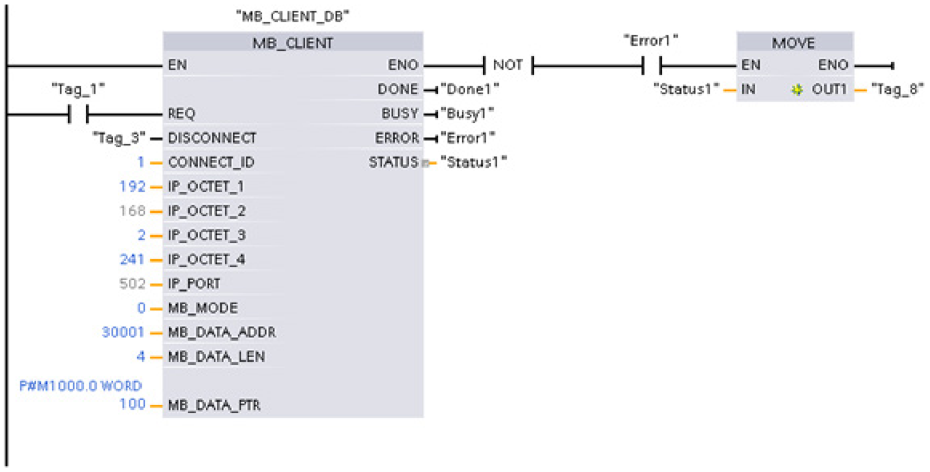
Modbus client requests can be sent over different connections. To accomplish this, different instance DBs, IP addresses, and connection IDs must be used.

The port number must be different if the connections are established to the same Modbus server. If the connections are on different servers, there is no port number restriction.

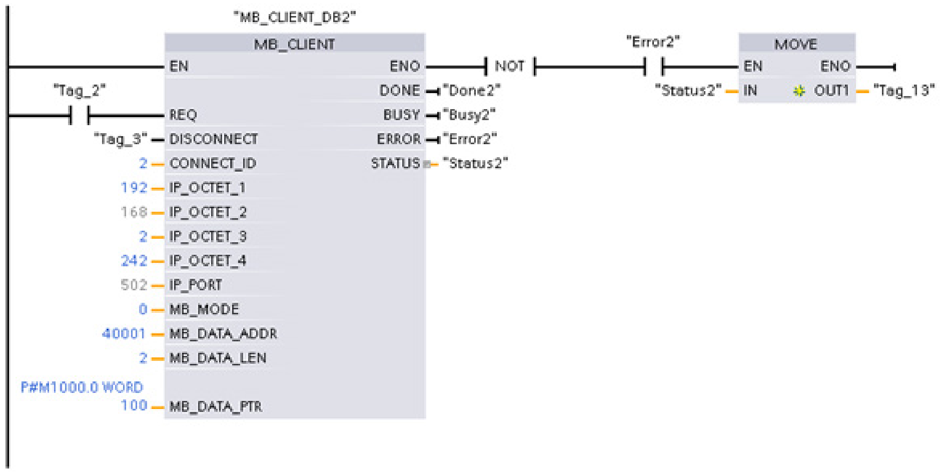
The example shows both clients writing to the same memory area. Also, a returned error is captured which is optional.

Network 1:

Modbus function 4 - Read input words (in S7-1200 memory)



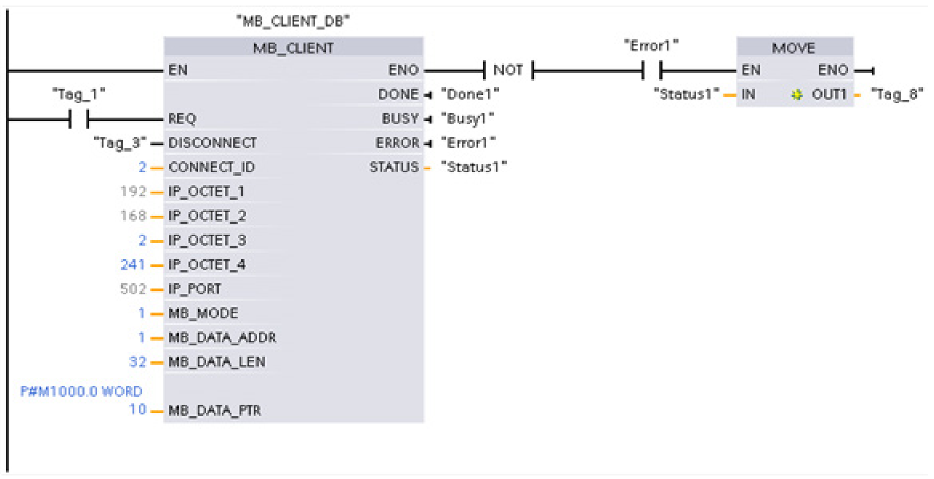
Network 2: Modbus function 3 - Read holding register words from a Modbus TCP server



13.8.7 Example: Legacy MB_CLIENT 3: Output image write request

This example shows a Modbus client request to write the S7-1200 output image.

Network 1: Modbus function 15 - Write S7-1200 output image bits

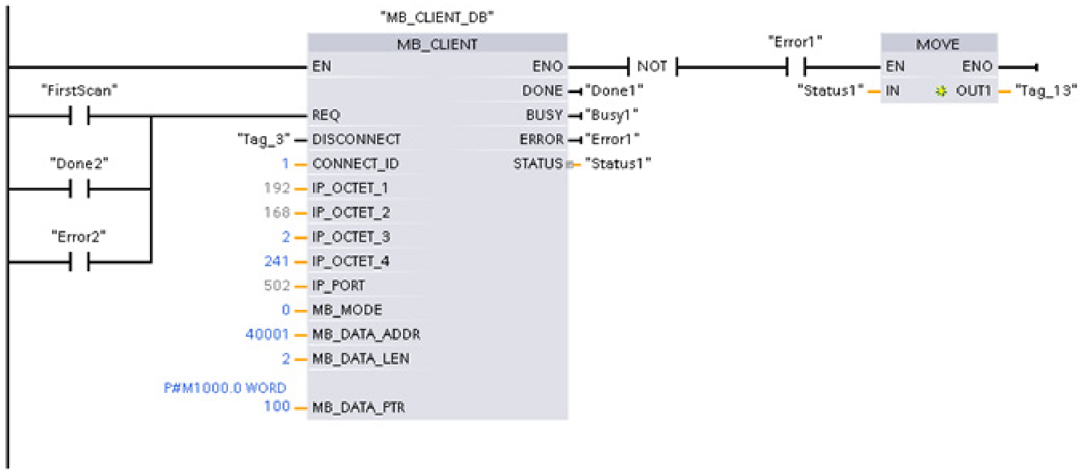


13.8.8 Example: Legacy MB_CLIENT 4: Coordinating multiple requests

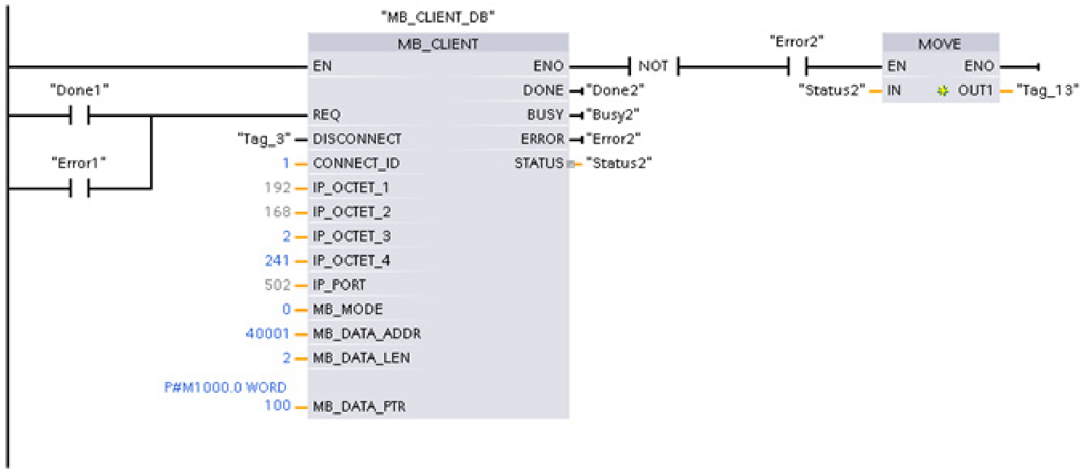
You must ensure that each individual Modbus TCP request finishes execution. This coordination must be provided by your program. The example below shows how the outputs of the first and second client requests can be used to coordinate execution.

The example shows both clients writing to the same memory area. Also, a returned error is captured which is optional.

Network 1: Modbus function 3 - Read holding register words



Network 2: Modbus function 3 - Read holding register words



13.9 Legacy Modbus RTU communication (CM/CB 1241 only)

13.9.1 Overview

Prior to the release of STEP 7 V13 SP1 and the S7-1200 V4.1 CPUs, the Modbus RTU communication instructions existed with different names, and in some cases, slightly different interfaces. The general concepts apply to both sets of instructions. Refer to the individual legacy Modbus RTU instructions for programming information.

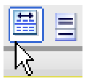
13.9.2 Selecting the version of the Modbus RTU instructions

There are two versions of the Modbus RTU instructions available in STEP 7:

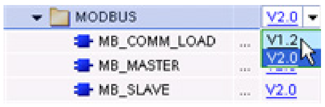
- Version 1 was initially available in STEP 7 Basic V10.5.
- Version 2 is available in STEP 7 Basic/Professional V11. The version 2 design adds REQ and DONE parameters to MB_COMM_LOAD. Also, the MB_ADDR parameter for MB_MASTER and MB_SLAVE now allows a UInt value for extended addressing.

For compatibility and ease of migration, you can choose which instruction version to insert into your user program.

You cannot use both versions of the instructions with the same module, but two different modules can use different versions of the instructions. Your program's Modbus RTU instructions must have the same major version number (**1**.x, **2**.y, or **V**.z). The individual instructions within a major version group may have different minor versions (1.**x**).



Click the icon on the instruction tree task card to enable the headers and columns of the instruction tree.



To change the version of the Modbus instructions, select the version from the drop-down list. You can select the group or individual instructions.

When you use the instruction tree to place a Modbus instruction in your program, a new FB instance is created in the project tree. You can see new FB instance in the project tree under PLC_x > Program blocks > System blocks > Program resources.

To verify the version of a Modbus instruction in a program, you must inspect project tree properties and not the properties of a box displayed in the program editor. Select a project tree Modbus FB instance, right-click, select "Properties", and select the "Information" page to see the Modbus instruction version number.

13.9.3 Legacy Modbus RTU instructions

13.9.3.1 MB_COMM_LOAD (Configure port on the PtP module for Modbus RTU) instruction

Table 13- 140 MB_COMM_LOAD instruction

LAD / FBD	SCL	Description

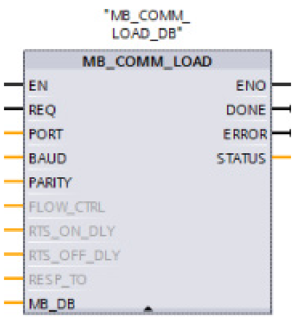
 "MB_COMM_LOAD_DB"
MB_COMM_LOAD
EN
REQ
PORT
BAUD
PARITY
FLOW_CTRL
RTS_ON_DLY
RTS_OFF_DLY
RESP_TO
MB_DB
ENO
DONE
ERROR
STATUS
	`"MB_COMM_LOAD_DB"(`
	`    REQ:=_bool_in,`
	`    PORT:=_uint_in_,`
	`    BAUD:=_udint_in_,`
	`    PARITY:=_uint_in_,`
	`    FLOW_CTRL:=_uint_in_,`
	`    RTS_ON_DLY:=_uint_in_,`
	`    RTS_OFF_DLY:=_uint_in_,`
	`    RESP_TO:=_uint_in_,`
	`    DONE=>_bool_out,`
	`    ERROR=>_bool_out_,`
	`    STATUS=>_word_out_,`
	`    MB_DB:=_fbtref_inout_);`
		The MB_COMM_LOAD instruction configures a PtP port for Modbus RTU protocol communications. Modbus port hardware options: Install up to three CMs (RS485 or RS232), plus one CB (R4845). An instance data block is assigned automatically when you place the MB_COMM_LOAD instruction in your program.

Table 13- 141 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	A low to high (positive edge) signal starts the operation. (Version 2.0 only)
PORT	IN	Port	After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table.
BAUD	IN	UDInt	Baud rate selection: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200, all other values are invalid
PARITY	IN	UInt	Parity selection: • 0 – None • 1 – Odd • 2 – Even
FLOW_CTRL [1]	IN	UInt	Flow control selection: • 0 – (default) no flow control • 1 – Hardware flow control with RTS always ON (does not apply to RS485 ports) • 2 – Hardware flow control with RTS switched

Parameter and type		Data type	Description
RTS_ON_DLY [1]	IN	UInt	RTS ON delay selection: • 0 – (default) No delay from RTS active until the first character of the message is transmitted • 1 to 65535 – Delay in milliseconds from RTS active until the first character of the message is transmitted (does not apply to RS485 ports). RTS delays shall be applied independent of the FLOW_CTRL selection.
RTS_OFF_DLY [1]	IN	UInt	RTS OFF delay selection: • 0 – (default) No delay from the last character transmitted until RTS goes inactive • 1 to 65535 – Delay in milliseconds from the last character transmitted until RTS goes inactive (does not apply to RS485 ports). RTS delays shall be applied independent of the FLOW_CTRL selection.
RESP_TO [1]	IN	UInt	Response timeout: Time in milliseconds allowed by MB_MASTER for the slave to respond. If the slave does not respond in this time period, MB_MASTER will retry the request or terminate the request with an error when the specified number of retries has been sent. 5 ms to 65535 ms (default value = 1000 ms).
MB_DB	IN	Variant	A reference to the instance data block used by the MB_MASTER or MB_SLAVE instructions. After MB_SLAVE or MB_MASTER is placed in your program, the DB identifier appears in the parameter helper drop-list available at the MB_DB box connection.
DONE	OUT	Bool	The DONE bit is TRUE for one scan, after the last request was completed with no error. (Version 2.0 only)
ERROR	OUT	Bool	The ERROR bit is TRUE for one scan, after the last request was terminated with an error. The error code value at the STATUS parameter is valid only during the single scan where ERROR = TRUE.
STATUS	OUT	Word	Execution condition code

[1] Optional parameters for MB_COMM_LOAD (V 2.x or later). Click the arrow at the bottom of a LAD/FBD box to expand the box and include these parameters.

MB_COMM_LOAD is executed to configure a port for the Modbus RTU protocol. Once a port is configured for the Modbus RTU protocol, it can only be used by either the MB_MASTER or MB_SLAVE instructions.

One execution of MB_COMM_LOAD must be used to configure each communication port that is used for Modbus communication. Assign a unique MB_COMM_LOAD instance DB for each port that you use. You can install up to three communication modules (RS232 or RS485) and one communication board (RS485) in the CPU. Call MB_COMM_LOAD from a startup OB and execute it one time or use the first scan system flag (Page 103) to initiate the call to execute it one time. Only execute MB_COMM_LOAD again if communication parameters like baud rate or parity must change.

An instance data block is assigned for MB_MASTER or MB_SLAVE when you place these instructions in your program. This instance data block is referenced when you specify the MB_DB parameter for the MB_COMM_LOAD instruction.

MB_COMM_LOAD data block variables

The following table shows the public static variables stored in the instance DB for the MB_COMM_LOAD that can be used in your program.

Table 13- 142 Static variables in the instance DB

Variable	Data type	Description
ICHAR_GAP	UInt	Delay for Inter-character gap between characters. This parameter is specified in milliseconds and is used to increase the expected amount of time between received characters. The corresponding number of bit times for this parameter is added to the Modbus default of 35 bit times (3.5 character times).
RETRIES	UInt	Number of retries that the master will attempt before returning the no response error code 0x80C8.
STOP_BITS	USInt	Number of stop bits used in framing each character. Valid values are 1 and 2.

Table 13- 143 MB_COMM_LOAD execution condition codes [1]

STATUS (W#16#)	Description
0000	No error
8180	Invalid port ID value (wrong port/hardware identifier for communication module)
8181	Invalid baud rate value
8182	Invalid parity value
8183	Invalid flow control value
8184	Invalid response timeout value (response timeout less than the 5 ms minimum)
8185	MB_DB parameter is not an instance data block of a MB_MASTER or MB_SLAVE instruction.

[1] In addition to the MB_COMM_LOAD errors listed above, errors can be returned from the underlying PtP communication instructions.

13.9.3.2 MB_MASTER (Communicate using the PtP port as Modbus RTU master) instruction

Table 13- 144 MB_MASTER instruction

LAD / FBD	SCL	Description

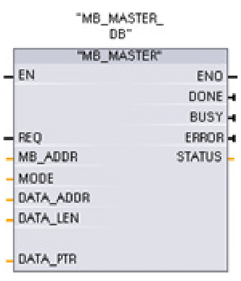
 "MB_MASTER_DB"
"MB_MASTER"
EN
ENO
DONE
BUSY
REQ
ERROR
MB_ADDR
STATUS
MODE
DATA_ADDR
DATA_LEN
DATA_PTR
	`"MB_MASTER_DB"(`
	`REQ:=_bool_in_,`
	`MB_ADDR:=_uint_in_,`
	`MODE:=_usint_in_,`
	`DATA_ADDR:=_udint_in_,`
	`DATA_LEN:=_uint_in_,`
	`DONE=>_bool_out_,`
	`BUSY=>_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_,`
	`DATA_PTR:=_variant_inout_);`
		The MB_MASTER instruction communicates as a Modbus master using a port that was configured by a previous execution of the MB_COMM_LOAD instruction. An instance data block is assigned automatically when you place the MB_MASTER instruction in your program. This MB_MASTER instance data block is used when you specify the MB_DB parameter for the MB_COMM_LOAD instruction.

Table 13- 145 Data types for the parameters

Parameter and type		Data type	Description
REQ	IN	Bool	0=No request 1= Request to transmit data to Modbus slave
MB_ADDR	IN	V1.0: USInt V2.0: UInt	Modbus RTU station address: Standard addressing range (1 to 247) Extended addressing range (1 to 65535) The value of 0 is reserved for broadcasting a message to all Modbus slaves. Modbus function codes 05, 06, 15 and 16 are the only function codes supported for broadcast.
MODE	IN	USInt	Mode Selection: Specifies the type of request (read, write, or diagnostic). See the Modbus functions table below for details.
DATA_ADDR	IN	UDInt	Starting Address in the slave: Specifies the starting address of the data to be accessed in the Modbus slave. See the Modbus functions table below for valid addresses.
DATA_LEN	IN	UInt	Data Length: Specifies the number of bits or words to be accessed in this request. See the Modbus functions table below for valid lengths.
DATA_PTR	IN	Variant	Data Pointer: Points to the M or DB address (Standard DB type) for the data being written or read.
DONE	OUT	Bool	The DONE bit is TRUE for one scan, after the last request was completed with no error.
BUSY	OUT	Bool	• 0 – No MB_MASTER operation in progress • 1 – MB_MASTER operation in progress
ERROR	OUT	Bool	The ERROR bit is TRUE for one scan, after the last request was terminated with an error. The error code value at the STATUS parameter is valid only during the single scan where ERROR = TRUE.
STATUS	OUT	Word	Execution condition code

Modbus master communication rules

- MB_COMM_LOAD must be executed to configure a port before a MB_MASTER instruction can communicate with that port.
- If a port is to be used to initiate Modbus master requests, that port should not be used by MB_SLAVE. One or more instances of MB_MASTER execution can be used with that port, but all MB_MASTER execution must use the same MB_MASTER instance DB for that port.
- The Modbus instructions do not use communication interrupt events to control the communication process. Your program must poll the MB_MASTER instruction for transmit and receive complete conditions.
- It is recommended that you call all MB_MASTER execution for a given port from a program cycle OB. Modbus master instructions may execute in only one of the program cycle or cyclic/time delay execution levels. They must not execute in both execution priority levels. Pre-emption of a Modbus Master instruction by another Modbus master instruction in a higher priority execution priority level will result in improper operation. Modbus master instructions must not execute in the startup, diagnostic or time error execution priority levels.
- Once a master instruction initiates a transmission, this instance must be continually executed with the EN input enabled until a DONE=1 state or ERROR=1 state is returned. A particular MB_MASTER instance is considered active until one of these two events occurs. While the original instance is active, any call to any other instance with the REQ input enabled will result in an error. If the continuous execution of the original instance stops, the request state remains active for a period of time specified by the static variable Blocked_Proc_Timeout. Once this period of time expires, the next master instruction called with an enabled REQ input will become the active instance. This prevents a single Modbus master instance from monopolizing or locking access to a port. If the original active instance is not enabled within the period of time specified by the static variable "Blocked_Proc_Timeout", then the next execution by this instance (with REQ not set) will clear the active state. If (REQ is set), then this execution initiates a new master request as if no other instance was active.

REQ parameter

0 = No request; 1 = Request to transmit data to Modbus Slave

You may control this input either through the use of a level or edge triggered contact. Whenever this input is enabled, a state machine is started to ensure that no other MB_MASTER using the same instance DB is allowed to issue a request, until the current request is completed. All other input states are captured and held internally for the current request, until the response is received or an error detected.

If the same instance of MB_MASTER is executed again with REQ input = 1 before the completion of the current request, then no subsequent transmissions are made. However, when the request is completed, a new request is issued whenever MB_MASTER is executed again with REQ input = 1.

DATA_ADDR and MODE parameters select the Modbus function type

DATA_ADDR (starting Modbus address in the slave): Specifies the starting address of the data to be accessed in the Modbus slave.

The MB_MASTER instruction uses a MODE input rather than a Function Code input. The combination of MODE and Modbus address determine the Function Code that is used in the actual Modbus message. The following table shows the correspondence between parameter MODE, Modbus function code, and Modbus address range.

Table 13- 146 Modbus functions

MODE	Modbus Function	Data length	Operation and data	Modbus Address
0	01	1 to 2000 1 to 1992 [1]	Read output bits: 1 to (1992 or 2000) bits per request	1 to 9999
0	02	1 to 2000 1 to 1992 [1]	Read input bits: 1 to (1992 or 2000) bits per request	10001 to 19999
0	03	1 to 125 1 to 124 [1]	Read Holding registers: 1 to (124 or 125) words per request	40001 to 49999 or 400001 to 465535
0	04	1 to 125 1 to 124 [1]	Read input words: 1 to (124 or 125) words per request	30001 to 39999
1	05	1	Write one output bit: One bit per request	1 to 9999
1	06	1	Write one holding register: 1 word per request	40001 to 49999 or 400001 to 465535
1	15	2 to 1968 2 to 1960 [1]	Write multiple output bits: 2 to (1960 or 1968) bits per request	1 to 9999
1	16	2 to 123 2 to 122 [1]	Write multiple holding registers: 2 to (122 or 123) words per request	40001 to 49999 or 400001 to 465535
2	15	1 to 1968 2 to 1960 [1]	Write one or more output bits: 1 to (1960 or 1968) bits per request	1 to 9999
2	16	1 to 123 1 to 122 [1]	Write one or more holding registers: 1 to (122 or 123) words per request	40001 to 49999 or 400001 to 465535
11	11	0	Read the slave communication status word and event counter. The status word indicates busy (0 – not busy, 0xFFFF - busy). The event counter is incremented for each successful completion of a message. Both the DATA_ADDR and DATA_LEN operands of MB_MASTER are ignored for this function.	
80	08	1	Check slave status using data diagnostic code 0x0000 (Loopback test – slave echoes the request) 1 word per request	

MODE	Modbus Function	Data length	Operation and data	Modbus Address
81	08	1	Reset slave event counter using data diagnostic code 0x000A 1 word per request	
3 to 10, 12 to 79, 82 to 255			Reserved	

[1] For "Extended Addressing" mode the maximum data lengths are reduced by 1 byte or 1 word depending upon the data type used by the function.

DATA_PTR parameter

The DATA_PTR parameter points to the DB or M address that is written to or read from. If you use a data block, then you must create a global data block that provides data storage for reads and writes to Modbus slaves.

Note

The DATA_PTR data block type must allow direct addressing

The data block must allow both direct (absolute) and symbolic addressing. When you create the data block the "Standard" access attribute must be selected.

Data block structures for the DATA_PTR parameter

- These data types are valid for **word reads** of Modbus addresses 30001 to 39999, 40001 to 49999, and 400001 to 465536 and also for **word writes** to Modbus addresses 40001 to 49999 and 400001 to 465536.
  - Standard array of WORD, UINT, or INT data types
  - Named WORD, UINT, or INT structure where each element has a unique name and 16 bit data type.
  - Named complex structure where each element has a unique name and a 16 or 32 bit data type.
- For **bit reads** and writes of Modbus addresses 00001 to 09999 and bit reads of 10001 to 19999.
  - Standard array of Boolean data types.
  - Named Boolean structure of uniquely named Boolean variables.
- Although not required, it is recommended that each MB_MASTER instruction have its own separate memory area. The reason for this recommendation is that there is a greater possibility of data corruption if multiple MB_MASTER instructions are reading and writing to the same memory area.
- There is no requirement that the DATA_PTR data areas be in the same global data block. You can create one data block with multiple areas for Modbus reads, one data block for Modbus writes, or one data block for each slave station.

Modbus master data block variables

The following table shows the public static variables stored in the instance DB for MB_MASTER that can be used in your program.

Table 13- 147 Static variables in the instance DB

Variable	Data type	Initial value	Description
Blocked_Proc_Timeout	Real	3.0	Amount of time (in seconds) to wait for a blocked Modbus Master instance before removing this instance as being ACTIVE. This can occur, for example, when a Master request has been issued and then the program stops calling the Master function before it has completely finished the request. The time value must be greater than 0 and less than 55 seconds, or an error occurs. The default value is .5 seconds.
Extended_Addressing	Bool	False	Configures single or double-byte slave addressing. The default value = 0. (0=single byte address, 1=double-byte address)

Your program can write values to the Blocked_Proc_Timeout and Extended_Addressing variables to control Modbus master operations. See the MB_SLAVE topic description of HR_Start_Offset and Extended_Addressing for an example of how to use these variables in the program editor and details about Modbus extended addressing (Page 1035).

Condition codes

Table 13- 148 MB_MASTER execution condition codes (communication and configuration errors) [1]

STATUS (W#16#)	Description
0000	No error
80C8	Slave timeout. Check baud rate, parity, and wiring of slave.
80D1	The receiver issued a flow control request to suspend an active transmission and never re-enabled the transmission during the specified wait time. This error is also generated during hardware flow control when the receiver does not assert CTS within the specified wait time.
80D2	The transmit request was aborted because no DSR signal is received from the DCE.
80E0	The message was terminated because the receive buffer is full.
80E1	The message was terminated as a result of a parity error.
80E2	The message was terminated as a result of a framing error.
80E3	The message was terminated as a result of an overrun error.
80E4	The message was terminated as a result of the specified length exceeding the total buffer size.
8180	Invalid port ID value or error with MB_COMM_LOAD instruction
8186	Invalid Modbus station address
8188	Invalid Mode specified for broadcast request
8189	Invalid Data Address value
818A	Invalid Data Length value
818B	Invalid pointer to the local data source/destination: Size not correct
818C	Invalid pointer for DATA_PTR or invalid Blocked_Proc_Timeout: The data area must be a DB (that allows both symbolic and direct access) or M memory.
8200	Port is busy processing a transmit request.

Table 13- 149 MB_MASTER execution condition codes (Modbus protocol errors) [1]

STATUS (W#16#)	Response code from slave	Modbus protocol errors
8380	-	CRC error
8381	01	Function code not supported
8382	03	Data length error
8383	02	Data address error or address outside the valid range of the DATA_PTR area
8384	Greater than 03	Data value error
8385	03	Data diagnostic code value not supported (function code 08)
8386	-	Function code in the response does not match the code in the request.
8387	-	Wrong slave responded
8388	-	The slave response to a write request is incorrect. The write request returned by the slave does not match what the master actually sent.

[1] In addition to the MB_MASTER errors listed above, errors can be returned from the underlying PtP communication instructions.

13.9.3.3 MB_SLAVE (Communicate using the PtP port as Modbus RTU slave) instruction

Table 13- 150 MB_SLAVE instruction

LAD / FBD	SCL	Description

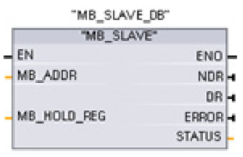
 "MB_SLAVE_DB"
"MB_SLAVE"
EN
ENO
MB_ADDR
NDR
DR
MB_HOLD_REG
ERROR
STATUS
	`"MB_SLAVE_DB"(`
	`    MB_ADDR:=_uint_in_,`
	`    NDR=>_bool_out_,`
	`    DR=>_bool_out_,`
	`    ERROR=>_bool_out_,`
	`    STATUS=>_word_out_,`
	`    MB_HOLD_REG:=_variant_inout_);`
		The MB_SLAVE instruction allows your program to communicate as a Modbus slave through a PtP port on the CM (RS485 or RS232) and CB (RS485). When a remote Modbus RTU master issues a request, your user program responds to the request by MB_SLAVE execution. STEP 7 automatically creates an instance DB when you insert the instruction. Use this MB_SLAVE_DB name when you specify the MB_DB parameter for the MB_COMM_LOAD instruction.

Table 13- 151 Data types for the parameters

Parameter and type		**Data type**	**Description**
MB_ADDR	IN	V1.0: USInt V2.0: UInt	The station address of the Modbus slave: Standard addressing range (1 to 247) Extended addressing range (0 to 65535)
MB_HOLD_REG	IN	Variant	Pointer to the Modbus Holding Register DB: The Modbus holding register can be M memory or a data block.
NDR	OUT	Bool	New Data Ready: • 0 – No new data • 1 – Indicates that new data has been written by the Modbus master
DR	OUT	Bool	Data Read: • 0 – No data read • 1 – Indicates that data has been read by the Modbus master
ERROR	OUT	Bool	The ERROR bit is TRUE for one scan, after the last request was terminated with an error. If execution is terminated with an error, then the error code value at the STATUS parameter is valid only during the single scan where ERROR = TRUE.
STATUS	OUT	Word	Execution error code

Modbus communication function codes (1, 2, 4, 5, and 15) can read and write bits and words directly in the input process image and output process image of the CPU. For these function codes, the MB_HOLD_REG parameter must be defined as a data type larger than a byte. The following table shows the example mapping of Modbus addresses to the process image in the CPU.

Table 13- 152 Mapping of Modbus addresses to the process image

Modbus functions						**S7-1200**	
Codes	Function	Data area	Address range			Data area	CPU address
01	Read bits	Output	1	to	8192	Output Process Image	Q0.0 to Q1023.7
02	Read bits	Input	10001	to	18192	Input Process Image	I0.0 to I1023.7
04	Read words	Input	30001	to	30512	Input Process Image	IW0 to IW1022
05	Write bit	Output	1	to	8192	Output Process Image	Q0.0 to Q1023.7
15	Write bits	Output	1	to	8192	Output Process Image	Q0.0 to Q1023.7

Modbus communication function codes (3, 6, 16) use a Modbus holding register which can be an M memory address range or a data block. The type of holding register is specified by the MB_HOLD_REG parameter on the MB_SLAVE instruction.

Note

MB_HOLD_REG data block type

A Modbus holding register data block must allow both direct (absolute) and symbolic addressing. When you create the data block the "Standard" access attribute must be selected.

The following table shows examples of Modbus address to holding register mapping that is used for Modbus function codes 03 (read words), 06 (write word), and 16 (write words). The actual upper limit of DB addresses is determined by the maximum work memory limit and M memory limit, for each CPU model.

Table 13- 153 Mapping of Modbus addresses to CPU memory

Modbus Master Address	**MB_HOLD_REG parameter examples**				
	MW100	**DB10.DBw0**	**MW120**	**DB10.DBW50**	**"Recipe".ingredient**
40001	MW100	DB10.DBW0	MW120	DB10.DBW50	"Recipe".ingredient[1]
40002	MW102	DB10.DBW2	MW122	DB10.DBW52	"Recipe".ingredient[2]
40003	MW104	DB10.DBW4	MW124	DB10.DBW54	"Recipe".ingredient[3]
40004	MW106	DB10.DBW6	MW126	DB10.DBW56	"Recipe".ingredient[4]
40005	MW108	DB10.DBW8	MW128	DB10.DBW58	"Recipe".ingredient[5]

Table 13- 154 Diagnostic functions

S7-1200 MB_SLAVE Modbus diagnostic functions		
Codes	Sub-function	Description
08	0000H	Return query data echo test: The MB_SLAVE will echo back to a Modbus master a word of data that is received.
08	000AH	Clear communication event counter: The MB_SLAVE will clear out the communication event counter that is used for Modbus function 11.
11		Get communication event counter: The MB_SLAVE uses an internal communication event counter for recording the number of successful Modbus read and write requests that are sent to the Modbus slave. The counter does not increment on any Function 8, Function 11, or broadcast requests. It is also not incremented on any requests that result in a communication error (for example, parity or CRC errors).

The MB_SLAVE instruction supports broadcast write requests from any Modbus master as long as the request is for accessing valid addresses. MB_SLAVE will produce error code 0x8188 for function codes not supported in broadcast.

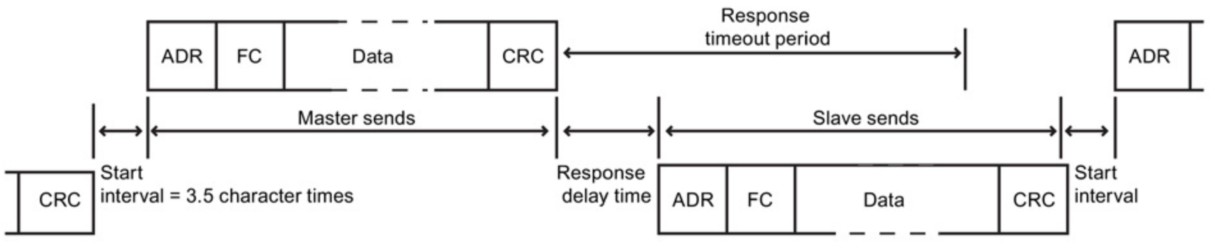
Modbus slave communication rules

- MB_COMM_LOAD must be executed to configure a port, before a MB_SLAVE instruction can communicate through that port.
- If a port is to respond as a slave to a Modbus master, then do not program that port with the MB_MASTER instruction.
- Only one instance of MB_SLAVE can be used with a given port, otherwise erratic behavior may occur.
- The Modbus instructions do not use communication interrupt events to control the communication process. Your program must control the communication process by polling the MB_SLAVE instruction for transmit and receive complete conditions.

- The MB_SLAVE instruction must execute periodically at a rate that allows it to make a timely response to incoming requests from a Modbus master. It is recommended that you execute MB_SLAVE every scan from a program cycle OB. Executing MB_SLAVE from a cyclic interrupt OB is possible, but is not recommended because of the potential for excessive time delays in the interrupt routine to temporarily block the execution of other interrupt routines.

Modbus signal timing

MB_SLAVE must be executed periodically to receive each request from the Modbus master and then respond as required. The frequency of execution for MB_SLAVE is dependent upon the response timeout period of the Modbus master. This is illustrated in the following diagram.



The response timeout period RESP_TO is the amount of time a Modbus master waits for the start of a response from a Modbus slave. This time period is not defined by the Modbus protocol, but is a parameter of each Modbus master. The frequency of execution (the time between one execution and the next execution) of MB_SLAVE must be based on the particular parameters of your Modbus master. At a minimum, you should execute MB_SLAVE twice within the response timeout period of the Modbus master.

Modbus slave variables

This table shows the public static variables stored in the MB_SLAVE instance data block that can be used in your program

Table 13- 155 Modbus slave variables

Variable	Data type	Description
Request_Count	Word	The number of all requests received by this slave
Slave_Message_Count	Word	The number of requests received for this specific slave
Bad_CRC_Count	Word	The number of requests received that have a CRC error
Broadcast_Count	Word	The number of broadcast requests received
Exception_Count	Word	Modbus specific errors that require a returned exception
Success_Count	Word	The number of requests received for this specific slave that have no protocol errors
HR_Start_Offset	Word	Specifies the starting address of the Modbus Holding register (default = 0)
Extended_Addressing	Bool	Configures single or double-byte slave addressing (0=single byte address, 1=double-byte address, default = 0)

Your program can write values to the HR_Start_Offset and Extended_Addressing variables and control Modbus slave operations. The other variables can be read to monitor Modbus status.

HR_Start_Offset

Modbus holding register addresses begin at 40001 or 400001. These addresses correspond to the beginning PLC memory address of the holding register. However, you can configure the "HR_Start_Offset" variable to start the beginning Modbus holding register address at another value instead of 40001 or 400001.

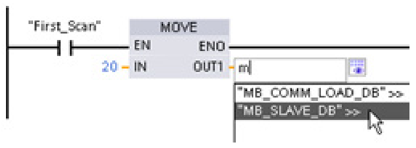
For example, if the holding register is configured to start at MW100 and is 100 words long. An offset of 20 specifies a beginning holding register address of 40021 instead of 40001. Any address below 40021 and above 400119 will result in an addressing error.

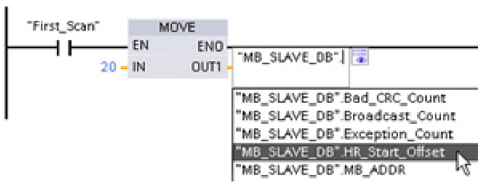
Table 13- 156 Example of Modbus holding register addressing

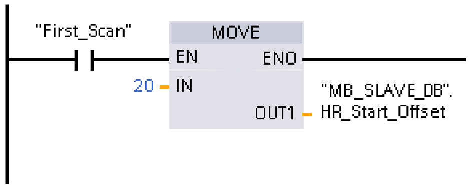
HR_Start_Offset	Address	Minimum	Maximum
0	Modbus address (Word)	40001	40099
	S7-1200 address	MW100	MW298
20	Modbus address (Word)	40021	40119
	S7-1200 address	MW100	MW298

HR_Start_Offset is a word value that specifies the starting address of the Modbus holding register and is stored in the MB_SLAVE instance data block. You can set this public static variable value by using the parameter helper drop-list, after MB_SLAVE is placed in your program.

For example, after MB_SLAVE is placed in a LAD network, you can go to a previous network and assign the HR_Start_Offset value. The value must be assigned prior to execution of MB_SLAVE.




"First_Scan"
MOVE
EN
ENO
20
IN
OUT1
"MB_SLAVE_DB".
"MB_SLAVE_DB".Bad_CRC_Count
"MB_SLAVE_DB".Broadcast_Count
"MB_SLAVE_DB".Exception_Count
"MB_SLAVE_DB".HR_Start_Offset
"MB_SLAVE_DB".MB_ADDR



Entering a Modbus slave variable using the default DB name:

1. Set the cursor in the parameter field and type an m character.
2. Select "MB_SLAVE_DB" from the drop-list.
3. Set the cursor at the right side of the DB name (after the quote character) and enter a period character.
4. Select "MB_SLAVE_DB.HR_Start_Offset" from the drop list.

Extended_Addressing

The Extended_Addressing variable is accessed in a similar way as the HR_Start_Offset reference discussed above except that the Extended_Addressing variable is a Boolean value. The Boolean value must be written by an output coil and not a move box.

Modbus slave addressing can be configured to be either a single byte (which is the Modbus standard) or double byte. Extended addressing is used to address more than 247 devices within a single network. Selecting extended addressing allows you to address a maximum of 64000 addresses. A Modbus function 1 frame is shown below as an example.

Table 13- 157 Single-byte slave address (byte 0)

Function 1	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	
Request	Slave addr.	F code	Start address		Length of coils		
Valid Response	Slave addr.	F code	Length	Coil data			
Error response	Slave addr.	0x81	E code				

Table 13- 158 Double-byte slave address (byte 0 and byte 1)

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Request	Slave address		F code	Start address		Length of coils	
Valid Response	Slave address		F code	Length	Coil data		
Error response	Slave address		0x81	E code			

Condition codes

Table 13- 159 MB_SLAVE execution condition codes (communication and configuration errors) 1

STATUS (W#16#)	Description
80D1	The receiver issued a flow control request to suspend an active transmission and never re-enabled the transmission during the specified wait time. This error is also generated during hardware flow control when the receiver does not assert CTS within the specified wait time.
80D2	The transmit request was aborted because no DSR signal is received from the DCE.
80E0	The message was terminated because the receive buffer is full.
80E1	The message was terminated as a result of a parity error.
80E2	The message was terminated as a result of a framing error.
80E3	The message was terminated as a result of an overrun error.
80E4	The message was terminated as a result of the specified length exceeding the total buffer size.
8180	Invalid port ID value or error with MB_COMM_LOAD instruction
8186	Invalid Modbus station address
8187	Invalid pointer to MB_HOLD_REG DB: Area is too small
818C	Invalid MB_HOLD_REG pointer to M memory or DB (DB area must allow both symbolic and direct address)

Table 13- 160 MB_SLAVE execution condition codes (Modbus protocol errors) [1]

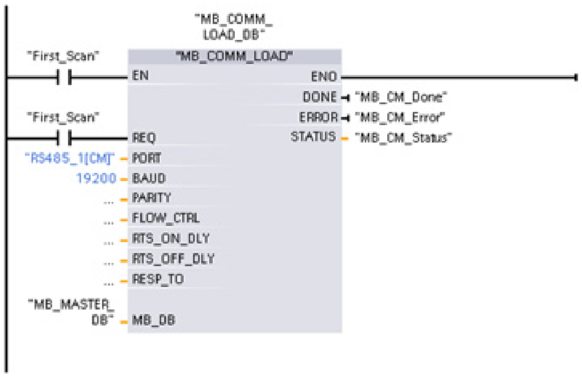
STATUS (W#16#)	Response code from slave	Modbus protocol errors
8380	No response	CRC error
8381	01	Function code not supported or not supported within broadcasts
8382	03	Data length error
8383	02	Data address error or address outside the valid range of the DATA_PTR area
8384	03	Data value error
8385	03	Data diagnostic code value not supported (function code 08)

[1] In addition to the MB_SLAVE errors listed above, errors can be returned from the underlying PtP communication instructions.

13.9.4 Example: Legacy Modbus RTU master program

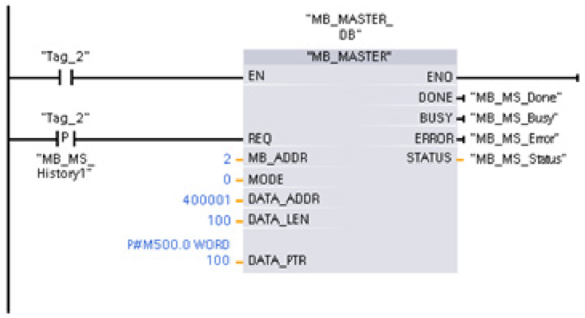
MB_COMM_LOAD is initialized during start-up by using the first scan flag. Execution of MB_COMM_LOAD in this manner should only be done when the serial port configuration will not change at runtime.

Network 1: Initialize the RS485 module parameters only once during the first scan.

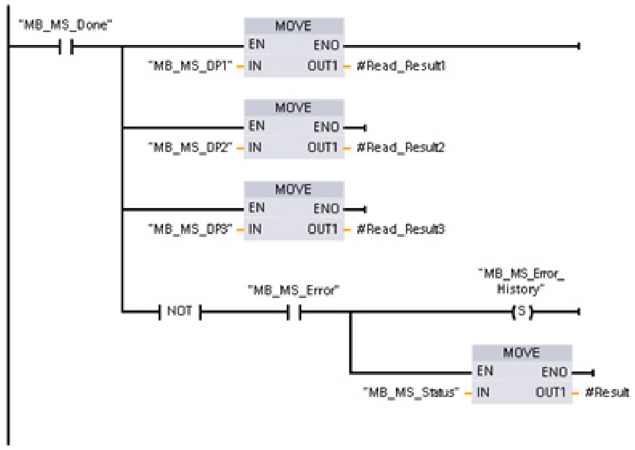


One MB_MASTER instruction is used in the program cycle OB to communicate with a single slave. Additional MB_MASTER instructions can be used in the program cycle OB to communicate with other slaves, or one MB_MASTER FB could be re-used to communicate with additional slaves.

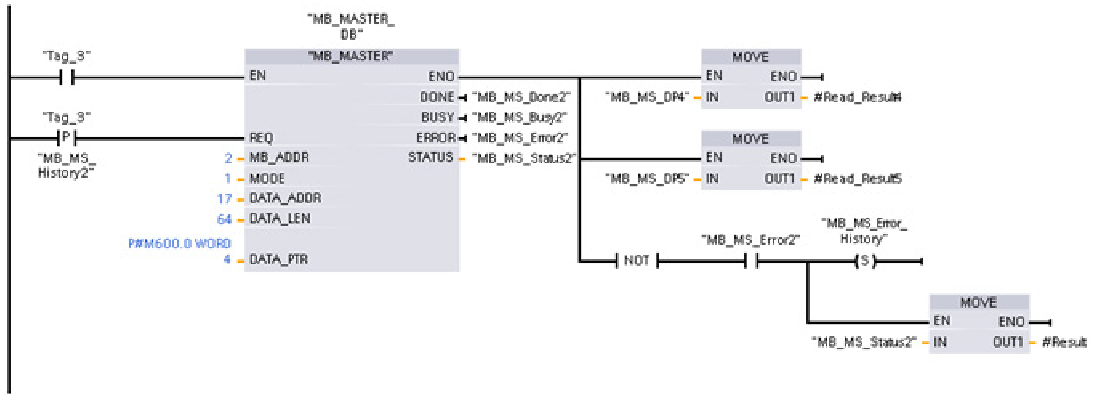
Network 2: Read 100 words from the slave holding register.



Network 3: This is an optional network that just shows the values of the first 3 words once the read operation is done.



Network 4: Write 64 bits to the output image register starting at slave address Q2.0.

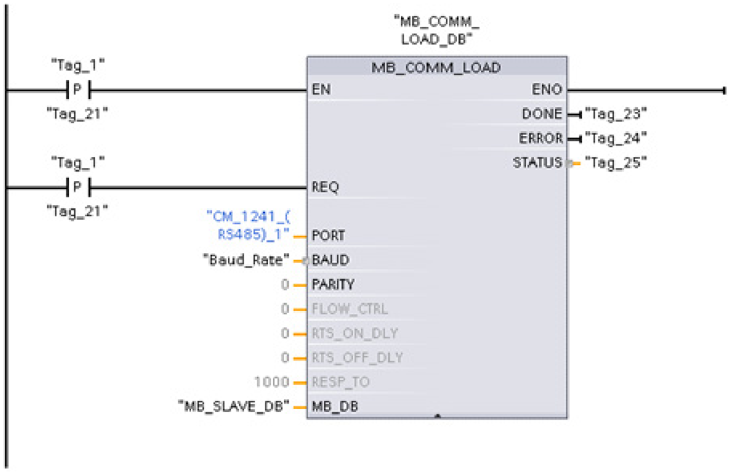


13.9.5 Example: Legacy Modbus RTU slave program

MB_COMM_LOAD shown below is initialized each time "Tag_1" is enabled.

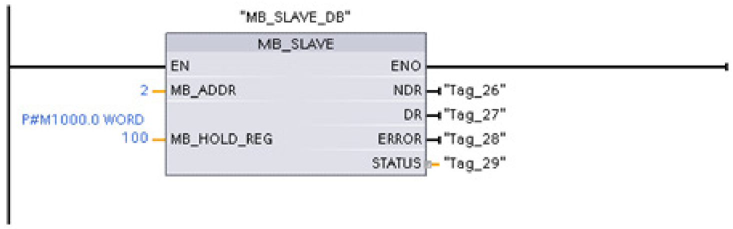
Execution of MB_COMM_LOAD in this manner should only be done when the serial port configuration will change at runtime, as a result of HMI configuration.

Network 1: Initialize the RS485 module parameters each time they are changed by an HMI device.



MB_SLAVE shown below is placed in a cyclic OB that is executed every 10ms. While this does not give the absolute fastest response by the slave, it does provide good performance at 9600 baud for short messages (20 bytes or less in the request).

Network 2: Check for Modbus master requests during each scan. The Modbus holding register is configured for 100 words starting at MW1000.



13.10 Telecontrol and TeleService with the CP 1242-7

13.10.1 Telecontrol CPs Overview

TeleControl CPs for the S7-1200

For TeleControl applications, the following communications processors are available:

- **CP 1243-1**

  Article number 6GK7 243-1BX30-0XE0

  Communications processor for connecting the SIMATIC S7-1200 via the public infrastructure (e.g. DSL) to a control center with TeleControl Server Basic (TCSB version V3).

  With the help of VPN technology and the firewall, the CP allows protected access to the S7-1200.

  The CP can be used as an additional Ethernet interface of the CPU for S7 communication.

  The communication between CP and CPU is via configurable data points that access PLC tags.

- **CP 1243-1 DNP3**

  Article number 6GK7 243-1JX30-0XE0

  Communications processor for connecting the SIMATIC S7-1200 to control centers using the DNP3 protocol.

  The communication between CP and CPU is via configurable data points that access PLC tags.

- **CP 1243-1 IEC**

  Article number 6GK7 243-1PX30-0XE0

  Communications processor for connecting SIMATIC S7-1200 to control centers using the IEC 60870-5 protocol.

  The communication between CP and CPU is via configurable data points that access PLC tags.

- **CP 1242-7**

  Article number 6GK7 242-7KX30-0XE0

  Communications processor for connecting the SIMATIC S7-1200 to a control center with TeleControl Server Basic using mobile wireless (GPRS) and the public infrastructure (DSL)

- **CP 1242-7 GPRS V2**

  Article number 6GK7 242-7KX31-0XE0

  Communications processor for connecting the SIMATIC S7-1200 to a control center with TeleControl Server Basic (TCSB version v3) using mobile wireless (GPRS) and the public infrastructure (DSL)

  With the help of VPN technology and the firewall, the CP allows protected access to the S7-1200.

  The CP can be used as an additional Ethernet interface of the CPU for S7 communication.

  The communication between CP and CPU is via configurable data points that access PLC tags.

- **CP 1243-7 LTE-xx**

  Communications processor for connecting the SIMATIC S7-1200 to a control center with TeleControl Server Basic (TCSB version v3) using mobile wireless (GPRS) and the public infrastructure (DSL)

  Support of the following mobile wireless specifications: GSM/GPRS, UMTS (G3), LTE

  To cover countries with different mobile wireless specifications, the CP is available in two variants:

  – CP 1243-7 LTE-US

    North American standard

    Article number 6GK7 243-7SX30-0XE0

  – CP 1243-7 LTE-EU

    Western European standard

    Article number 6GK7 243-7KX30-0XE0

  With the help of VPN technology and the firewall, the CP allows protected access to the S7-1200.

  The CP can be used as an additional Ethernet interface of the CPU for S7 communication.

  The communication between CP and CPU is via configurable data points that access PLC tags.

13.10.2 Connection to a GSM network

IP-based WAN communication via GPRS

Using the CP 1242-7 communications processor, the S7-1200 can be connected to GSM networks. The CP 1242-7 allows WAN communication from remote stations with a control center and inter-station communication.

Inter-station communication is possible only via a GSM network. For communication between a remote station and a control room, the control center must have a PC with Internet access.

The CP 1242-7 supports the following services for communication via the GSM network:

- GPRS (General Packet Radio Service)

  The packet-oriented service for data transmission "GPRS" is handled via the GSM network.
- SMS (Short Message Service)

  The CP 1242-7 can receive and send SMS messages. The communications partner can be a mobile phone or an S7-1200.

The CP 1242-7 is suitable for use in industry worldwide and supports the following frequency bands:

- 850 MHz
- 900 MHz
- 1,800 MHz
- 1,900 MHz

Requirements

The equipment used in the stations or the control center depends on the particular application.

- For communication with or via a central control room, the control center requires a PC with Internet access.
- Apart from the station equipment, a remote S7-1200 station with a CP 1242-7 must meet the following requirements to be able to communicate via the GSM network:
  - A contract with a suitable GSM network provider

    If GPRS is used, the contract must allow the use of the GPRS service.

    If there is to be direct communication between stations only via the GSM network, the GSM network provider must assign a fixed IP address to the CPs. In this case, communication between stations is not via the control center.
  - The SIM card belonging to the contract

    The SIM card is inserted in the CP 1242-7.
  - Local availability of a GSM network in the range of the station

13.10.3 Applications of the CP 1242-7

The CP 1242-7 can be used for the following applications:

Telecontrol applications

- Sending messages by SMS

  Via the CP 1242-7, the CPU of a remote S7-1200 station can receive SMS messages from the GSM network or send messages by SMS to a configured mobile phone or an S7-1200.

- Communication with a control center

  Remote S7-1200 stations communicate via the GSM network and the Internet with a telecontrol server in the master station. For data transfer using GPRS, the "TELECONTROL SERVER BASIC" application is installed on the telecontrol server in the master station. The telecontrol server communicates with a higher-level central control system using the integrated OPC server function.

- Communication between S7-1200 stations via a GSM network

  Communication between remote stations with a CP 1242-7 can be handled in two different ways:

  – Inter-station communication via a master station

    In this configuration, a permanent secure connection between S7-1200 stations that communicate with each other and the telecontrol server is established in the master station. Communication between the stations is via the telecontrol server. The CP 1242-7 operates in "Telecontrol" mode.

  – Direct communication between the stations

    For direct communication between stations without the detour via the master station, SIM cards with a fixed IP address are used that allow the stations to address each other directly. The possible communications services and security functions (for example VPN) depend on what is offered by the network provider. The CP 1242-7 operates in "GPRS direct" mode.

TeleService via GPRS

A TeleService connection can be established between an engineering station with STEP 7 and a remote S7-1200 station with a CP 1242-7 via the GSM network and the Internet. The connection runs from the engineering station via a telecontrol server or a TeleService gateway that acts as an intermediary forwarding frames and establishing the authorization. These PCs use the functions of the "TELECONTROL SERVER BASIC" application.

You can use the TeleService connection for the following purposes:

- Downloading configuration or program data from the STEP 7 project to the station
- Querying diagnostics data on the station

13.10.4 Other properties of the CP-1242-7

Other services and functions of the CP 1242-7

- Time-of-day synchronization of the CP via the Internet

  You can set the time on the CP as follows:

  – In "Telecontrol" mode, the time of day is transferred by the telecontrol server. The CP uses this to set its time.

  – In "GPRS direct" mode, the CP can request the time using SNTP.

  To synchronize the CPU time, you can read out the current time from the CP using a block.
- Interim buffering of messages to be sent if there are connection problems
- Increased availability thanks to the option of connecting to a substitute telecontrol server
- Optimized data volume (temporary connection)

  As an alternative to a permanent connection to the telecontrol server, the CP can be configured in STEP 7 with a temporary connection to the telecontrol server. In this case, a connection to the telecontrol server is established only when required.
- Logging the volume of data

  The volumes of data transferred are logged and can be evaluated for specific purposes.

13.10.5 Configuration and electrical connections

Configuration and module replacement

To configure the module, the following configuration tool is required:

STEP 7 version V11.0 SP1 or higher

For STEP 7 V11.0 SP1, you also require support package "CP 1242-7" (HSP0003001).

For process data transfer using GPRS, use the telecontrol communications instructions in the user program of the station.

The configuration data of the CP 1242-7 is stored on the local CPU. This allows simple replacement of the CP when necessary.

You can insert up to three modules of the CP 1242-7 type per S7-1200. This, for example, allows redundant communications paths to be established.

Electrical connections

- Power supply of the CP 1242-7

  The CP has a separate connection for the external 24 VDC power supply.
- Wireless interface for the GSM network

  An extra antenna is required for GSM communication. This is connected via the SMA socket of the CP.

13.10.6 Further information

Further information

The CP 1242-7 manual contains detailed information. You will find this on the Internet on the pages of Siemens Industrial Automation Customer Support under the following entry ID:

45605894 (http://support.automation.siemens.com/WW/view/en/45605894)

13.10.7 Accessories

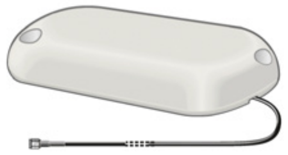
The ANT794-4MR GSM/GPRS antenna

The following antennas are available for use in GSM/GPRS networks and can be installed both indoors and outdoors:

- Quadband antenna ANT794-4MR

Short name	Order no.	Explanation
ANT794-4MR	6NH9 860-1AA00	Quadband antenna (900, 1800/1900 MHz, UMTS); weatherproof for indoor and outdoor areas; 5 m connecting cable connected permanently to the antenna; SMA connector, including installation bracket, screws, wall plugs

- Flat antenna ANT794-3M



Short name	Order no.	Explanation
ANT794-3M	6NH9 870-1AA00	Flat antenna (900, 1800/1900 MHz); weatherproof for indoor and outdoor areas; 1.2 m connecting cable connected permanently to the antenna; SMA con-nector, including adhesive pad, screws mounting possible

The antennas must be ordered separately.

13.10.8 Reference to GSM antenna manual

Further information

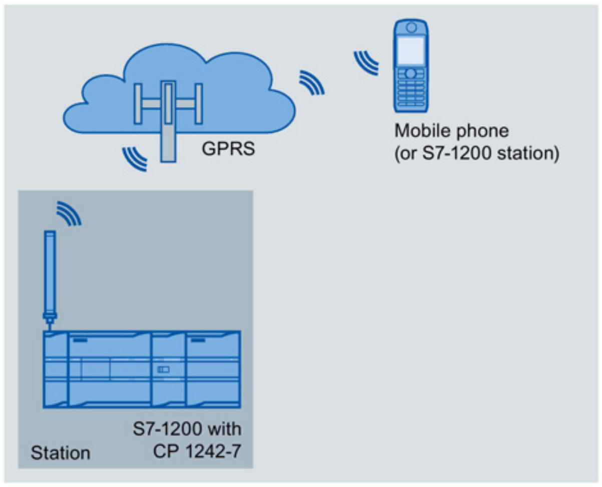
You will find detailed information in the device manual. You will find this on the Internet on the pages of Siemens Industrial Automation Customer Support under the following entry ID:

23119005 (http://support.automation.siemens.com/WW/view/en/23119005)

13.10.9 Configuration examples for telecontrol

Below, you will find several configuration examples for stations with a CP 1242-7.

Sending messages by SMS



A SIMATIC S7-1200 with a CP 1242-7 can send messages by SMS to a mobile phone or a configured S7-1200 station.

Telecontrol by a control center

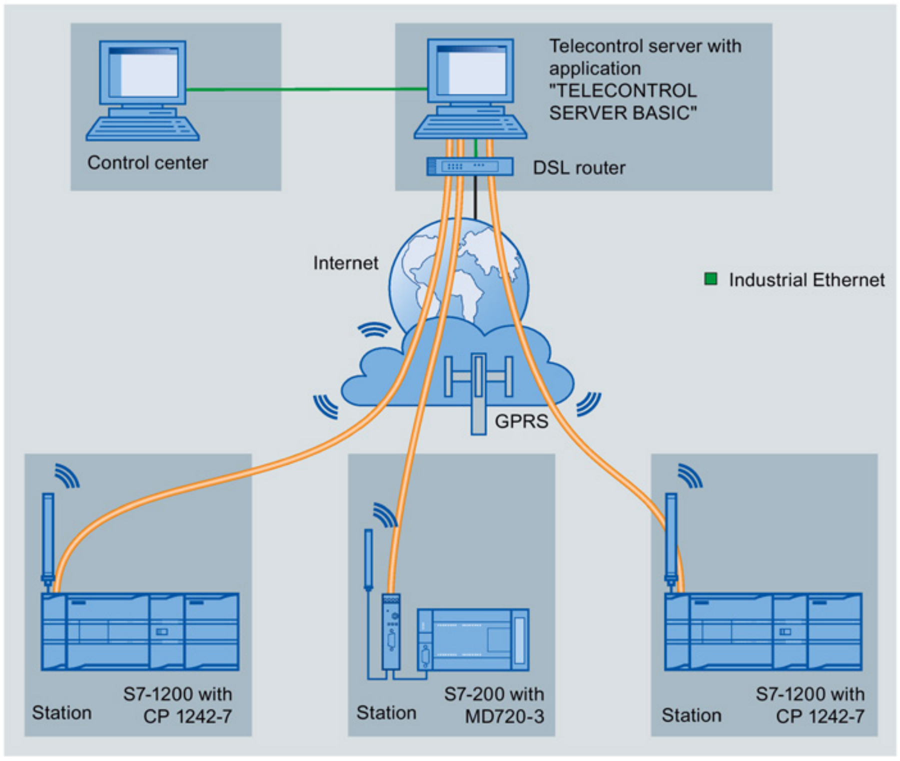


Figure 13-1 Communication between S7-1200 stations and a control center

In telecontrol applications, SIMATIC S7-1200 stations with a CP 1242-7 communicate with a control center via the GSM network and the Internet. The "TELECONTROL SERVER BASIC" (TCSB) application is installed on the telecontrol server in the master station. This results in the following use cases:

- Telecontrol communication between station and control center

  In this use case, data from the field is sent by the stations to the telecontrol server in the master station via the GSM network and Internet. The telecontrol server is used to monitor remote stations.

- Communication between a station and a control room with OPC client

  As in the first case, the stations communicate with the telecontrol server. Using its integrated OPC server, the telecontrol server exchanges data with the OPC client of the control room.

  The OPC client and telecontrol server can be located on a single computer, for example when TCSB is installed on a control center computer with WinCC.

- Inter-station communication via a control center

  Inter-station communication is possible with S7 stations equipped with a CP 1242-7.

  To allow inter-station communication, the telecontrol server forwards the messages of the sending station to the receiving station.

Direct communication between stations

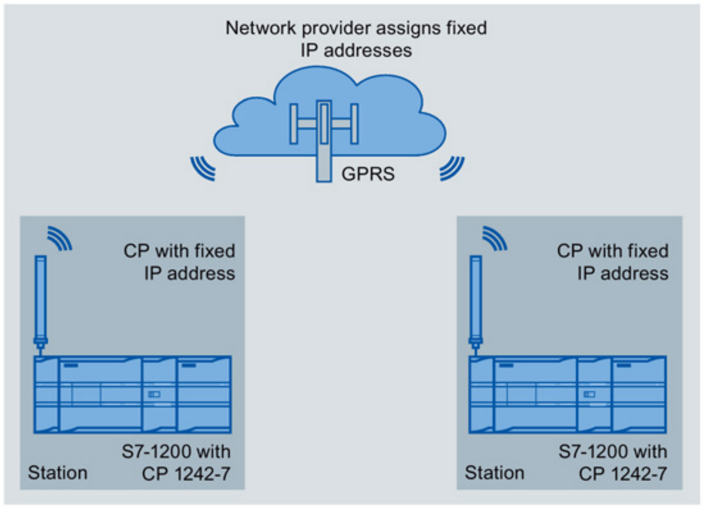


Figure 13-2 Direct communication between two S7-1200 stations

In this configuration, two SIMATIC S7-1200 stations communicate directly with each other using the CP 1242-7 via the GSM network. Each CP 1242-7 has a fixed IP address. The relevant service of the GSM network provider must allow this.

TeleService via GPRS

In TeleService via GPRS, an engineering station on which STEP 7 is installed communicates via the GSM network and the Internet with the CP 1242-7 in the S7-1200.

Since a firewall is normally closed for connection requests from the outside, a switching station between the remote station and the engineering station is required. This switching station can be a telecontrol server or, if there is no telecontrol server in the configuration, a TeleService gateway.

TeleService with telecontrol server

The connection runs via the telecontrol server.

- The engineering station and telecontrol server are connected via the Intranet (LAN) or Internet.
- The telecontrol server and remote station are connected via the Internet and via the GSM network.

The engineering station and telecontrol server can also be the same computer; in other words, STEP 7 and TCSB are installed on the same computer.

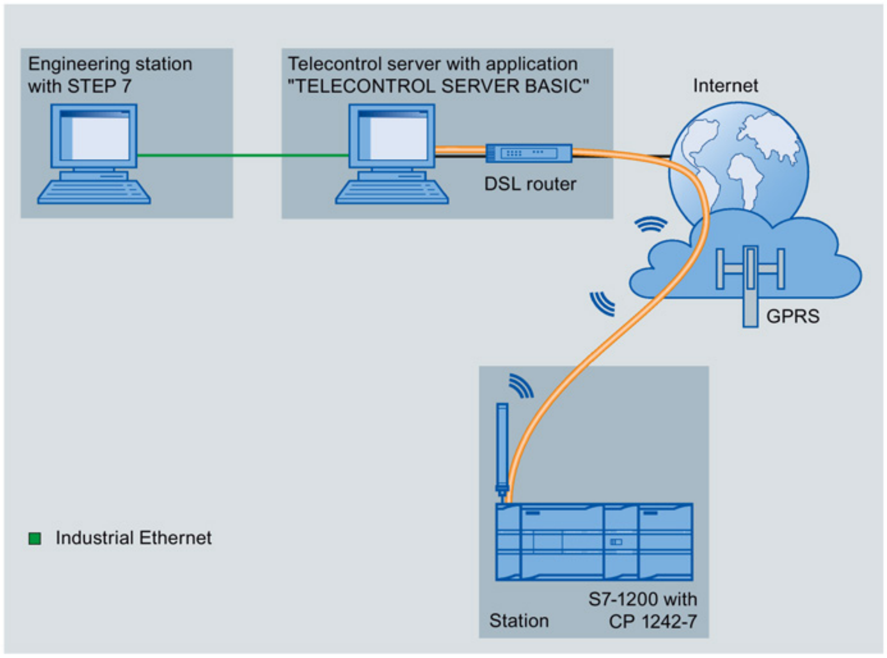

Engineering station with STEP 7
Telecontrol server with application "TELECONTROL SERVER BASIC"
Internet
DSL router
GPRS
Industrial Ethernet
S7-1200 with CP 1242-7
Station

Figure 13-3 TeleService via GPRS in a configuration with telecontrol server

TeleService without a telecontrol server

The connection runs via the TeleService gateway.

The connection between the engineering station and the TeleService gateway can be local via a LAN or via the Internet.

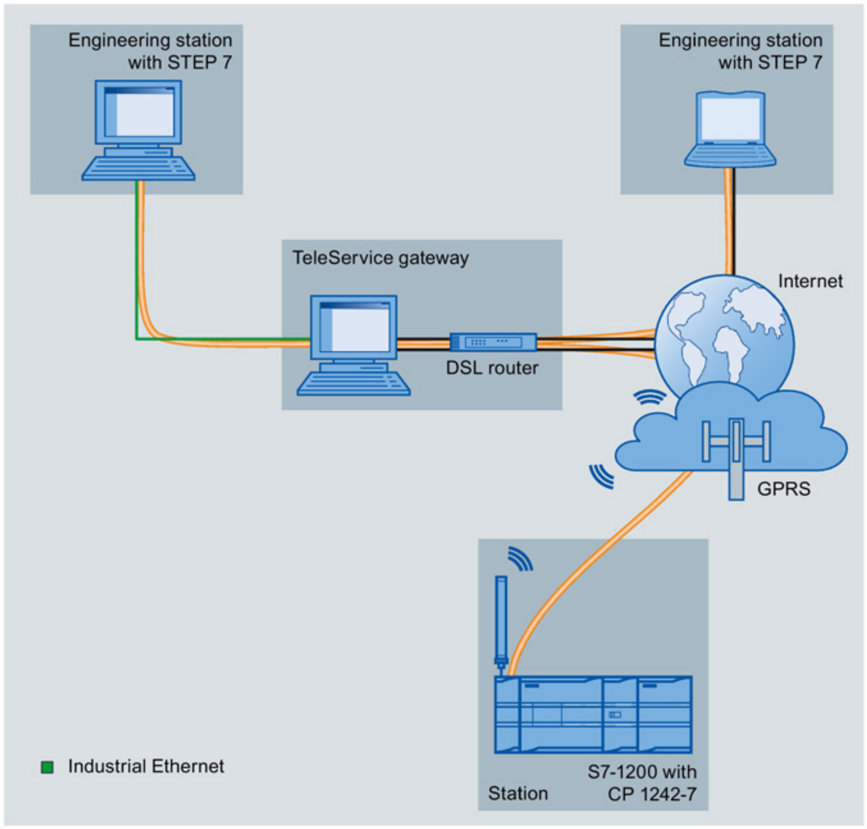


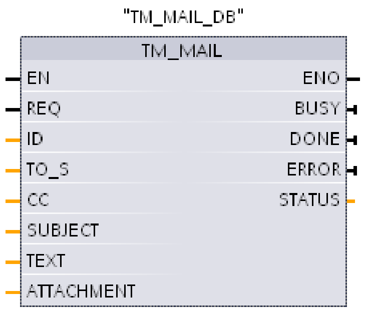
Figure 13-4 TeleService via GPRS in a configuration with TeleService gateway

TeleService communication (SMTP email) 14

14.1 TM_Mail (Send email) instruction

Table 14- 1 TM_MAIL instruction

LAD / FBD	SCL	Description

 "TM_MAIL_DB"
TM_MAIL
EN
REQ
ID
TO_S
CC
SUBJECT
TEXT
ATTACHMENT
ENO
BUSY
DONE
ERROR
STATUS
	`"TM_MAIL_DB"(`
	`REQ:=_bool_in_,`
	`ID:=_int_in_,`
	`TO_S:=_string_in_,`
	`CC:=_string_in_,`
	`SUBJECT:=_string_in_,`
	`TEXT:= _string_in_,`
	`ATTACHMENT:=_variant_in_,`
	`BUSY=>_bool_out_,`
	`DONE=>_bool_out_,`
	`ERROR=>_bool_out_,`
	`STATUS=>_word_out_,);`
		The TM_MAIL instruction sends an email message using the SMTP (Simple Mail Transfer Protocol) over TCP/IP via the CPU Industrial Ethernet connection. Where Ethernet-based Internet connectivity is not available, an optional Teleservice adapter can be used for connection with telephone land lines. TM_MAIL executes asynchronously and the job extends over multiple TM_MAIL calls. When you call TM_MAIL, you must assign an instance DB. **The instance DB retentive attribute must not be set**. This ensures that the instance DB is initialized in the transition of the CPU from STOP to RUN and that a new TM_MAIL operation can be triggered.

1 STEP 7 automatically creates the instance DB when you insert the instruction.

You start sending an email with a positive edge change from 0 to 1, at input parameter REQ. The following table shows the relationship between BUSY, DONE and ERROR. You can monitor the progress of TM_MAIL execution and detect completion, by evaluating these parameters in successive calls.

The output parameters DONE, ERROR, STATUS, and SFC_STATUS are valid for only one cycle, when the state of the output parameter BUSY changes from 1 to 0. Your program logic must save temporary output state values, so you can detect state changes in subsequent program execution cycles.

Table 14- 2 Interaction of the Done, Busy and Error parameters

DONE	BUSY	ERROR	Description
Irrelevant	1	Irrelevant	Job is in progress.
1	0	0	The job was completed successfully.
0	0	1	The job was terminated with an error. For the cause of the error, refer to the STATUS parameter.
0	0	0	No job in progress

If the CPU is changed to STOP mode while TM_MAIL is active, then the communication connection to the email server is terminated. The communication connection to the email server is also lost if problems occur in CPU communication on the Industrial Ethernet bus. In these cases, the send process is suspended and the email does not reach the recipient.

NOTICE
Modifying user programs Deletion and replacement of program blocks, the calls to TM_MAIL, or calls to the instance DBs of TM_MAIL can break the linking of program blocks. If you fail to maintain linked program blocks, then the TPC/IP communication functions can enter an undefined state, possibly resulting in property damage. After transferring a modified program block, you would have to perform a CPU restart (warm) or cold start. To avoid breaking the linking of program blocks, only change the parts of your user program that directly affect the TM_MAIL calls in the following cases: • The CPU in the STOP mode • No email is sent (REQ and BUSY = 0)

Data consistency

The input parameter ADDR_MAIL_SERVER is read when the operation is started. A new value does not take effect until the current operation is complete and a new TM_MAIL operation is initiated.

In contrast, the parameters WATCH_DOG_TIME, TO_S, CC, FROM, SUBJECT, TEXT, ATTACHMENT, USERNAME and PASSWORD are read during the execution of TM_MAIL and may be changed only when the job is finished (BUSY = 0)

Dial-up connection: Configuring the TS adapter IE parameters

You must configure the Teleservice adapter IE parameters for outgoing calls to connect with the dial-up server of your Internet Service Provider. If you set the call "on demand" attribute, then the connection is established only when an e-mail will be sent. For an analog modem connection, more time is required for the connection process (approx. a minute longer). You must include the extra time, in the WATCH_DOG_TIME value.

Table 14- 3 Data types for the parameters

Parameter and type		Data types	Description
REQ	IN	Bool	A low to high (positive edge) signal starts the operation.
ID	IN	Int	Connection identifier: See the ID parameter of the instructions TCON, TDISCON, TSEND and TRCV. A number that is not used for any additional instances of this instruction in the user program must be used.
TO_S	IN	String	Recipient addresses: STRING data with a maximum length of 240 characters
CC	IN	String	CC copy to recipient addresses (optional): STRING data with a maximum length of 240 characters

Parameter and type		Data types	Description
SUBJECT	IN	String	Subject name of the email: STRING data with a maximum length 240 characters.
TEXT	IN	String	Text message of the email (optional): STRING data with a maximum length of 240 characters. If this parameter is an empty string, then the email will be sent without message text.
ATTACHMENT	IN	Variant	Pointer to email attachment data: Byte, word, or double word data with a maximum length of 65534 bytes. If no value is assigned, then the email sent without an attachment.
DONE	OUT	Bool	• 0 - Job not yet started or still executing. • 1 - Job was executed error-free.
BUSY	OUT	Bool	• 0 - No operation in progress • 1- Operation in progress
ERROR	OUT	Bool	The ERROR bit =1 for one scan, after the last request was terminated with an error. The error code value at the STATUS output is valid only during the single scan where ERROR = 1.
STATUS	OUT	Word	Return value or error information of the TM_MAIL instruction.
ADDR_MAIL_SERVER	1 Static	DWord	IP address of the mail server: You must assign each IP address fragment as an octet of two 4-bit hexadecimal characters. If the IP address fragment = decimal value 10 which equals hexadecimal value A, then you must enter "0A" for that octet. For example: IP address = 192.168.0.10 ADDR_MAIL_SERVER = DW#16#C0A8000A, where: • 192 = 16#C0, • 168 =16#A8 • 0 = 16#00 • 10 = 16#0A
WATCH_DOG_TIME	1 Static	Time	The maximum time allowed for TM_MAIL to complete the entire SMTP process, from the initiation of the connection to the SMTP to the end of the SMTP transmission. If this time is exceeded, then TM_MAIL execution ends with an error. The actual time delay until TM_MAIL ends and the error is issued may exceed the WATCH_DOG_TIME, because of the additional time required for the disconnect operation. At first you should set a time of 2 minutes. This time can be much smaller for an ISDN phone connection.
USERNAME	1 Static	String	Mail account user name: STRING data with a maximum length 180 characters.
PASSWORD	1 Static	String	Mail server password: STRING data with a maximum length 180 characters.

Parameter and type		Data types	Description
FROM	[1] Static	String	Sender address: STRING with a maximum length of 240 characters
SFC_STATUS	[1] Static	Word	Execution condition code of the called communication blocks

[1] The values of these parameters are not modified at every call of TM_MAIL. The values are assigned in the TM_MAIL instance data block and are only referenced once, on the first call of TM_MAIL.

SMTP authentication

TM_MAIL supports the SMTP AUTH LOGIN authentication method. For information on this authentication method, please refer to the manual of the mail server or the website of your internet service provider.

The AUTH LOGIN authentication method uses the TM_MAIL USERNAME and PASSWORD parameters to connect with the mail server. The user name and password must be previously set up on an email account at an email server.

If no value is assigned for the USERNAME parameter, then the AUTH LOGIN authentication method is not used and the email is sent without authentication.

TO_S:, CC:, and FROM: parameters

The parameters TO_S:, CC: and FROM: are strings, as shown in the following examples:

TO: <wenna@mydomain.com>, <ruby@mydomain.com>,

CC: <admin@mydomain.com>, <judy@mydomain.com>,

FROM: <admin@mydomain.com>

The following rules must be used when entering these character strings:

- The characters "TO:", "CC:" and "FROM:" must be entered, including the colon character.
- A space character and an opening angle bracket "<" must precede each address. For example, there must be a space character between "TO:" and <email address>.
- A closing angle bracket ">" must be entered after each address.
- A comma character "," must be entered after each email address for the TO_S: and CC: addresses. For example, the comma after the single email address is required in "TO: <email address>,".
- Only one email address may be used for the FROM: entry, with no comma at the end.

Because of run-time mode and memory usage, a syntax check is not performed on the TM_MAIL TO_S:, CC: and FROM: data. If the format rules above are not followed exactly. The SMTP email server transaction will fail.

STATUS and SFC_STATUS parameters

The execution condition codes returned by TM_MAIL can be classified as follows:

- W#16#0000: Operation of TM_MAIL was completed successfully
- W#16#7xxx: Status of TM_MAIL operation
- W#16#8xxx: An error in an internal call to a communication device or the mail server

The following table shows the execution condition codes of TM_MAIL with the exception of the error codes from internally called communication modules.

Note

Email server requirements

TM_MAIL can only communicate with an email server using SMTP via port 25. The assigned port number cannot be changed.

Most IT departments and external email servers now block port 25 to prevent a PC infected with a virus from becoming a rogue email generator.

You can connect to an internal mail server via SMTP and let the internal server manage the current security enhancements that are required to relay email through the Internet to an external mail server.

Example: Internal email server configuration

If you use Microsoft Exchange as an internal mail server, then you can configure the server to allow SMTP access from the IP address assigned the S7-1200 PLC. Configure the Exchange management console: Server configuration > Hub transport > Receive connectors > IP relay. On the Network tab, there is a box named "Receive mail from remote servers that have these IP addresses". This is where you put the IP address of the PLC that is executing the TM_MAIL instruction. No authentication is required for this type of connection with an internal Microsoft Exchange server.

Email server configuration

TM_MAIL can only use an email server that allows port 25 communication, SMTP, and AUTH LOGIN authentication (optional).

Configure a compatible email server account to accept remote SMTP log in. Then edit the instance DB for TM_MAIL to put in the TM_MAIL USERNAME and PASSWORD character strings that are used to authenticate the connection with your email account.

Table 14- 4 Condition codes

STATUS (W#16#...):	**SFC_STATUS (W#16#...):**	**Description**
0000	-	The TM_MAIL operation completed without error. This zero STATUS code does not guarantee that an email was actually sent (See the first item in the note following this table).
7001	-	TM_MAIL is active (BUSY = 1).
7002	7002	TM_MAIL is active (BUSY = 1).
8xxx	xxxx	The TM_MAIL operation was completed with an error in the internal communication instruction calls. For more information about the SFC_STATUS parameter, see the descriptions of the STATUS parameter of the underlying PROFINET open user communication instructions.
8010	xxxx	Failed to connect: For more information about the SFC_STATUS parameter, see the STATUS parameter of the TCON instruction.
8011	xxxx	Error sending data: For more information about SFC_STATUS parameter, see the STATUS parameter of the TSEND instruction.
8012	xxxx	Error while receiving data: For more information about the SFC_STATUS parameter, see the STATUS parameter descriptions of the TRCV instruction.
8013	xxxx	Failed to connect: For more information for evaluating the SFC_STATUS parameter, see the STATUS parameter descriptions of the TCON and TDISCON instructions.
8014	-	Failed to connect: You may have entered an incorrect mail server IP address (ADDR_MAIL_SERVER) or too little time (WATCH_DOG_TIME) for the connection. It is also possible that the CPU has no connection to the network or the CPU configuration is incorrect.
8015	-	Invalid pointer for ATTACHMENT parameter: Use a variant pointer with a data type and length assignment. For example, "P#DB.DBX0.0" is incorrect and "P#DB.DBX0.0 byte 256" is correct.
82xx, 84xx, 85xx	-	The error message comes from the mail server and corresponds to error number "8" of the SMTP protocol. See the second item in the note following this table.
8450	-	Operation does not run: Mailbox is not available; try again later.
8451	-	Operation aborted: Local error in processing, .try again later
8500	-	Command syntax error: The cause may be that the email server does not support the LOGIN authentication process. Check the parameters of TM_MAIL. Try to send an email without authentication. Try replacing the parameter USERNAME with an empty string.
8501	-	Syntax error: Incorrect parameter or argument; you may have typed an incorrect address in the TO_S or CC parameters.
8502	-	Command is unknown or not implemented: Check your entries, especially the parameter FROM. Perhaps this is incomplete and you have omitted the "@" or "." characters.
8535	-	SMTP authentication is incomplete. You may have entered an incorrect username or password.
8550	-	The mail server cannot be reached, or you have no access rights. You may have entered an incorrect username or password or your mail server does not support log in access. Another cause of this error could be an erroneous entry of the domain name after the "@" character in the TO_S or CC parameters.
8552	-	Operation aborted: Exceeded the allocated memory size; try again later.
8554	-	Transmission failed: Try again later.

Note

Possible unreported email transmission errors

- Incorrect entry of a recipient address does not generate a STATUS error for TM_MAIL. In this case, there is no guarantee that additional recipients (with correct email addresses), will receive the email.
- More information on SMTP error codes can be found on the internet or in the error documentation for the mail server. You can also read the last error message from the mail server. The error message is stored in buffer1parameter of the instance DB for TM_MAIL.

15 Online and diagnostic tools

15.1 Status LEDs

The CPU and the I/O modules use LEDs to provide information about either the operational status of the module or the I/O.

Status LEDs on a CPU

The CPU provides the following status indicators:

- STOP/RUN
  - Solid yellow indicates STOP mode
  - Solid green indicates RUN mode
  - Flashing (alternating green and yellow) indicates that the CPU is in STARTUP mode
- ERROR
  - Flashing red indicates an error, such as an internal error in the CPU, a error with the memory card, or a configuration error (mismatched modules)
  - Defective state:

    - Solid red indicates defective hardware

    - All LEDs flash if the defect is detected in the firmware
- MAINT (Maintenance) flashes whenever you insert a memory card. The CPU then changes to STOP mode. After the CPU has changed to STOP mode, perform one of the following functions to initiate the evaluation of the memory card:
  - Change the CPU to RUN mode
  - Perform a memory reset (MRES)
  - Power-cycle the CPU

You can also use the LED instruction (Page 385) to determine the status of the LEDs.

Table 15- 1 Status LEDs for a CPU

Description	**STOP/RUN** **Yellow / Green**	**ERROR** **Red**	**MAINT** **Yellow**
Power is off	Off	Off	Off
Startup, self-test, or firmware update	Flashing (alternating yellow and green)	-	Off
Stop mode	On (yellow)	-	-
Run mode	On (green)	-	-
Remove the memory card	On (yellow)	-	Flashing
Error	On (either yellow or green)	Flashing	-

Description	STOP/RUN Yellow / Green	ERROR Red	MAINT Yellow
Maintenance requested • Forced I/O • Battery replacement re-quired (if battery board in-stalled)	On (either yellow or green)	-	On
Defective hardware	On (yellow)	On	Off
LED test or defective CPU firmware	Flashing (alternating yellow and green)	Flashing	Flashing
Unknown or incompatible version of CPU configuration	On (yellow)	Flashing	Flashing

Note

"Unknown or incompatible version of CPU configuration" error

Attempting to download an S7-1200 V3.0 program to an S7-1200 V4.0 CPU causes a CPU error, and the CPU displays a corresponding error message in the diagnostic buffer. If you reached this state by using an invalid version program transfer card (Page 135), then remove the card, perform a STOP to RUN transition, a memory reset (MRES) or cycle power. If you reach this state by an invalid program download, reset the CPU to factory settings (Page 1073). After you recover the CPU from the error condition, you can download a valid V4.0 CPU program.

The CPU also provides two LEDs that indicate the status of the PROFINET communications. Open the bottom terminal block cover to view the PROFINET LEDs.

- Link (green) turns on to indicate a successful connection
- Rx/Tx (yellow) turns on to indicate transmission activity

The CPU and each digital signal module (SM) provide an I/O Channel LED for each of the digital inputs and outputs. The I/O Channel (green) turns on or off to indicate the state of the individual input or output.

S7-1200 behavior following a fatal error

If the CPU firmware detects a fatal error it attempts a defect-mode restart, and if successful, signals the defective mode by continually flashing the STOP/RUN, ERROR and MAINT LEDs. The user program and hardware configuration are not loaded following the defect-mode restart.

If the CPU successfully completes the defect-mode restart, the CPU and signal board outputs are set to 0, and the outputs of central rack signal modules and distributed I/O are set to the configured "Reaction to CPU STOP".

If the defect-mode restart fails, (for example, due to a hardware fault), the STOP and ERROR LEDs are ON and the MAINT LED is OFF.

WARNING

Operation in defect state cannot be guaranteed

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

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

Status LEDs on an SM

In addition, each digital SM 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 SM provides an I/O Channel LED for each of the analog inputs and outputs.

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

In addition, each analog SM 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 SM detects the presence or absence of power to the module (field-side power, if required).

Table 15- 2 Status LEDs for a signal module (SM)

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

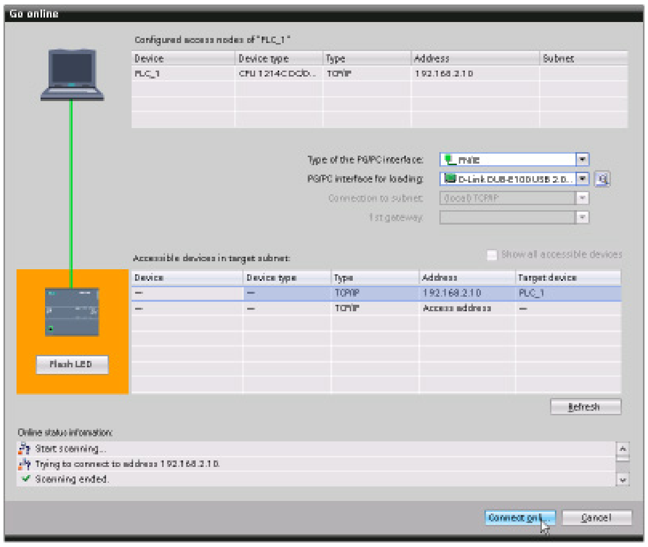
15.2 Going online and connecting to a CPU

You must establish an online connection between the programming device and CPU for loading programs and project engineering data as well as for activities such as the following:

- Testing user programs
- Displaying and changing the operating mode of the CPU (Page 1075)
- Displaying and setting the date and time of day of the CPU (Page 1072)
- Displaying the module information
- Comparing and synchronizing (Page 1077) offline to online program blocks
- Uploading and downloading program blocks
- Displaying diagnostics and the diagnostics buffer (Page 1076)
- Using a watch table (Page 1081) to test the user program by monitoring and modifying values
- Using a force table to force values in the CPU (Page 1084)

To establish an online connection to a configured CPU, click the CPU from the Project Navigation tree and click the "Go online" button from the Project View:

Go online



Configured access nodes of "PLC_1"

Device	Device type	Type	Address	Subnet
PLC_1	CPU 1214C DC/D...	TCP/IP	192.168.2.10	

Type of the PG/PC interface: PN/IE
PG/PC interface for loading: D-Link DUB-E100 USB 2.0...
Connection to subnet: (local) TCP/IP
1st gateway:

Accessible devices in target subnet:
Show all accessible devices

Device	Device type	Type	Address	Target device
—	—	TCP/IP	192.168.2.10	PLC_1
—	—	TCP/IP	Access address	—

Online status information:
Start scanning...
Trying to connect to address 192.168.2.10.
Scanning ended.

If this is the first time to go online with this CPU, you must select the type of PG/PC interface and the specific PG/PC interface from the Go Online dialog before establishing an online connection to a CPU found on that interface.

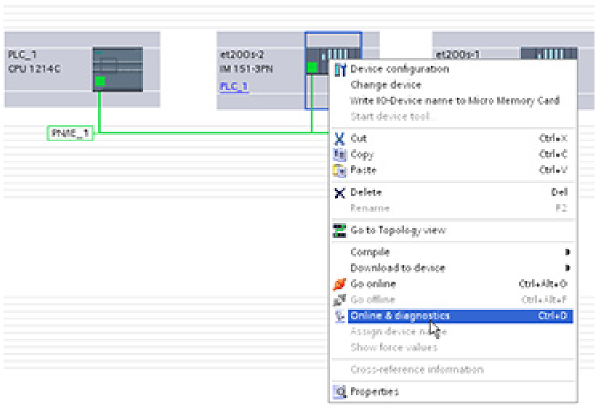
You have now connected your programming device to the CPU. The orange color frames indicate an online connection. You can now use the Online & diagnostics tools from the Project tree and the Online tools task card.

15.3 Assigning a name to a PROFINET IO device online

The devices on your PROFINET network must have an assigned name before you can connect with the CPU. Use the "Devices & networks" editor to assign names to your PROFINET devices if the devices have not already been assigned a name or if the name of the device is to be changed.

For each PROFINET IO device, you must assign the same name to that device in both the STEP 7 project and, using the "Online & diagnostics" tool, to the PROFINET IO device configuration memory (for example, an ET200 S interface module configuration memory). If a name is missing or does not match in either location, the PROFINET IO data exchange mode will not run.

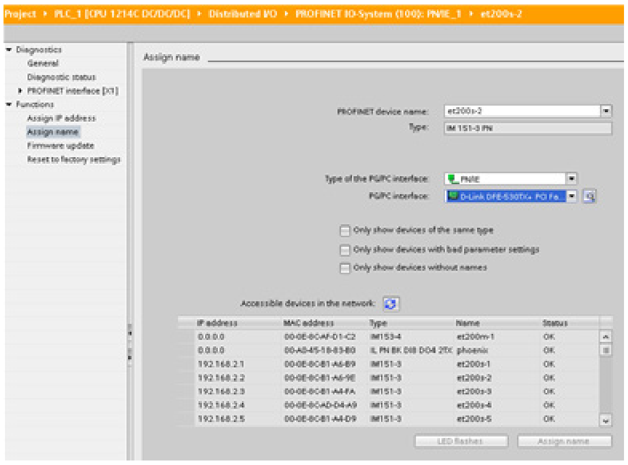
1. In the "Devices & networks" editor, right-click on the required PROFINET IO device, and select "Online & diagnostics".



2. In the "Online & diagnostics" dialog, make the following menu selections:

- "Functions"
- "Assign name"

Click the "Accessible devices in the network" icon to display all of the PROFINET IO devices on the network.



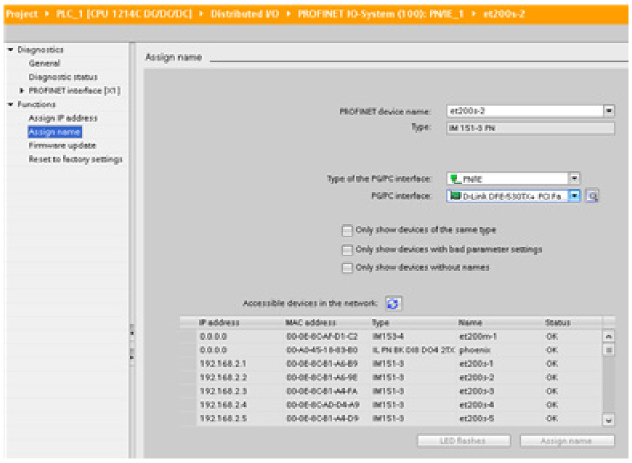
PROFINET device name: et200s-2
Type: IM 151-3 PN
Type of the PG/PC interface: PN/IE
PG/PC interface: D-Link DFE-530TX+ PCI Fa...

Only show devices of the same type
Only show devices with bad parameter settings
Only show devices without names

Accessible devices in the network:

IP address	MAC address	Type	Name	Status
0.0.0.0	00-0E-8C-AF-D1-C2	IM153-4	et200m-1	OK
0.0.0.0	00-A0-45-18-83-B0	IL PN BK DI8 DO4 2TX	phoenix	OK
192.168.2.1	00-0E-8C-B1-A6-B9	IM151-3	et200s-1	OK
192.168.2.2	00-0E-8C-B1-A6-9E	IM151-3	et200s-2	OK
192.168.2.3	00-0E-8C-B1-A4-FA	IM151-3	et200s-3	OK
192.168.2.4	00-0E-8C-AD-D4-A9	IM151-3	et200s-4	OK
192.168.2.5	00-0E-8C-B1-A4-D9	IM151-3	et200s-5	OK

3. In the list that is displayed, click the required PROFINET IO device, and click the "Assign name" button to write the name to the PROFINET IO device configuration memory.



PROFINET device name: et200s-2
Type: IM 151-3 PN

Type of the PG/PC interface: PN/IE
PG/PC interface: D-Link DFE-530TX+ PCI Fa

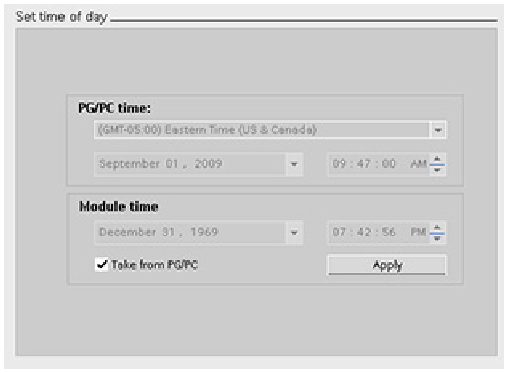
☐ Only show devices of the same type
☐ Only show devices with bad parameter settings
☐ Only show devices without names

Accessible devices in the network:

IP address	MAC address	Type	Name	Status
0.0.0.0	00-0E-8C-AF-D1-C2	IM153-4	et200m-1	OK
0.0.0.0	00-A0-45-18-83-B0	IL PN BK DI8 DO4 2TX	phoenix	OK
192.168.2.1	00-0E-8C-81-A6-B9	IM151-3	et200s-1	OK
192.168.2.2	00-0E-8C-81-A6-9E	IM151-3	et200s-2	OK
192.168.2.3	00-0E-8C-81-A4-FA	IM151-3	et200s-3	OK
192.168.2.4	00-0E-8C-AD-D4-A9	IM151-3	et200s-4	OK
192.168.2.5	00-0E-8C-81-A4-D9	IM151-3	et200s-5	OK

15.4 Setting the IP address and time of day

You can set the IP address (Page 625) and time of day in the online CPU. After accessing "Online & diagnostics" from the Project tree for an online CPU, you can display or change the IP address. You can also display or set the time and date parameters of the online CPU.



Set time of day

PG/PC time:

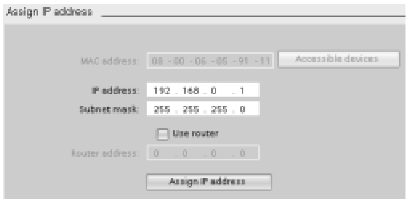
(GMT-05:00) Eastern Time (US & Canada)

September 01 , 2009 — 09 : 47 : 00 AM

Module time

December 31 , 1969 — 07 : 42 : 56 PM

[✓] Take from PG/PC



Assign IP address

MAC address: 08 - 00 - 06 - 05 - 91 - 11

IP address: 192 . 168 . 0 . 1

Subnet mask: 255 . 255 . 255 . 0

[ ] Use router

Router address: 0 . 0 . 0 . 0

Note

This feature is available only for a CPU that either has only a MAC address (has not yet been assigned an IP address) or has been reset to factory settings.

15.5 Resetting to factory settings

You can reset an S7-1200 to its original factory settings under the following conditions:

- The CPU has an online connection.
- The CPU is in STOP mode.

Note

If the CPU is in RUN mode and you start the reset operation, you can place it in STOP mode after acknowledging a confirmation prompt.

Procedure

To reset a CPU to its factory settings, follow these steps:

1. Open the Online and Diagnostics view of the CPU.
2. Select "Reset to factory settings" from the "Functions" folder.
3. Select the "Retain IP address" check box if you want to retain the IP address or the "Delete IP address" check box if you want to delete the IP address.
4. Click the "Reset" button.
5. Acknowledge the confirmation prompt with "OK".

Result

The module switches to STOP mode if necessary, and it resets the factory settings. The CPU perfoms the following actions:

With memory card installed in CPU	Without memory card installed in CPU
• Clears the diagnostics buffer • Resets the time of day • Restores work memory from the memory card • Sets all operand areas to configured initial values • Sets all parameters to their configured values • Retains or deletes the IP address based on the selection you made. (The MAC address is fixed and is never changed.)[1] • Deletes the control data record (Page 151), if present	• Clears the diagnostics buffer • Resets the time of day • Clears the work memory and internal load memory • Sets all operand areas to configured initial values • Sets all parameters to their configured values • Retains or deletes the IP address based on the selection you made. (The MAC address is fixed and is never changed.)[1] • Deletes the control data record, if present

[1] If you selected "Retain IP address", the CPU sets the IP address, subnet mask, and router address (if used) to the settings in your hardware configuration, unless you have modified these values from the user program or another tool, in which case the CPU restores the modified values.

15.6 Updating firmware

You can update the firmware of the connected CPU from the STEP 7 online and diagnostics tools.

To perform a firmware update, follow these steps:

1. Open the Online and Diagnostics view of the connected CPU.
2. Select "Firmware update" from the "Functions" folder.
3. Click the Browse button and navigate to the location that contains the firmware update file. This could be a location on your hard drive to which you have downloaded an S7-1200 (http://support.automation.siemens.com/WW/view/en/34612486/133100) firmware update file from the service and support Web site (http://www.siemens.com/automation/).
4. Select a file that is compatible with your module. For a selected file, the table displays the compatible modules.
5. Click the "Run update" button. Follow the dialogs, if necessary, to change the operating mode of your CPU.

STEP 7 displays progress dialogs as it loads the firmware update. When it finishes, it prompts you to start the module with the new firmware.

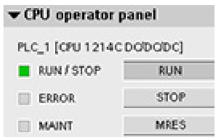
Note

If you do not choose to start the module with the new firmware, the previous firmware remains active until you reset the module, for example by cycling power. The new firmware becomes active only after you reset the module.

You can also perform a firmware update by one of the following additional methods:

- Using a memory card (Page 141)
- Using the Web server "Module Information" standard Web page (Page 804)

15.7 CPU operator panel for the online CPU



The "CPU operator panel" displays the operating mode (STOP or RUN) of the online CPU. The panel also shows whether the CPU has an error or if values are being forced.

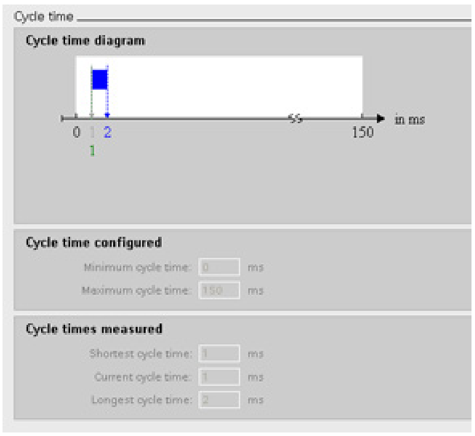
Use the CPU operating panel of the Online Tools task card to change the operating mode of an online CPU. The Online Tools task card is accessible whenever the CPU is online.

15.8 Monitoring the cycle time and memory usage

You can monitor the cycle time and memory usage of an online CPU.

After connecting to the online CPU, open the Online tools task card to view the following measurements:

- Cycle time
- Memory usage



Cycle time configured

Minimum cycle time: 0 ms

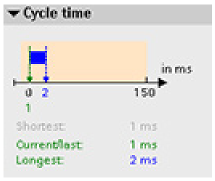
Maximum cycle time: 150 ms

Cycle times measured

Shortest cycle time: 1 ms

Current cycle time: 1 ms

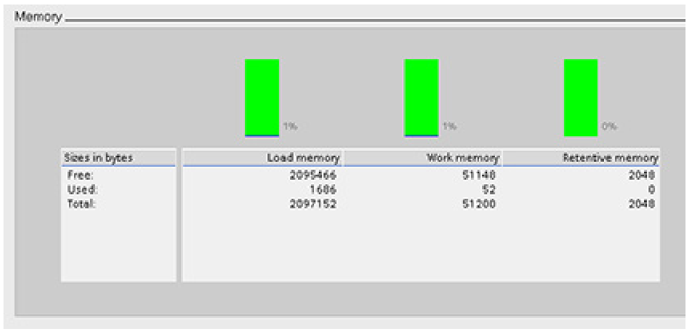
Longest cycle time: 2 ms



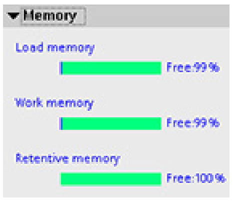
Shortest: 1 ms

Current/last: 1 ms

Longest: 2 ms



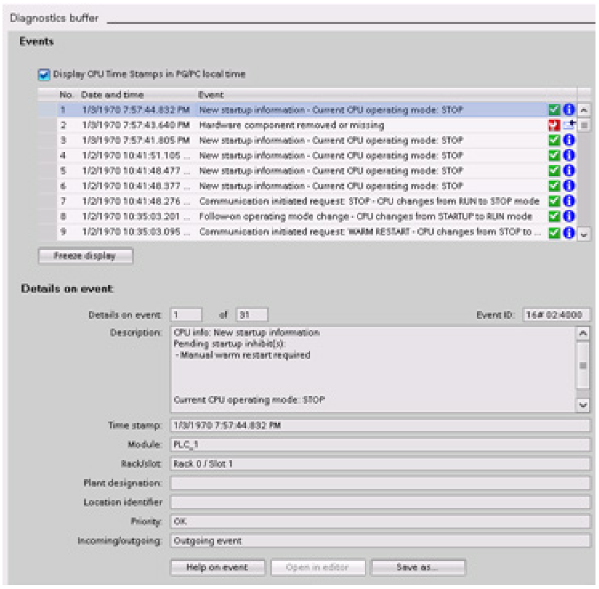
Sizes in bytes	Load memory	Work memory	Retentive memory
Free:	2095466	51148	2048
Used:	1686	52	0
Total:	2097152	51200	2048



15.9 Displaying diagnostic events in the CPU

Use the diagnostics buffer to review the recent activity in the CPU. The diagnostics buffer is accessible from "Online & Diagnostics" for an online CPU in the Project tree. It contains the following entries:

- Diagnostic events
- Changes in the CPU operating mode (transitions to STOP or RUN mode)



Diagnostics buffer

Events

✓ Display CPU Time Stamps in PG/PC local time

No.	Date and time	Event
1	1/3/1970 7:57:44.832 PM	New startup information - Current CPU operating mode: STOP
2	1/3/1970 7:57:43.640 PM	Hardware component removed or missing
3	1/3/1970 7:57:41.805 PM	New startup information - Current CPU operating mode: STOP
4	1/2/1970 10:41:51.105 ...	New startup information - Current CPU operating mode: STOP
5	1/2/1970 10:41:48.477 ...	New startup information - Current CPU operating mode: STOP
6	1/2/1970 10:41:48.377 ...	New startup information - Current CPU operating mode: STOP
7	1/2/1970 10:41:48.276 ...	Communication initiated request: STOP - CPU changes from RUN to STOP mode
8	1/2/1970 10:35:03.201 ...	Follow-on operating mode change - CPU changes from STARTUP to RUN mode
9	1/2/1970 10:35:03.095 ...	Communication initiated request: WARM RESTART - CPU changes from STOP to ...

Details on event:

Details on event:	1 of 31
Event ID:	16# 02:4000
Description:	CPU info: New startup information Pending startup inhibit(s): - Manual warm restart required Current CPU operating mode: STOP
Time stamp:	1/3/1970 7:57:44.832 PM
Module:	PLC_1
Rack/slot:	Rack 0 / Slot 1
Plant designation:	
Location identifier	
Priority:	OK
Incoming/outgoing:	Outgoing event

The first entry contains the latest event. Each entry in the diagnostic buffer contains the date and time the event was logged, and a description.

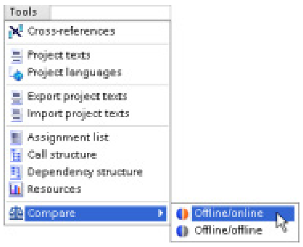
The maximum number of entries is dependent on the CPU. A maximum of 50 entries is supported.

Only the 10 most recent events in the diagnostic buffer are stored permanently. Resetting the CPU to the factory settings resets the diagnostic buffer by deleting the entries.

You can also use the GET_DIAG instruction (Page 398) to collect the diagnostic information.

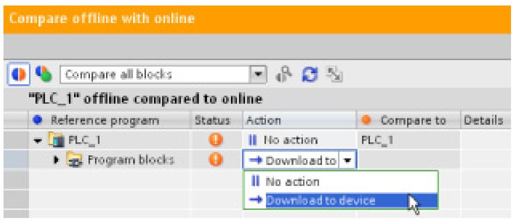
15.10 Comparing offline and online CPUs

You can compare the code blocks in an online CPU with the code blocks in your project. If the code blocks of your project do not match the code blocks of the online CPU, the "Compare" editor allows you to synchronize your project with the online CPU by downloading the code blocks of your project to the CPU, or by deleting blocks from the project that do not exist in the online CPU.



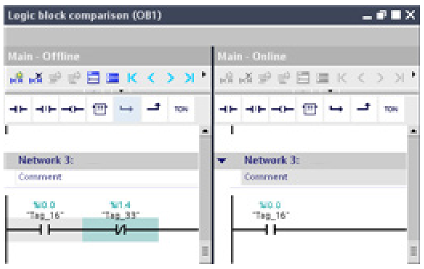
Select the CPU in your project.

Use the "Compare Offline/online" command to open the "Compare" editor. (Access the command either from the "Tools" menu or by right-clicking the CPU in your project.)



Click in the "Action" column for an object to select whether to delete the object, take no action, or download the object to the device.

Click the "Synchronize" button to load the code blocks.



Right-click an object in the "Compare to" column and select "Start detailed comparison" button to show the code blocks side-by-side.

The detailed comparison highlights the differences between the code blocks of online CPU and the code blocks of the CPU in your project.

15.11 Monitoring and modifying values in the CPU

STEP 7 provides online tools for monitoring the CPU:

- You can display or monitor the current values of the tags. The monitoring function does not change the program sequence. It presents you with information about the program sequence and the data of the program in the CPU.
- You can also use other functions to control the sequence and the data of the user program:
  - You can modify the value for the tags in the online CPU to see how the user program responds.
  - You can force a peripheral output (such as Q0.1:P or "Start":P) to a specific value.
  - You can enable outputs in STOP mode.

Note

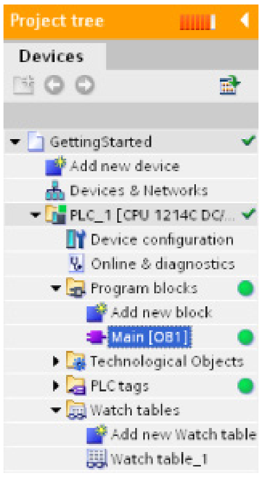
Always exercise caution when using control functions. These functions can seriously influence the execution of the user/system program.

Table 15- 3 Online capabilities of the STEP 7 editors

Editor	Monitor	Modify	Force
Watch table	Yes	Yes	No
Force table	Yes	No	Yes
Program editor	Yes	Yes	No
Tag table	Yes	No	No
DB editor	Yes	No	No

15.11.1 Going online to monitor the values in the CPU

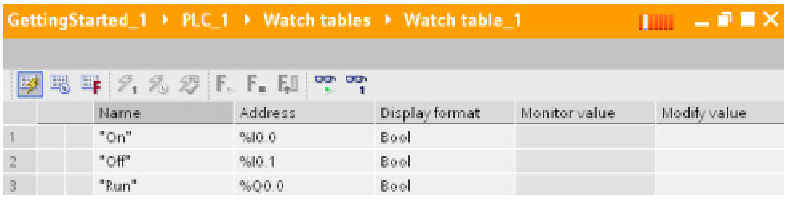
To monitor the tags, you must have an online connection to the CPU. Simply click the "Go online" button in the toolbar.



When you have connected to the CPU, STEP 7 turns the headers of the work areas orange.

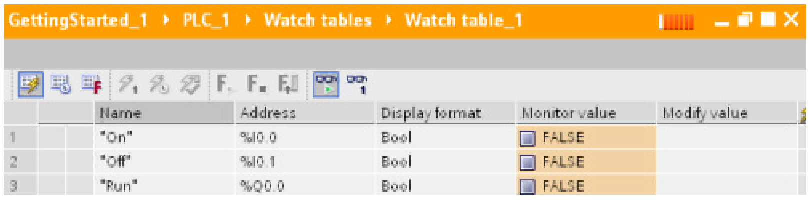
The project tree displays a comparison of the offline project and the online CPU. A green circle means that the CPU and the project are synchronized, meaning that both have the same configuration and user program.

Tag tables show the tags. Watch tables can also show the tags, as well as direct addresses.



	Name	Address	Display format	Monitor value	Modify value
1	"On"	%I0.0	Bool		
2	"Off"	%I0.1	Bool		
3	"Run"	%Q0.0	Bool		

To monitor the execution of the user program and to display the values of the tags, click the "Monitor all" button in the toolbar.



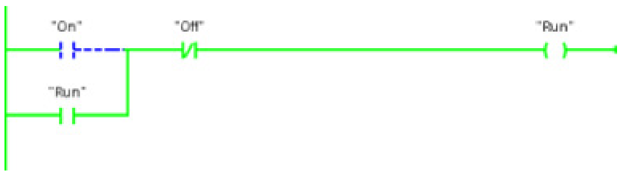
	Name	Address	Display format	Monitor value	Modify value
1	"On"	%I0.0	Bool	FALSE	
2	"Off"	%I0.1	Bool	FALSE	
3	"Run"	%Q0.0	Bool	FALSE	

The "Monitor value" field shows the value for each tag.

15.11.2 Displaying status in the program editor

You can monitor the status of up to 50 tags in the LAD and FBD program editors. Use the editor bar to display the LAD editor. The editor bar allows you to change the view between the open editors without having to open or close the editors.

In the toolbar of the program editor, click the "Monitoring on/off" button to display the status of your user program.



The network in the program editor displays power flow in green.

You can also right-click on the instruction or parameter to modify the value for the instruction.

15.11.3 Capturing the online values of a DB to reset the start values

You can capture the current values being monitored in an online CPU to become the start values for a global DB.

- You must have an online connection to the CPU.
- The CPU must be in RUN mode.
- You must have opened the DB in STEP 7.

Use the "Show a snapshot of the monitored values" button to capture the current values of the selected tags in the DB. You can then copy these values into the "Start value" column of the DB.

1. In the DB editor, click the "Monitor all tags" button. The "Monitor value" column displays the current data values.
2. Click the "Show a snapshot of the monitored values" button to display the current values in the "Snapshot" column.
3. Click the "Monitor all" button to stop monitoring the data in the CPU.
4. Copy a value in the "Snapshot" column for a tag.
   – Select a value to be copied.
   – Right-click the selected value to display the context menu.
   – Select the "Copy" command.
5. Paste the copied value into the corresponding "Start value" column for the tag. (Right-click the cell and select "Paste" from the context menu.)

6. Save the project to configure the copied values as the new start values for the DB.
7. Compile and download the DB to the CPU. The DB uses the new start values after the CPU goes to RUN mode.

Note

The values that are shown in the "Monitor value" column are always copied from the CPU. STEP 7 does not check whether all values come from the same scan cycle of the CPU.

15.11.4 Using a watch table to monitor and modify values in the CPU

A watch table allows you to perform monitoring and control functions on data points as the CPU executes your program. These data points can be process image (I or Q), M, DB or physical inputs (I_:P), depending on the monitor or control function. You cannot accurately monitor the physical outputs (Q_:P) because the monitor function can only display the last value written from Q memory and does not read the actual value from the physical outputs.

The monitoring function does not change the program sequence. It presents you with information about the program sequence and the data of the program in the CPU.

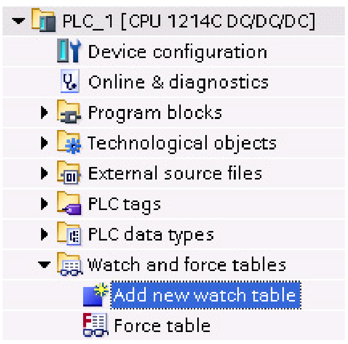
Control functions enable the user to control the sequence and the data of the program. You must exercise caution when using control functions. These functions can seriously influence the execution of the user/system program. The three control functions are Modify, Force and Enable Outputs in STOP.

With the watch table, you can perform the following online functions:

- Monitoring the status of the tags
- Modifying values for the individual tags

You select when to monitor or modify the tag:

- Beginning of scan cycle: Reads or writes the value at the beginning of the scan cycle
- End of scan cycle: Reads or writes the value at the end of the scan cycle
- Switch to stop



To create a watch table:

1. Double-click "Add new watch table" to open a new watch table.
2. Enter the tag name to add a tag to the watch table.

The following options are available for monitoring tags:

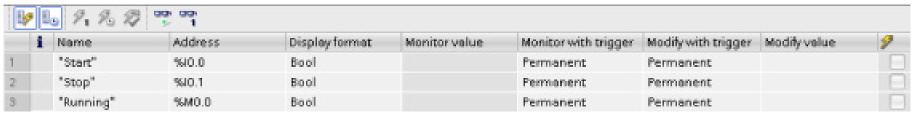
- Monitor all: This command starts the monitoring of the visible tags in the active watch table.
- Monitor now: This command starts the monitoring of the visible tags in the active watch table. The watch table monitors the tags immediately and once only.

The following options are available for modifying tags:

- "Modify to 0" sets the value of a selected address to "0".
- "Modify to 1" sets the value of a selected address to "1".
- "Modify now" immediately changes the value for the selected addresses for one scan cycle.
- "Modify with trigger" changes the values for the selected addresses.

  This function does not provide feedback to indicate that the selected addresses were actually modified. If feedback of the change is required, use the "Modify now" function.
- "Enable peripheral outputs" disables the command output disable and is available only when the CPU is in STOP mode.

To monitor the tags, you must have an online connection to the CPU.



	i	Name	Address	Display format	Monitor value	Monitor with trigger	Modify with trigger	Modify value	
1		"Start"	%I0.0	Bool		Permanent	Permanent		☐
2		"Stop"	%I0.1	Bool		Permanent	Permanent		☐
3		"Running"	%M0.0	Bool		Permanent	Permanent		☐

You use the buttons at the top of the watch table to select the various functions.

Enter the tag name to monitor and select a display format from the dropdown selection. With an online connection to the CPU, click the "Monitor" button to display the actual value of the data point in the "Monitor value" field.

15.11.4.1 Using a trigger when monitoring or modifying PLC tags

Triggering determines at what point in the scan cycle the selected address will be monitored or modified.

Table 15- 4 Types of triggers

Trigger	Description
Permanent	Continuously collects the data
At scan cycle start	Permanent: Continuously collects the data at the start of the scan cycle, after the CPU reads the inputs
	Once: Collects the data at the start of the scan cycle, after the CPU reads the inputs
At scan cycle end	Permanent: Continuously collects the data at the end of the scan cycle, before the CPU writes the outputs
	Once: Collects the data once at the end of the scan cycle, before the CPU writes the outputs
At transition to STOP	Permanent: Continuously collects data when the CPU transitions to STOP
	Once: Collects the data once after the CPU transitions to STOP

For modifying a PLC tag at a given trigger, select either the start or the end of cycle.

- Modifying an output: The best trigger event for modifying an output is at the end of the scan cycle, immediately before the CPU writes the outputs.

  Monitor the value of the outputs at the beginning of the scan cycle to determine what value is written to the physical outputs. Also, monitor the outputs before the CPU writes the values to the physical outputs in order to check program logic and to compare to the actual I/O behavior.
- Modifying an input: The best trigger event for modifying an input is at the start of the cycle, immediately after the CPU reads the inputs and before the user program uses the input values.

  If you suspect values are changing during the scan, you might want to monitor the value of the inputs at the end of the scan cycle to ensure that the value of the input at the end the scan cycle has not changed from the start of the scan cycle. If there is a difference in the values, your user program might be erroneously writing to inputs.

To diagnose why the CPU might have gone to STOP, use the "Transition to STOP" trigger to capture the last process values.

15.11.4.2 Enabling outputs in STOP mode

The watch table allows you to write to the outputs when the CPU is in STOP mode. This functionality allows you to check the wiring of the outputs and verify that the wire connected to an output pin initiates a high or low signal to the terminal of the process device to which it is connected.

⚠ WARNING

Risks in writing to physical outputs in STOP mode

Even though the CPU is in STOP mode, enabling a physical output can activate the process point to which it is connected, possibly resulting in unexpected equipment operation. Unexpected equipment operation can cause death or severe personal injury.

Before writing to an output from the watch table, ensure that changing the physical output cannot cause unexpected equipment operation. Always observe safety precautions for your process equipment.

You can change the state of the outputs in STOP mode when the outputs are enabled. If the outputs are disabled, you cannot modify the outputs in STOP mode. To enable the modification in STOP mode of the outputs from the watch table, follow these steps:

1. Select the "Expanded mode" menu command from the "Online" menu.
2. Select the "Enable peripheral outputs" option of the "Modify" command of the "Online" menu, or from the context menu after right-clicking the row of the Watch table.

You cannot enable outputs in STOP mode if you have configured distributed I/O. An error is returned when you try to do this.

Setting the CPU to RUN mode disables "Enable peripheral outputs" option.

If any inputs or outputs are forced, the CPU is not allowed to enable outputs while in STOP mode. The force function must first be cancelled.

15.11.5 Forcing values in the CPU

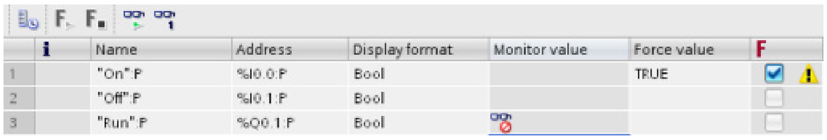
15.11.5.1 Using the force table

A force table provides a "force" function that overwrites the value for an input or output point to a specified value for the peripheral input or peripheral output address. The CPU applies this forced value to the input process image prior to the execution of the user program and to the output process image before the outputs are written to the modules.

Note

The force values are stored in the CPU and not in the force table.

You cannot force an input (or "I" address) or an output (or "Q" address). However, you can force a peripheral input or peripheral output. The force table automatically appends a ":P" to the address (for example: "On":P or "Run":P).



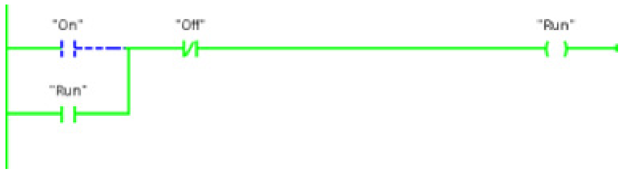
	i	Name	Address	Display format	Monitor value	Force value	F
1		"On":P	%I0.0:P	Bool		TRUE	☑ ⚠
2		"Off":P	%I0.1:P	Bool			☐
3		"Run":P	%Q0.1:P	Bool			☐

In the "Force value" cell, enter the value for the input or output to be forced. You can then use the check box in the "Force" column to enable forcing of the input or output.

Use the "Start or replace forcing" button to force the value of the tags in the force table. Click the "Stop forcing" button to reset the value of the tags.

In the force table, you can monitor the status of the forced value for an input. However, you cannot monitor the forced value of an output.

You can also view the status of the forced value in the program editor.



Note

When an input or output is forced in a force table, the force actions become part of the project configuration. If you close STEP 7, the forced elements remain active in the CPU program until they are cleared. To clear these forced elements, you must use STEP 7 to connect with the online CPU and then use the force table to turn off or stop the force function for those elements.

15.11.5.2 Operation of the Force function

The CPU allows you to force input and output point(s) by specifying the physical input or output address (I_:P or Q_:P) in the force table and then starting the force function.

In the program, reads of physical inputs are overwritten by the forced value. The program uses the forced value in processing. When the program writes a physical output, the output value is overwritten by the force value. The forced value appears at the physical output and is used by the process.

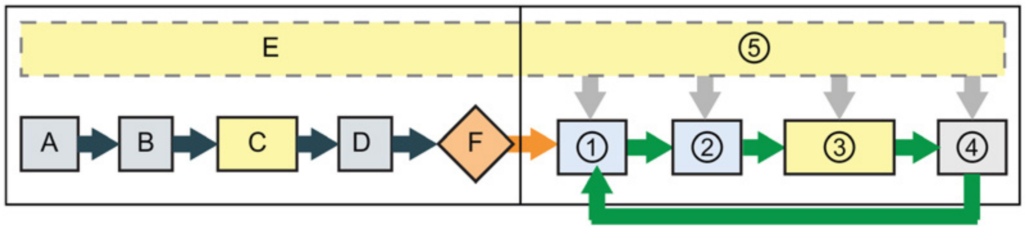
When an input or output is forced in the force table, the force actions become part of the user program. Even though the programming software has been closed, the force selections remain active in the operating CPU program until they are cleared by going online with the programming software and stopping the force function. Programs with forced points loaded on another CPU from a memory card will continue to force the points selected in the program.

If the CPU is executing the user program from a write-protected memory card, you cannot initiate or change the forcing of I/O from a watch table because you cannot override the values in the write-protected user program. Any attempt to force the write-protected values generates an error. If you use a memory card to transfer a user program, any forced elements on that memory card will be transferred to the CPU.

Note

Digital I/O points assigned to HSC, PWM, and PTO cannot be forced

The digital I/O points used by the high-speed counter (HSC), pulse-width modulation (PWM), and pulse-train output (PTO) devices are assigned during device configuration. When digital I/O point addresses are assigned to these devices, the values of the assigned I/O point addresses cannot be modified by the force function of the force table.



Startup		RUN	
A	The clearing of the I memory area is not affected by the Force function.	①	While writing Q memory to the physical outputs, the CPU applies the force value as the outputs are updated.
B	The initialization of the outputs values is not affected by the Force function.	②	When reading the physical inputs, the CPU applies the force values just prior to copying the inputs into I memory.
C	During the execution of the startup OBs, the CPU applies the force value when the user program accesses the physical input.	③	During the execution of the user program (program cycle OBs), the CPU applies the force value when the user program accesses the physical input or writes the physical output.
D	The storing of interrupt events into the queue is not affected.	④	Handling of communication requests and self-test diagnostics are not affected by the Force function.
E	The enabling of the writing to the outputs is not affected.	⑤	The processing of interrupts during any part of the scan cycle is not affected.

15.12 Downloading in RUN mode

The CPU supports "Download in RUN mode". This capability is intended to allow you to make small changes to a user program with minimal disturbance to the process being controlled by the program. However, implementing this capability also allows massive program changes that could be disruptive or even dangerous.

WARNING

Risks with downloading in RUN mode

When you download changes to the CPU in RUN mode, the changes immediately affect process operation. Changing the program in RUN mode can result in unexpected system operation, which could cause death or serious injury to personnel, and/or damage to equipment.

Only authorized personnel who understand the effects of RUN mode changes on system operation should perform a download in RUN mode.

The "Download in RUN mode" feature allows you to make changes to a program and download them to your CPU without switching to STOP mode:

- You can make minor changes to your current process without having to shut down (for example, change a parameter value).
- You can debug a program more quickly with this feature (for example, invert the logic for a normally open or normally closed switch).

You can make the following program block and tag changes and download them in RUN mode:

- Create, overwrite, and delete Functions (FC), Function Blocks (FB), and Tag tables.
- Create, delete, and overwrite Data Blocks (DB) and instance data blocks for Function Blocks (FB). You can add to DB structures and download them in RUN mode. The CPU can maintain the values of existing block tags and initialize the new data block tags to their initial values, or the CPU can set all data block tags to initial values, depending on your configuration settings (Page 1092). You cannot download a web server DB (control or fragment) in RUN mode.
- Overwrite Organization Blocks (OB); however, you cannot create or delete OBs.

You can download a maximum number of twenty blocks in RUN mode at one time. If you must download more than twenty blocks, you must place the CPU in STOP mode.

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), it is vital to think through the possible safety consequences to machines and machine operators before you download.

Note

If the CPU is in RUN mode and program changes have been made, STEP 7 always tries to download in RUN first. If you do not want this to happen, you must put the CPU into STOP.

If the changes made are not supported in "Download in RUN", STEP 7 prompts the user that the CPU must go to STOP.

15.12.1 Prerequisites for "Download in RUN mode"

To be able to download your program changes to a CPU that is in RUN mode, you must meet these prerequisites:

- Your CPU version is V3.0 or later

Note

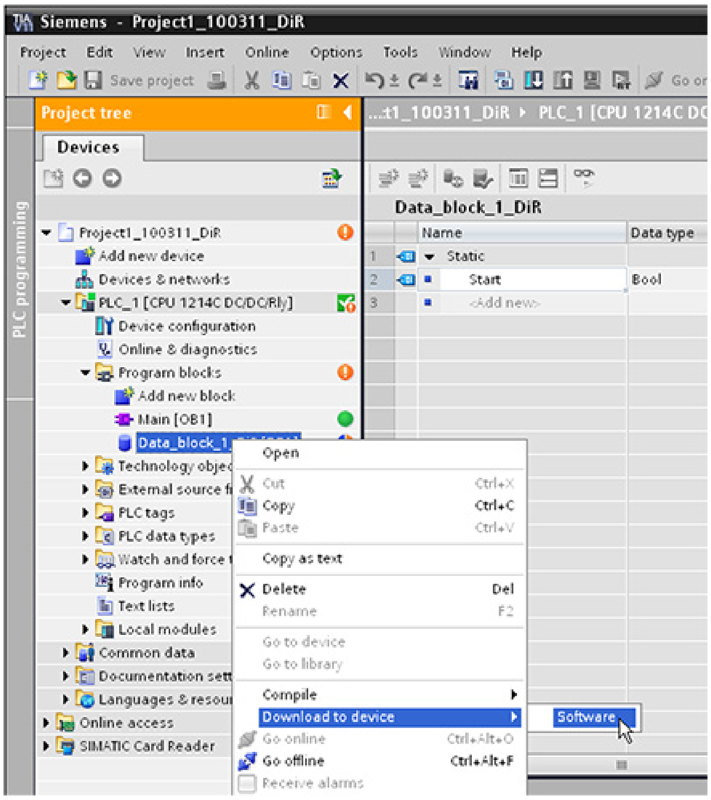
Your CPU version must be V4.0 or later to modify existing blocks and download the extended block interface in RUN mode. (Page 1092)

- Your program must compile successfully.
- You must have successfully established communication between the programming device where you are running STEP 7 and the CPU.

15.12.2 Changing your program in RUN mode

To change the program in RUN mode, your must first ensure that the CPU and program meet the prerequisites (Page 1088), and then follow these steps:

1. To download your program in RUN mode, select one of the following methods:
   – Select the "Download to device" command from the "Online" menu.
   – Click the "Download to device" button in the toolbar.
   – In the "Project tree", right-click "Program blocks" and select the "Download to device > Software" command.



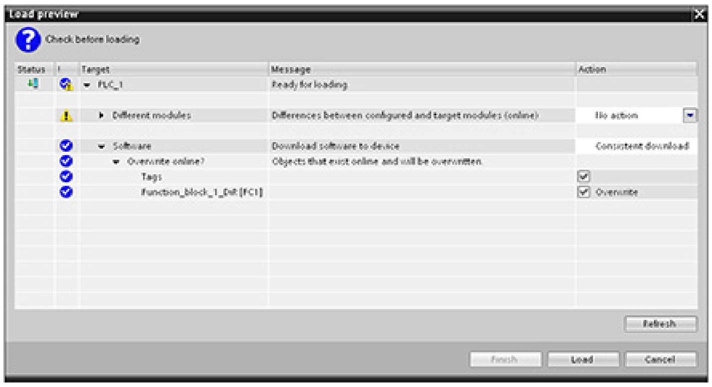
   If the program compiles successfully, STEP 7 starts to download the program to the CPU.

2. When STEP 7 prompts you to load your program or cancel the operation, click "Load" to download the program to the CPU.

15.12.3 Downloading selected blocks

From the Program blocks folder, you can select a single block or a selection of blocks for downloading.

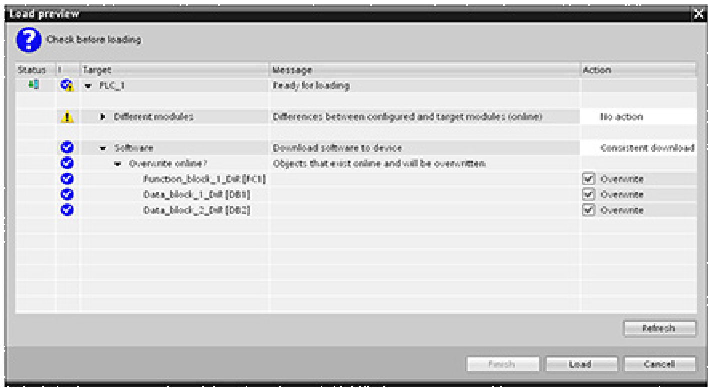
If you select a single block for downloading, then the only option in the "Action" column is "Consistent download". You can expand the category line to be sure what blocks are to be loaded. In this example, a small change was made to the offline block, and no other blocks need to be loaded.


Load preview

Check before loading

Status	!	Target	Message	Action
		PLC_1	Ready for loading	
		Different modules	Differences between configured and target modules (online)	No action
		Software	Download software to device	Consistent download
		Overwrite online?	Objects that exist online and will be overwritten.	
		Tags		✓
		Function_block_1_Diff [FC1]		✓ Overwrite

In this example, more than one block is needed for down-loading.


Load preview

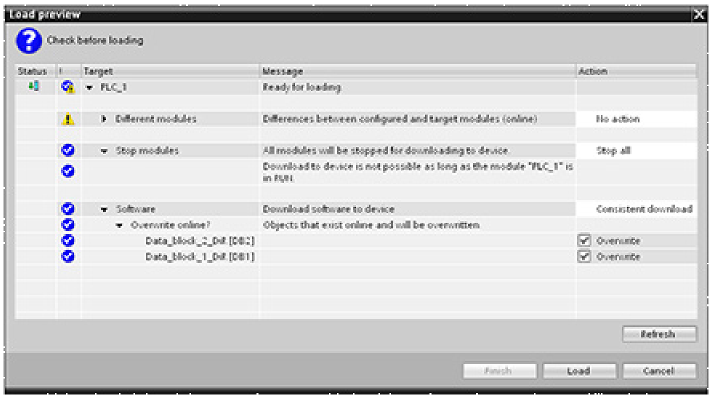
Check before loading

Status	!	Target	Message	Action
		PLC_1	Ready for loading	
		Different modules	Differences between configured and target modules (online)	No action
		Software	Download software to device	Consistent download
		Overwrite online?	Objects that exist online and will be overwritten.	
		Function_block_1_Diff [FC1]		✓ Overwrite
		Data_block_1_Diff [DB1]		✓ Overwrite
		Data_block_2_Diff [DB2]		✓ Overwrite

Note

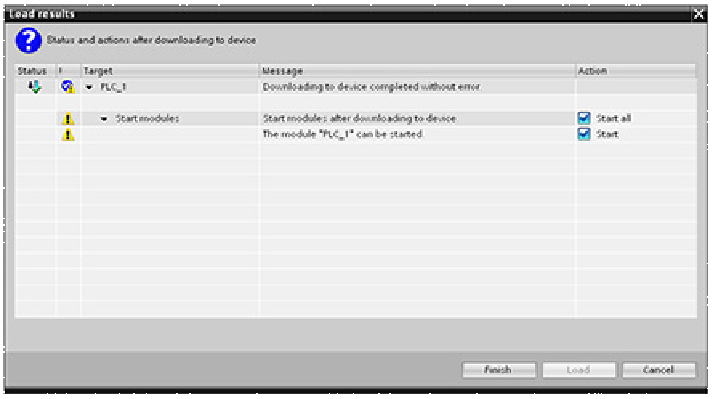
You can download a maximum number of twenty blocks in RUN mode at one time. If you must download more than twenty blocks, you must place the CPU in STOP mode.

If you attempt to download in RUN, but the system detects that this is not possible prior to the actual download, then the Stop modules category line appears in the dialog.



Status	!	Target	Message	Action
		▾ PLC_1	Ready for loading	
	⚠	▸ Different modules	Differences between configured and target modules (online)	No action
	✓	▾ Stop modules	All modules will be stopped for downloading to device	Stop all
	✓		Download to device is not possible as long as the module "PLC_1" is in RUN	
	✓	▾ Software	Download software to device	Consistent download
	✓	▾ Overwrite online?	Objects that exist online and will be overwritten	
	✓	Data_block_2_Dit [DB2]		☑ Overwrite
	✓	Data_block_1_Dit [DB1]		☑ Overwrite

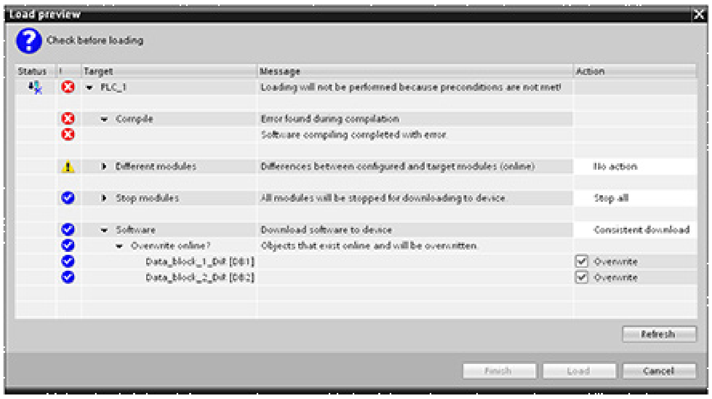
Click the "Load" button, and the "Load results" dialog appears. Click the "Finish" button to complete the download.



Status	!	Target	Message	Action
		▾ PLC_1	Downloading to device completed without error.	
	⚠	▾ Start modules	Start modules after downloading to device	☑ Start all
	⚠		The module "PLC_1" can be started	☑ Start

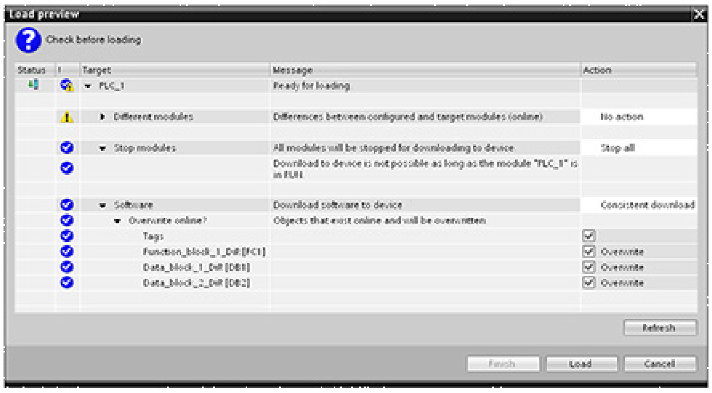
15.12.4 Downloading a single selected block with a compile error in another block

If you attempt a consistent download with a compile error in another block, then the dialog indicates an error, and the load button is disabled.



Status	!	Target	Message	Action
	✖	▾ PLC_1	Loading will not be performed because preconditions are not met!	
	✖	▾ Compile	Error found during compilation	
	✖		Software compiling completed with error	
	⚠	▸ Different modules	Differences between configured and target modules (online)	No action
	✓	▸ Stop modules	All modules will be stopped for downloading to device	Stop all
	✓	▾ Software	Download software to device	Consistent download
	✓	▾ Overwrite online?	Objects that exist online and will be overwritten	
	✓	Data_block_1_Dit [DB1]		☑ Overwrite
	✓	Data_block_2_Dit [DB2]		☑ Overwrite

You must correct the compile error in the other block. Then, the "Load" button becomes active.


Load preview

Check before loading

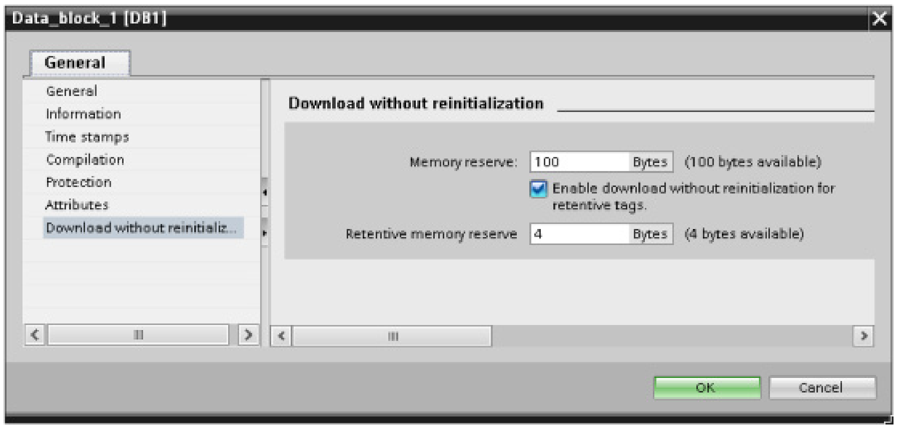
Status	!	Target	Message	Action
		PLC_1	Ready for loading	
		Different modules	Differences between configured and target modules (online)	No action
		Stop modules	All modules will be stopped for downloading to device	Stop all
			Download to device is not possible as long as the module "PLC_1" is in RUN	
		Software	Download software to device	Consistent download
		Overwrite online?	Objects that exist online and will be overwritten	
		Tags		✓
		Function_block_1_Dif [FC1]		✓ Overwrite
		Data_block_1_Dif [DB1]		✓ Overwrite
		Data_block_2_Dif [DB2]		✓ Overwrite

15.12.5 Modifying and downloading existing blocks in RUN mode

The Download in Run feature allows you to add and modify tags in data blocks and function blocks and then download the changed block to the CPU in RUN mode.

Download without reinitialization

Each DB and FB has an amount of reserved memory, which you can use for adding tags to the block that you can subsequently download in RUN mode. By default, the initial size of the memory reserve is 100 bytes. You can add additional tags to your data up to the size of the memory reserve and download the extended block to the CPU in RUN mode. You can also increase the memory reserve if you need more memory for additional tags in your block. If you add more tags than the amount of memory you have allocated, you cannot download the extended block to the CPU in RUN mode.


Data_block_1 [DB1]

Download without reinitialization

Memory reserve: 100 Bytes (100 bytes available)

✓ Enable download without reinitialization for retentive tags.

Retentive memory reserve 4 Bytes (4 bytes available)

The "Download without reinitialization" feature allows you to extend a data block by adding more data block tags and download the extended data block in RUN mode. In this way, you can add tags to a data block and download it without reinitializing your program. The CPU retains the values of the existing data block tags and initializes the newly-added tags to their start values.

To enable this function for an online project with a CPU in RUN mode, follow these steps:

1. From the Program blocks folder in the STEP 7 project tree, open the block.
2. Click the "Download without reinitialization" toggle button in the block editor to enable the function. (The icon has a box around it when you have enabled it:)
3. Click OK on the prompt to confirm your choice.
4. Add tags to the block interface and download the block in RUN mode. You can add and download as many new tags as your memory reserve allows.

If you have added more bytes to your block than you have configured for the memory reserve, STEP 7 displays an error when you attempt to download the block in RUN mode. You must edit the block properties to increase the amount. You cannot delete existing entries or modify the “Memory reserve” of the block while the “Download without reinitialization” function is enabled. To disable the "Download without reinitialization" function, follow these steps:

1. Click the "Download without reinitialization" toggle button in the block editor to disable the function. (The icon does not have a box around it when you have disabled it:)
2. Click OK on the prompt to confirm your choice.
3. Download the block. On the download dialog, you must select "reinitialize" in order to download the extended block.

The download then reinitializes all existing and new block tags to their start values.

Downloading retentive block tags

Downloading retentive block tags in RUN mode requires the allocation of a retentive memory reserve. To configure this retentive memory reserve, follow these steps:

1. From the Program blocks folder in the STEP 7 project tree, right-click the block and select "Properties" from the context menu.
2. Select the "Download without reinitialization" property.
3. Select the check box for "Enable download without reinitialization for retentive tags".
4. Configure the number of bytes available for the retentive memory reserve.
5. Click OK to save your changes.
6. Add retentive data block tags to the data block and download the data block in RUN mode. You can add and download as many new retentive data block tags as your retentive memory reserve allows.

If you have added more retentive bytes to your data block than you have configured for the retentive memory reserve, STEP 7 displays an error when you attempt to download the block in RUN mode. You can only add retentive block tags up to the retentive memory reserve in order to be able to download them in RUN mode.

When you download the extended retentive block tags, the tags contain their current values.

Configuring amount of reserved memory for new blocks

The default memory reserve size for new data blocks is 100 bytes. When you create a new block, it has 100 bytes available in reserve. If you want the memory reserve size to be different for new blocks, you can change the setting in the PLC programming settings:

1. From STEP 7, select the **Options > Settings** menu command.
2. From the Settings dialog, expand "PLC programming" and select "General".
3. In the "Download without reinitialization" section, enter the number of bytes for the memory reserve.

When you create new blocks, STEP 7 uses the memory reserve configuration that you entered for the new blocks.

Restrictions

The following restrictions apply to editing and downloading blocks in RUN mode:

- Extending the block interface by adding new tags and downloading in RUN mode is only available for optimized blocks (Page 180).
- You cannot change the structure of a block and download the changed block in RUN mode without reinitializing. Adding new members to a Struct (Page 125) tag, changing tag names, array sizes, data types, or retentive status all require that you reinitialize the block if you download it in RUN mode. The only modifications to existing block tags that you can perform and still download the block in RUN mode without reinitialization are changes to start values (data blocks), default values (function blocks) or comments.

Note

The ability to modify blocks and download them in RUN mode is new with V4.0 of the S7-1200 CPU. Prior to V4.0, you could only download modified blocks in STOP mode.

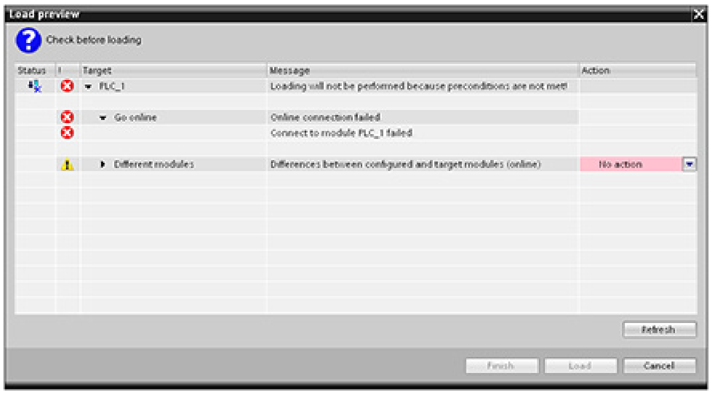
- You cannot download more new block tags in RUN mode than the memory reserve can accommodate.
- You cannot download more new retentive block tags in RUN mode than the retentive memory reserve can accommodate.

See also

Device exchange and spare parts compatibility (Page 1287)

15.12.6 System reaction if the download process fails

During the initial Download in RUN operation, if a network connection failure occurs, STEP 7 displays the following "Load preview" dialog:



Status	!	Target	Message	Action
	⊗	▾ PLC_1	Loading will not be performed because preconditions are not met!	
	⊗	▾ Go online	Online connection failed	
	⊗		Connect to module PLC_1 failed	
	⚠	▸ Different modules	Differences between configured and target modules (online)	No action

15.12.7 Considerations when downloading in RUN mode

Before downloading the program in RUN mode, consider the effect of a RUN-mode modification on the operation of the CPU for the following situations:

- If you deleted the control logic for an output, the CPU maintains the last state of the output until the next power cycle or transition to STOP mode.
- If you deleted a high-speed counter or pulse output functions which were running, the high-speed counter or pulse output continues to run until the next power cycle or transition to STOP mode.
- Any logic that is conditional on the state of the first scan bit will not be executed until the next power cycle or transition from STOP to RUN mode. The first scan bit is set only by the transition to RUN mode and is not affected by a download in RUN mode.

- The current values of data blocks (DB) and/or tags can be overwritten.

Note

Before you can download your program in RUN mode, the CPU must support changes in RUN mode, the program must compile with no errors, and the communication between STEP 7, and the CPU must be error-free.

You can make the following changes in program blocks and tags and download them in RUN mode:

- Create, overwrite, and delete Functions (FC), Function Blocks (FB), and Tag tables.
- Create and delete Data Blocks (DB); however, DB structure changes cannot be overwritten. Initial DB values can be overwritten. You cannot download a web server DB (control or fragment) in RUN mode.
- Overwrite Organization Blocks (OB); however, you cannot create or delete OBs.

You can download a maximum number of twenty blocks in RUN mode at one time. If you must download more than twenty blocks, you must place the CPU in STOP mode.

Once you initiate a download, you cannot perform other tasks in STEP 7 until the download completes.

Instructions that might fail due to "Download in RUN mode"

The following instructions might experience a temporary error when download in run changes are being activated in the CPU. The error occurs when the instruction is initiated while the CPU is preparing to activate the downloaded changes. During this time, the CPU suspends initiation of user-program access to the Load Memory, while it completes in-progress user-program access to Load Memory. This is done so that downloaded changes can be activated consistently.

Instruction	Response while Activation is Pending
DataLogCreate	STATUS = W#16#80C0, ERROR = TRUE
DataLogOpen	STATUS = W#16#80C0, ERROR = TRUE
DataLogWrite	STATUS = W#16#80C0, ERROR = TRUE
DataLogClose	STATUS = W#16#80C0, ERROR = TRUE
DataLogNewFile	STATUS = W#16#80C0, ERROR = TRUE
READ_DBL	RET_VAL = W#16#82C0
WRIT_DBL	RET_VAL = W#16#82C0
RTM	RET_VAL = 0x80C0

In all cases the RLO output from the instruction will be false when the error occurs. The error is temporary. If it occurs, the instruction should be retried later.

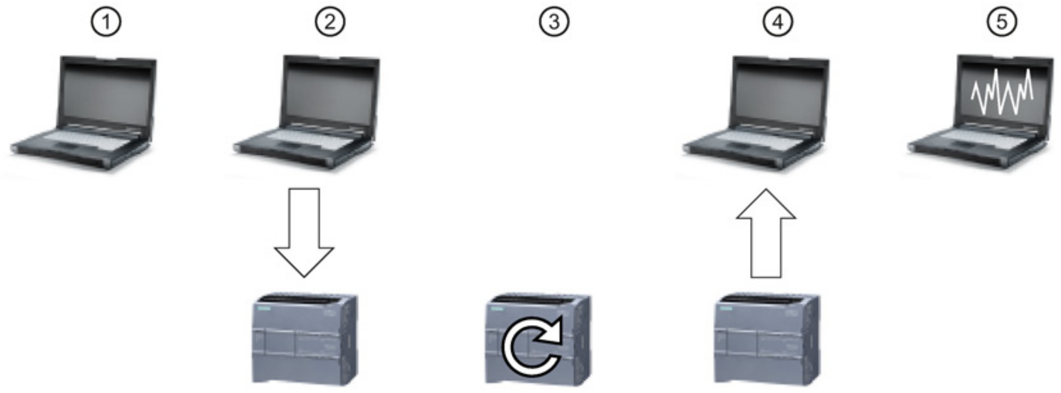
Note

You must not retry the operation in the current execution of the OB.

15.13 Tracing and recording CPU data on trigger conditions

STEP 7 provides trace and logic analyzer functions with which you can configure variables for the PLC to trace and record. You can then upload the recorded trace data to your programming device and use STEP 7 tools to analyze, manage, and graph your data. You use the Traces folder in the STEP 7 project tree to create and manage traces.

The following figure shows the various steps of the trace feature:



①	Configure the trace in the trace editor of STEP 7. You can configure the data values to record, the recording duration, the recording frequency, and the trigger condition.
②	Transfer the trace configuration from STEP 7 to the PLC.
③	The PLC executes the program, and when the trigger condition occurs, begins recording the trace data.
④	Transfer the recorded values from the PLC to STEP 7.
⑤	Use the tools in STEP 7 to analyze the data, display it graphically, and save it.

The maximum size of a trace is 512 Kbytes per trace.

Access to examples

See the STEP 7 information system for details about how to program a trace, how to download the configuration, upload the trace data, and display the data in the logic analyzer. You can find detailed examples there in the "Using online and diagnostics functions > Using the trace and logic analyzer function" chapter.

In addition the online manual "Industry Automation SINAMICS/SIMATIC Using the trace and logic analyzer function" (http://support.automation.siemens.com/WW/view/en/64897128) is an excellent reference.

Technical specifications A

A.1 General technical specifications

Standards compliance

The S7-1200 automation system design conforms with the following standards and test specifications. The test criteria for the S7-1200 automation system are based on these standards and test specifications.

Note that not all S7-1200 models may be certified to these standards, and certification status may change without notification. It is your responsibility to determine applicable certifications by referring to the ratings marked on the product. Consult your local Siemens representative if you need additional information related to the latest listing of exact approvals by part number.

CE approval

The S7-1200 Automation System satisfies requirements and safety related objectives according to the EC directives listed below, and conforms to the harmonized European standards (EN) for the programmable controllers listed in the Official Journals of the European Community.

- EC Directive 2006/95/EC (Low Voltage Directive) "Electrical Equipment Designed for Use within Certain Voltage Limits"
  - EN 61131-2:2007 Programmable controllers - Equipment requirements and tests
- EC Directive 2004/108/EC (EMC Directive) "Electromagnetic Compatibility"
  - Emission standard
    EN 61000-6-4:2007+A1:2011: Industrial Environment
  - Immunity standard
    EN 61000-6-2:2005: Industrial Environment
- EC Directive 94/9/EC (ATEX) "Equipment and Protective Systems Intended for Use in Potentially Explosive Atmosphere"
  - EN 60079-15:2010: Type of Protection 'n'

The CE Declaration of Conformity is held on file available to competent authorities at:

Siemens AG
Sector Industry
I IA AS FA DH AMB
Postfach 1963
D-92209 Amberg
Germany

cULus approval

Underwriters Laboratories Inc. complying with:

- Underwriters Laboratories, Inc.: UL 508 Listed (Industrial Control Equipment)
- Canadian Standards Association: CSA C22.2 Number 142 (Process Control Equipment)

Note

The SIMATIC S7-1200 series meets the CSA standard.

The cULus logo indicates that the S7-1200 has been examined and certified by Underwriters Laboratories (UL) to standards UL 508 and CSA 22.2 No. 142.

FM approval

Factory Mutual Research (FM)
Approval Standard Class Number 3600 and 3611
Approved for use in:
Class I, Division 2, Gas Group A, B, C, D, Temperature Class T3C Ta = 60 °C
Class I, Zone 2, IIC, Temperature Class T3 Ta = 60 °C
Canadian Class I, Zone 2 Installation per CEC 18-150

IMPORTANT EXCEPTION: See Technical Specifications for the number of inputs or outputs allowed on simultaneously. Some models are de-rated for Ta = 60 °C.

WARNING

Substitution of components can impair the suitability for Class I, Division 2 and Zone 2.

Repair of units should only be performed by an authorized Siemens Service Center.

IECEx approval

EN 60079-0: Explosive Atmospheres – General Requirements

EN60079-15: Electrical Apparatus for Potentially Explosive Atmospheres;
Type of protection ‘nA’
IECEX FMG14.0012X
Ex nA IIC Tx Gc

IECEx rating information may appear on the product with the FM Hazardous Location information.

Only products marked with an IECEx rating are approved. Consult your local Siemens representative if you need additional information related to the latest listing of exact approvals by part number.

Relay models are not included in IECEx approvals.

Refer to specific product marking for temperature rating.

Install modules in a suitable enclosure providing a minimum degree of protection of IP54 according to IEC 60079-15.

ATEX approval

ATEX approval applies to DC models only. ATEX approval does not apply to AC and Relay models.

EN 60079-0:2009: Explosive Atmospheres - General Requirements

EN 60079-15:2010: Electrical Apparatus for Potentially Explosive Atmospheres;
Type of protection 'nA'
II 3 G Ex nA IIC T4 or T3 Gc

Install modules in a suitable enclosure providing a minimum degree of protection of IP54 according to EN 60529, or in a location providing an equivalent degree of protection.

Attached cables and conductors should be rated for the actual temperature measured under rated conditions.

The installation should ensure that transients are limited to less than 119 V. See Surge immunity in this section.

IMPORTANT EXCEPTION: See Technical Specifications for the number of inputs or outputs allowed on simultaneously. Some models are de-rated for Ta = 60 °C.

C-Tick approval

The S7-1200 automation system satisfies requirements of standards to AS/NZS CISPR16 (Class A).

Korea Certification

The S7-1200 automation system satisfies the requirements of the Korean Certification (KC Mark). It has been defined as Class A Equipment and is intended for industrial applications and has not been considered for home use.

Eurasian Customs Union approval (Belarus, Kazakhstan, Russian Federation)

EAC (Eurasion Conformity): Declaration of Conformity according to Technical Regulation of Customs Union (TR CU)

Maritime approval

The S7-1200 products are periodically submitted for special agency approvals related to specific markets and applications. Consult your local Siemens representative if you need additional information related to the latest listing of exact approvals by part number.

Classification societies:

- ABS (American Bureau of Shipping)
- BV (Bureau Veritas)
- DNV (Det Norske Veritas)
- GL (Germanischer Lloyd)
- LRS (Lloyds Register of Shipping)
- Class NK (Nippon Kaiji Kyokai)
- Korean Register of Shipping

Industrial environments

The S7-1200 automation system is designed for use in industrial environments.

Table A- 1 Industrial environments

Application field	Emission requirements	Immunity requirements	Noise immunity requirements
Industrial	EN 61000-6-4:2007+A1:2011	EN 61000-6-2:2005	EN 61000-6-2:2005

Note

The S7-1200 automation system is intended for use in industrial areas; use in residential areas can have an impact on radio or TV reception. If you use the S7-1200 in residential areas, you must ensure that its radio interference emission complies with the limit value Class B in accordance with EN 55011.

Examples of suitable measures for achieving RF interference, level Class B include:

- Installation of the S7-1200 in a grounded control cabinet

- Use of noise filters in the supply lines

Ensure that the radio interference emission complies with Class B in accordance with EN 55011.

Individual acceptance is required (final installation must meet all safety and EMC requirements of a residential installation).

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	8 kV air discharge to all surfaces 6 kV contact discharge to exposed conductive surfaces
EN 61000-4-3 Radiated, radio-frequency, electromagnetic field immunity test	80 to 1000 MHz, 10 V/m, 80% AM at 1 kHz 1.4 to 2.0 GHz, 3 V/m, 80% AM at 1 kHz 2.0 to 2.7 GHz, 1 V/m, 80% AM at 1 kHz
EN 61000-4-4 Fast transient bursts	2 kV, 5 kHz with coupling network to AC and DC system power 2 kV, 5 kHz with coupling clamp to I/O
EN 6100-4-5 Surge immunity	AC systems - 2 kV common mode, 1 kV differential mode DC systems - 2 kV common mode, 1 kV differential mode For DC systems, refer to Surge immunity below
EN 61000-4-6 Conducted disturbances	150 kHz to 80 MHz, 10 V RMS, 80% AM at 1kHz
EN 61000-4-11 Voltage dips	AC systems 0% for 1 cycle, 40% for 12 cycles and 70% for 30 cycles at 60 Hz

Surge immunity

Wiring systems subject to surges from lightning strike coupling must be equipped with external protection. One specification for evaluation of protection from lightning type surges is found in EN 61000-4-5, with operational limits established by EN 61000-6-2. S7-1200 DC CPUs and signal modules require external protection to maintain safe operation when subject to surge voltages defined by this standard.

Listed below are some devices that support the needed surge immunity protection. These devices only provide the protection if they are properly installed according to the manufacturer's recommendations. Devices manufactured by other vendors with the same or better specifications can also be used:

Table A- 3 Devices that support surge immunity protection

Sub-system	Protection device
+24 VDC power	BLITZDUCTOR VT, BVT AVD 24, Part Number 918 422
Industrial Ethernet	DEHNpatch DPA M CLE RJ45B 48, Part Number 929 121
RS-485	BLITZDUCTOR XT, Basic Unit BXT BAS, Part Number 920 300
	BLITZDUCTOR XT, Module BXT ML2 BD HFS 5, Part Number 920 271
RS-232	BLITZDUCTOR XT, Basic Unit BXT BAS, Part Number 920 300
	BLITZDUCTOR XT, Module BXT ML2 BE S 12, Part Number 920 222
+24 VDC digital inputs	DEHN, Inc., Type DCO SD2 E 24, Part Number 917 988
+24 VDC digital outputs and sensor supply	DEHN, Inc., Type DCO SD2 E 24, Part Number 917 988

Sub-system	Protection device
Analog IO	DEHN, Inc., Type DCO SD2 E 12, Part Number 917 987
Relay outputs	None required

Table A- 4 Conducted and radiated emissions per EN 61000-6-4

Electromagnetic compatibility - Conducted and radiated emissions per EN 61000-6-4		
Conducted Emissions EN 55011, Class A, Group 1	0.15 MHz to 0.5 MHz	<79dB (μV) quasi-peak; <66 dB (μV) average
	0.5 MHz to 5 MHz	<73dB (μV) quasi-peak; <60 dB (μV) average
	5 MHz to 30 MHz	<73dB (μV) quasi-peak; <60 dB (μV) average
Radiated Emissions EN 55011, Class A, Group 1	30 MHz to 230 MHz	<40dB (μV/m) quasi-peak; measured at 10m
	230 MHz to 1 GHz	<47dB (μV/m) quasi-peak; measured at 10m
	1 GHz to 3 GHz	< 76dB (uV/m) quasi peak, measured at 10m

Environmental conditions

Table A- 5 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 60068-2-32, Free fall	0.3 m, 5 times, product packaging
Atmospheric pressure	1080 to 660h Pa (corresponding to an altitude of -1000 to 3500m)

Table A- 6 Operating conditions

Environmental conditions - Operating	
Ambient temperature range (Inlet Air 25 mm below unit)	-20 °C to 60 °C horizontal mounting -20 °C to 50 °C vertical mounting 95% non-condensing humidity Unless otherwise specified
Atmospheric pressure	1080 to 795 hPa (corresponding to an altitude of -1000 to 2000m)
Concentration of contaminants	$S0_2$: < 0.5 ppm; H_2S: < 0.1 ppm; RH < 60% non-condensing
	ISA-S71.04 severity level G1, G2, G3
EN 60068-2-14, Test Nb, temperature change	5 °C to 55 °C, 3 °C/minute

Environmental conditions - Operating	
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 5-9 Hz, 1G from 9 - 150 Hz Panel Mount: 7.0 mm from 5-9 Hz, 2G from 9 to 150 Hz 10 sweeps each axis, 1 octave per minute

Table A- 7 High potential isolation test

High potential isolation test	
24 VDC / 5 VDC nominal circuits	520 VDC (type test of optical isolation boundaries)
115 VAC / 230 VAC circuits to ground	1500 VAC
115 VAC / 230 VAC circuits to 115 VAC / 230 VAC circuits	1500 VAC
115 VAC / 230 VAC circuits to 24 VDC / 5 VDC circuits	1500 VAC (3000 VAC/4242 VDC type test)
Ethernet port to 24 VDC / 5 VDC circuits and ground[1]	1500 VAC (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.

Protection class

- Protection Class II according to EN 61131-2 (Protective conductor not required)

Degree of protection

- IP20 Mechanical Protection, EN 60529
- Protects against finger contact with high voltage as tested by standard probe. External protection required for dust, dirt, water and foreign objects of < 12.5mm in diameter.

Rated voltages

Table A- 8 Rated voltages

Rated voltage	Tolerance
24 VDC	20.4 VDC to 28.8 VDC
120/230 VAC	85 VAC to 264 VAC, 47 to 63 Hz

Reverse voltage protection

Reverse voltage protection circuitry is provided on each terminal pair of +24 VDC power or user input power for CPUs, signal modules (SMs), and signal boards (SBs). It is still possible to damage the system by wiring different terminal pairs in opposite polarities.

Some of the 24 VDC power input ports in the S7-1200 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 VDC power supply of the CPU, the sensor power of the CPU, the power input for the relay coil of an SM, and the power supply for a non-isolated analog input. All non-isolated M terminals must connect to the same external reference potential.

WARNING

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 S7-1200 system are connected to the same reference potential.

DC Outputs

Short -circuit protection circuitry is not provided for DC outputs on CPUs, signal modules (SMs) and signal boards (SBs).

Relay electrical service life

The typical performance data estimated from sample tests 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. N.C. contacts have a typical service life of about one-third that of the N.O. contact under inductive and lamp load conditions.

An external protective circuit will increase the service life of the contacts.

Table A- 9 Typical performance data

Data for selecting an actuator				
Continuous thermal current		2 A max.		
Switching capacity and life of the contacts				
	For ohmic load	Voltage	Current	Number of operating cycles (typical)
		24 VDC	2.0 A	0.1 million
		24 VDC	1.0 A	0.2 million
		24 VDC	0.5 A	1.0 million
		48 VAC	1.5 A	1.5 million
		60 VAC	1.5 A	1.5 million
		120 VAC	2.0 A	1.0 million
		120 VAC	1.0 A	1.5 million
		120 VAC	0.5 A	2.0 million
		230 VAC	2.0 A	1.0 million
		230 VAC	1.0 A	1.5 million
		230 VAC	0.5 A	2.0 million
	For inductive load (according to IEC 947-5-1 DC13/AC15)	Voltage	Current	Number of operating cycles (typical)
		24 VDC	2.0 A	0.05 million
		24 VDC	1.0 A	0.1 million
		24 VDC	0.5 A	0.5 million
		24 VAC	1.5 A	1.0 million
		48 VAC	1.5 A	1.0 million
		60 VAC	1.5 A	1.0 million
		120 VAC	2.0 A	0.7 million
		120 VAC	1.0 A	1.0 million
		120 VAC	0.5 A	1.5 million
		230 VAC	2.0 A	0.7 million
		230 VAC	1.0 A	1.0 million
		230 VAC	0.5 A	1.5 million
Activating a digital input		Possible		
Switching frequency				
	Mechanical	Max. 10 Hz		
	At ohmic load	Max. 1 Hz		
	At inductive load (according to IEC 947-5-1 DC13/AC15)	Max. 0.5 Hz		
	At lamp load	Max. 1Hz		

Internal CPU memory retention

- Lifetime of retentive data and data log data: 10 years
- Power down retentive data, Write cycle endurance: 2 million cycles
- Data log data, up to 2 KB per data log entry, Write cycle endurance: 500 million data log entries

---

Note

Effect of data logs on internal CPU memory

Each data log write consumes at a minimum 2 KB of memory. If your program writes small amounts of data frequently, it is consuming at least 2 KB of memory on each write. A better implementation would be to accumulate the small data items in a data block (DB), and to write the data block to the data log at less frequent intervals.

If your program writes many data log entries at a high frequency, consider using a replaceable SD memory card.

A.2 CPU 1211C

A.2.1 General specifications and features

Table A- 10 General specifications

Technical data	CPU 1211C AC/DC/Relay	CPU 1211C DC/DC/Relay	CPU 1211C DC/DC/DC
Article number	6ES7 211-1BE40-0XB0	6ES7 211-1HE40-0XB0	6ES7 211-1AE40-0XB0
Dimensions W x H x D (mm)	90 x 100 x 75		
Shipping weight	420 grams	380 grams	370 grams
Power dissipation	10 W	8 W	
Current available (CM bus)	750 mA max. (5 VDC)		
Current available (24 VDC)	300 mA max. (sensor power)		
Digital input current consumption (24 VDC)	4 mA/input used		

Table A- 11 CPU features

Technical data		Description
User memory (Refer to "General technical specifications" (Page 1099), "Internal CPU memory retention".)	Work	50 Kbytes
	Load	1 Mbyte internal, expandable up to SD card size
	Retentive	10 Kbytes
On-board digital I/O		6 inputs/4 outputs
On-board analog I/O		2 inputs
Process image size		1024 bytes of inputs (I) /1024 bytes of outputs (Q)
Bit memory (M)		4096 bytes
Temporary (local) memory		• 16 Kbytes for startup and program cycle (including associated FBs and FCs) • 6 Kbytes for each of the other interrupt priority levels (including FBs and FCs)
Signal modules expansion		none
SB, CB, BB expansion		1 max.
Communication module expansion		3 CMs max.
High-speed counters		Up to 6 configured to use any built-in or SB inputs. Refer to table, CPU 1211C: HSC default address assignments (Page 463) 100/[1]80 kHz (Ia.0 to Ia.5)

Technical data	Description
Pulse outputs[2]	Up to 4 configured to use any built-in or SB outputs 100 kHz (Qa.0 to Qa.3)
Pulse catch inputs	6
Time delay interrupts	4 total with 1 ms resolution
Cyclic interrupts	4 total with 1 ms resolution
Edge interrupts	6 rising and 6 falling (10 and 10 with optional signal board)
Memory card	SIMATIC Memory Card (optional)
Real time clock accuracy	+/- 60 seconds/month
Real time clock retention time	20 days typ./12 days min. at 40 °C (maintenance-free Super Capacitor)

1 The slower speed is applicable when the HSC is configured for quadrature mode of operation.

2 For CPU models with relay outputs, you must install a digital signal board (SB) to use the pulse outputs.

Table A- 12 Performance

Type of instruction	Execution speed
Boolean	0.08 μs/instruction
Move Word	1.7 μs/instruction
Real math	2.3 μs/instruction

A.2.2 Timers, counters, and code blocks supported by CPU 1211C

Table A- 13 Blocks, timers and counters supported by CPU 1211C

Element		Description
Blocks	Type	OB, FB, FC, DB
	Size	30 Kbytes
	Quantity	Up to 1024 blocks total (OBs + FBs + FCs + DBs)
	Address range for FBs, FCs, and DBs	FB and FC: 1 to 65535 (such as FB 1 to FB 65535) DB: 1 to 59999
	Nesting depth	16 from the program cycle or startup OB 6 from any interrupt event OB
	Monitoring	Status of 2 code blocks can be monitored simultaneously
OBs	Program cycle	Multiple
	Startup	Multiple
	Time-delay interrupt	4 (1 per event)
	Cyclic interrupts	4 (1 per event)
	Hardware interrupts	50 (1 per event)
	Time error interrupts	1
	Diagnostic error interrupts	1
	Pull or plug of modules	1
	Rack or station failure	1

Element		Description
	Time of day	Multiple
	Status	1
	Update	1
	Profile	1
Timers	Type	IEC
	Quantity	Limited only by memory size
	Storage	Structure in DB, 16 bytes per timer
Counters	Type	IEC
	Quantity	Limited only by memory size
	Storage	Structure in DB, size dependent upon count type • SInt, USInt: 3 bytes • Int, UInt: 6 bytes • DInt, UDInt: 12 bytes

Table A- 14 Communication

Technical data	Description
Number of ports	1
Type	Ethernet
HMI device	4
Programming device (PG)	1
Connections	• 8 for Open User Communication (active or passive): TSEND_C, TRCV_C, TCON, TDISCON, TSEND, and TRCV • 3 for server GET/PUT (CPU-to-CPU) S7 communication • 8 for client GET/PUT (CPU-to-CPU) S7 communication
Data rates	10/100 Mb/s
Isolation (external signal to PLC logic)	Transformer isolated, 1500 VAC, for short term event safety only
Cable type	CAT5e shielded

Table A- 15 Power supply

Technical data		CPU 1211C AC/DC/Relay	CPU 1211C DC/DC/Relay	CPU 1211C DC/DC/DC
Voltage range		85 to 264 VAC	20.4 VDC to 28.8 VDC	
Line frequency		47 to 63 Hz	--	
Input current	CPU only at max. load	60 mA at 120 VAC 30 mA at 240 VAC	30 mA at 24 VDC	300 mA at 24 VDC
	CPU with all expansion accessories at max. load	180 mA at 120 VAC 90 mA at 240 VAC	900 mA at 24 VDC	
Inrush current (max.)		20 A at 264 VAC	12 A at 28.8 VDC	
Isolation (input power to logic)		1500 VAC	Not isolated	
Ground leakage, AC line to functional earth		0.5 mA max.	--	
Hold up time (loss of power)		20 ms at 120 VAC 80 ms at 240 VAC	10 ms at 24 VDC	
Internal fuse, not user replaceable		3 A, 250 V, slow blow		

Table A- 16 Sensor power

Technical data	CPU 1211C AC/DC/Relay	CPU 1211C DC/DC/Relay	CPU 1211C DC/DC/DC
Voltage range	20.4 to 28.8 VDC	L+ minus 4 VDC min.	
Output current rating (max.)	300 mA (short-circuit protected)		
Maximum ripple noise (<10 MHz)	< 1 V peak to peak	Same as input line	
Isolation (CPU logic to sensor power)	Not isolated		

A.2.3 Digital inputs and outputs

Table A- 17 Digital inputs

Technical data	CPU 1211C AC/DC/Relay, CPU 1211C DC/DC/Relay, and CPU 1211C DC/DC/DC
Number of inputs	6
Type	Sink/Source (IEC Type 1 sink)
Rated voltage	24 VDC at 4 mA, nominal
Continuous permissible voltage	30 VDC, max.
Surge voltage	35 VDC for 0.5 sec.
Logic 1 signal (min.)	15 VDC at 2.5 mA
Logic 0 signal (max.)	5 VDC at 1 mA
Isolation (field side to logic)	500 VAC for 1 minute
Isolation groups	1
Filter times	us settings: 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 10.0, 12.8, 20.0 ms settings: 0.05, 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 10.0, 12.8, 20.0
HSC clock input rates (max.) (Logic 1 Level = 15 to 26 VDC)	100/80 kHz (Ia.0 to Ia.5)
Number of inputs on simultaneously	6 at 60 °C horizontal, 50 °C vertical
Cable length (meters)	500 m shielded, 300 m unshielded, 50 m shielded for HSC inputs

Table A- 18 Digital outputs

Technical data	CPU 1211C AC/DC/Relay and CPU 1211C DC/DC/Relay	CPU 1211C DC/DC/DC
Number of outputs	4	
Type	Relay, mechanical	Solid state - MOSFET (sourcing)
Voltage range	5 to 30 VDC or 5 to 250 VAC	20.4 to 28.8 VDC
Logic 1 signal at max. current	--	20 VDC min.
Logic 0 signal with 10 KΩ load	--	0.1 VDC max.
Current (max.)	2.0 A	0.5 A
Lamp load	30 W DC / 200 W AC	5 W
ON state resistance	0.2 Ω max. when new	0.6 Ω max.
Leakage current per point	--	10 μA max.
Surge current	7 A with contacts closed	8 A for 100 ms max.
Overload protection	No	
Isolation (field side to logic)	1500 VAC for 1 minute (coil to contact) None (coil to logic)	500 VAC for 1 minute
Isolation resistance	100 MΩ min. when new	--
Isolation between open contacts	750 VAC for 1 minute	--
Isolation groups	1	
Inductive clamp voltage	--	L+ minus 48 VDC, 1 W dissipation

Technical data	CPU 1211C AC/DC/Relay and CPU 1211C DC/DC/Relay	CPU 1211C DC/DC/DC
Maximum relay switching frequency	1 Hz	--
Switching delay (Qa.0 to Qa.3)	10 ms max.	1.0 μs max., off to on 3.0 μs max., on to off
Pulse Train Output rate	Not recommended [1]	100 kHz (Qa.0 to Qa.3)[2], 2 Hz min.
Lifetime mechanical (no load)	10,000,000 open/close cycles	--
Lifetime contacts at rated load	100,000 open/close cycles	--
Behavior on RUN to STOP	Last value or substitute value (default value 0)	
Number of outputs on simultaneously	4 at 60 °C horizontal, 50 °C vertical	
Cable length (meters)	500 m shielded, 150 m unshielded	

[1] For CPU models with relay outputs, you must install a digital signal board (SB) to use the pulse outputs.

[2] Depending on your pulse receiver and cable, an additional load resistor (at least 10% of rated current) may improve pulse signal quality and noise immunity.

A.2.4 Analog inputs

Table A- 19 Analog inputs

Technical data	Description
Number of inputs	2
Type	Voltage (single-ended)
Full-scale range	0 to 10 V
Full-scale range (data word)	0 to 27648
Overshoot range	10.001 to 11.759 V
Overshoot range (data word)	27649 to 32511
Overflow range	11.760 to 11.852 V
Overflow range (data word)	32512 to 32767
Resolution	10 bits
Maximum withstand voltage	35 VDC
Smoothing	None, Weak, Medium, or Strong See the table for Step response (ms) for the analog inputs of the CPU (Page 1115).
Noise rejection	10, 50, or 60 Hz
Impedance	≥100 KΩ
Isolation (field side to logic)	None
Accuracy (25 °C / 0 to 55 °C)	3.0% / 3.5% of full-scale
Cable length (meters)	100 m, shielded twisted pair

A.2.4.1 Step response of the built-in analog inputs of the CPU

Table A- 20 Step Response (ms), 0 V to 10 V measured at 95%

Smoothing selection (sample averaging)	**Rejection frequency (Integration time)**		
	60 Hz	**50 Hz**	**10 Hz**
None (1 cycle): No averaging	50 ms	50 ms	100 ms
Weak (4 cycles): 4 samples	60ms	70 ms	200 ms
Medium (16 cycles): 16 samples	200 ms	240 ms	1150 ms
Strong (32 cycles): 32 samples	400 ms	480 ms	2300 ms
Sample time	**4.17 ms**	**5 ms**	**25 ms**

A.2.4.2 Sample time for the built-in analog ports of the CPU

Table A- 21 Sample time for built-in analog inputs of the CPU

Rejection frequency (Integration time selection)	**Sample time**
60 Hz (16.6 ms)	4.17 ms
50 Hz (20 ms)	5 ms
10 Hz (100 ms)	25 ms

A.2.4.3 Measurement ranges of the analog inputs for voltage (CPUs)

Table A- 22 Analog input representation for voltage (CPUs)

System		**Voltage Measuring Range**	
Decimal	**Hexadecimal**	**0 to 10 V**	
32767	7FFF	11.851 V	Overflow
32512	7F00		
32511	7EFF	11.759 V	Overshoot range
27649	6C01		
27648	6C00	10 V	Rated range
20736	5100	7.5 V	
34	22	12 mV	
0	0	0 V	
Negative values		Negative values are not supported	

A.2.5 CPU 1211 wiring diagrams

Table A- 23 CPU 1211C AC/DC/Relay (6ES7 211-1BE40-0XB0)

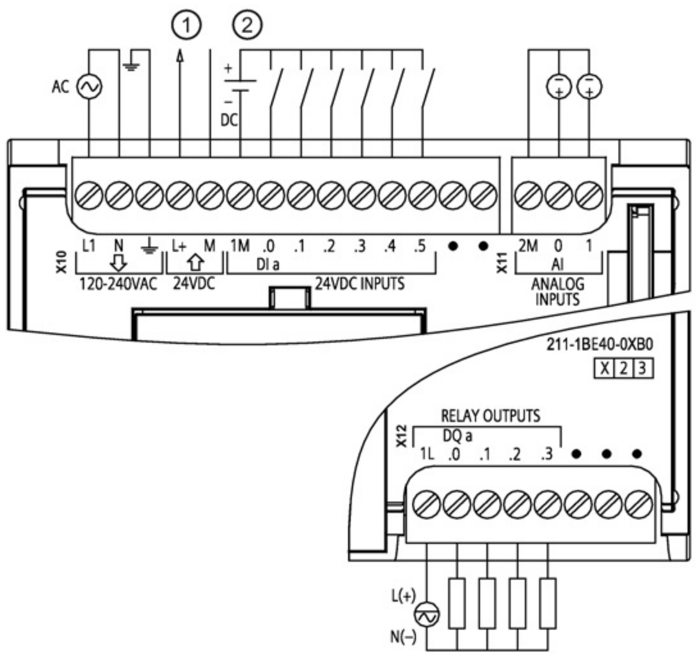
 ① ② AC DC + − L1 N L+ M 1M .0 .1 .2 .3 .4 .5 2M 0 1 X10 120-240VAC 24VDC DI a 24VDC INPUTS X11 AI ANALOG INPUTS 211-1BE40-0XB0 X 2 3 RELAY OUTPUTS X12 DQ a 1L .0 .1 .2 .3 L(+) N(−)	①	24 VDC Sensor Power Out For additional noise immunity, connect "M" to chassis ground even if not using sensor supply.
	②	For sinking inputs, connect "-" to "M" (shown). For sourcing inputs, connect "+" to "M".
	Note 1: X11 connectors must be gold. See Appendix C, Spare Parts for article number.	
	Note 2: Either the L1 or N (L2) terminal can be connected to a voltage source up to 240 VAC. The N terminal can be considered L2 and is not required to be grounded. No polarization is required for L1 and N (L2) terminals.	
	Note 3: See Device Configuration (Page 145) for information about the Ethernet port of the CPU.	

Table A- 24 Connector pin locations for CPU 1211C AC/DC/Relay (6ES7 211-1BE40-0XB0)

Pin	X10	X11 (gold)	X12
1	L1 / 120-240 VAC	2 M	1L
2	N / 120-240 VAC	AI 0	DQ a.0
3	Functional Earth	AI 1	DQ a.1
4	L+ / 24 VDC Sensor Out	--	DQ a.2
5	M / 24 VDC Sensor Out	--	DQ a.3
6	1M	--	No connection
7	DI a.0	--	No connection
8	DI a.1	--	No connection
9	DI a.2	--	--
10	DI a.3	--	--
11	DI a.4	--	--
12	DI a.5	--	--

Pin	X10	X11 (gold)	X12
13	No connection	--	--
14	No connection	--	--

Table A- 25 CPU 1211C DC/DC/Relay (6ES7 211-1HE40-0XB0)

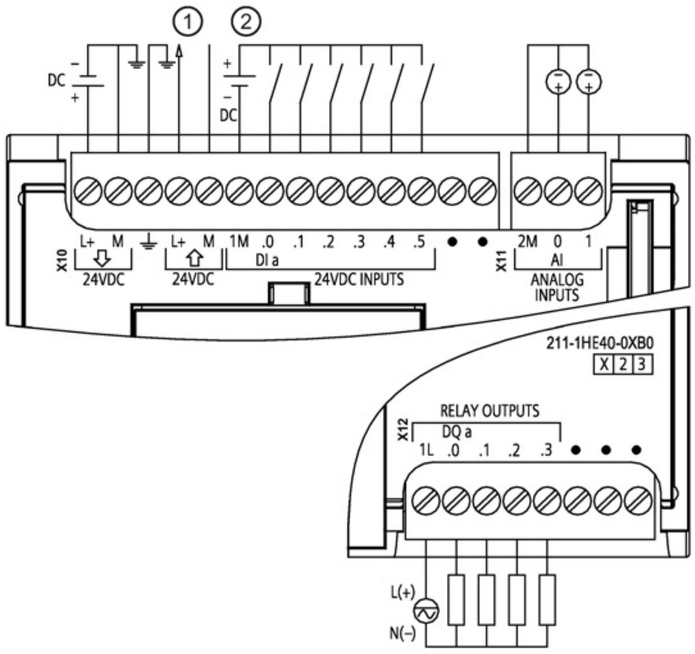
 ① ② DC DC L+ M ⏚ L+ M 1M .0 .1 .2 .3 .4 .5 ● ● X10 24VDC 24VDC DI a 24VDC INPUTS 2M 0 1 X11 AI ANALOG INPUTS 211-1HE40-0XB0 X 2 3 RELAY OUTPUTS X12 DQ a 1L .0 .1 .2 .3 ● ● ● L(+) N(-)	①	24 VDC Sensor Power Out For additional noise immunity, connect "M" to chassis ground even if not using sensor supply.
	②	For sinking inputs, connect "-" to "M" (shown). For sourcing inputs, connect "+" to "M".
	Note 1: X11 connectors must be gold. See Appendix C, Spare Parts for article number.	
	Note 2: See Device Configuration (Page 145) for information about the Ethernet port of the CPU.	

Table A- 26 Connector pin locations for CPU 1211C DC/DC/Relay (6ES7 211-1HE40-0XB0)

Pin	X10	X11 (gold)	X12
1	L+ / 24 VDC	2 M	1L
2	M / 24 VDC	AI 0	DQ a.0
3	Functional Earth	AI 1	DQ a.1
4	L+ / 24 VDC Sensor Out	--	DQ a.2
5	M / 24 VDC Sensor Out	--	DQ a.3
6	1M	--	No connection
7	DI a.0	--	No connection
8	DI a.1	--	No connection
9	DI a.2	--	--
10	DI a.3	--	--
11	DI a.4	--	--
12	DI a.5	--	--
13	No connection	--	--
14	No connection	--	--

Table A- 27 CPU 1211C DC/DC/DC (6ES7 211-1AE40-0XB0)

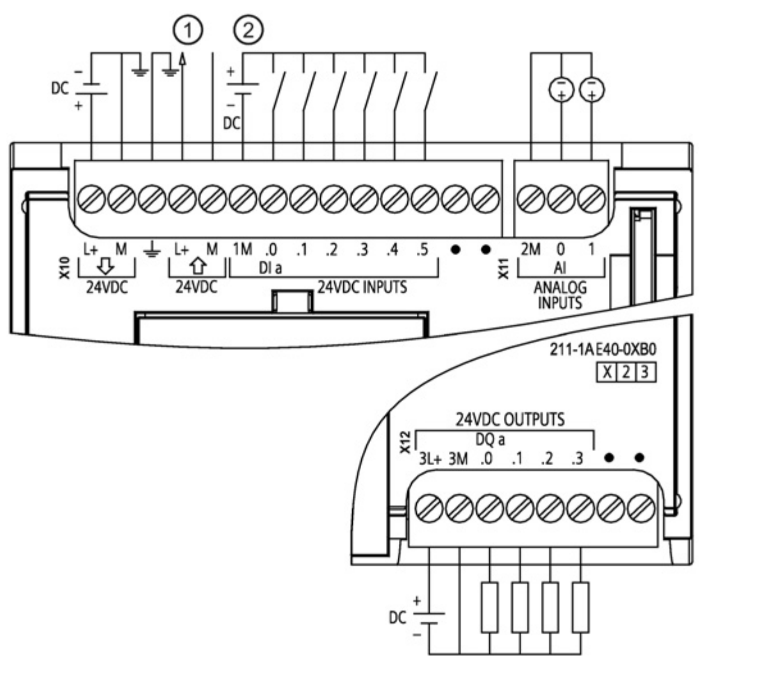
 ① ② DC – + + – DC L+ M L+ M 1M .0 .1 .2 .3 .4 .5 ● ● X10 24VDC 24VDC DI a 24VDC INPUTS 2M 0 1 X11 AI ANALOG INPUTS 211-1AE40-0XB0 X 2 3 24VDC OUTPUTS X12 DQ a 3L+ 3M .0 .1 .2 .3 ● ● DC + –	①	24 VDC Sensor Power Out For additional noise immunity, connect "M" to chassis ground even if not using sensor supply.
	②	For sinking inputs, connect "-" to "M" (shown). For sourcing inputs, connect "+" to "M".
	Note 1: X11 connectors must be gold. See Appendix C, Spare Parts for article number.	
	Note 2: See Device Configuration (Page 145) for information about the Ethernet port of the CPU.	

Table A- 28 Connector pin locations for CPU 1211C DC/DC/DC (6ES7 211-1AE40-0XB0)

Pin	X10	X11 (gold)	X12
1	L+ / 24 VDC	2 M	3L+
2	M / 24 VDC	AI 0	3M
3	Functional Earth	AI 1	DQ a.0
4	L+ / 24 VDC Sensor Out	--	DQ a.1
5	M / 24 VDC Sensor Out	--	DQ a.2
6	1M	--	DQ a.3
7	DI a.0	--	No connection
8	DI a.1	--	No connection
9	DI a.2	--	--
10	DI a.3	--	--
11	DI a.4	--	--
12	DI a.5	--	--
13	No connection	--	--
14	No connection	--	--

Note

Unused analog inputs should be shorted.

A.3 CPU 1212C

A.3.1 General specifications and features

Table A- 29 General

Technical data	CPU 1212C AC/DC/Relay	CPU 1212C DC/DC/Relay	CPU 1212C DC/DC/DC
Article number	6ES7 212-1BE40-0XB0	6ES7 212-1HE40-0XB0	6ES7 212-1AE40-0XB0
Dimensions W x H x D (mm)	90 x 100 x 75		
Shipping weight	425 grams	385 grams	370 grams
Power dissipation	11 W	9 W	
Current available (SM and CM bus)	1000 mA max. (5 VDC)		
Current available (24 VDC)	300 mA max. (sensor power)		
Digital input current consumption (24 VDC)	4 mA/input used		

Table A- 30 CPU features

Technical data		Description
User memory (Refer to "General technical specifications (Page 1099)", "Internal CPU memory retention".)	Work	75 Kbytes
	Load	1 Mbyte internal, expandable up to SD card size
	Retentive	10 Kbytes
On-board digital I/O		8 inputs/6 outputs
On-board analog I/O		2 inputs
Process image size		1024 bytes of inputs (I)/1024 bytes of outputs (Q)
Bit memory (M)		4096 bytes
Temporary (local) memory		• 16 Kbytes for startup and program cycle (including associated FBs and FCs) • 6 Kbytes for each of the other interrupt priority levels (including FBs and FCs)
Signal modules expansion		2 SMs max.
SB, CB, BB expansion		1 max.
Communication module expansion		3 CMs max.
High-speed counters		Up to 6 configured to use any built-in or SB inputs. See table, CPU 1212C: HSC default address assignments (Page 463) • 100/[1]80 kHz (Ia.0 to Ia.5) • 30 /[1]20 kHz (Ia.6 to Ia.7)

Technical data	Description
Pulse outputs[2]	Up to 4 configured to use any built-in or SB outputs • 100 kHz (Qa.0 to Qa.3) • 30 kHz (Qa.4 to Qa.5)
Pulse catch inputs	8
Time delay interrupts	4 total with 1 ms resolution
Cyclic interrupts	4 total with 1 ms resolution
Edge interrupts	8 rising and 8 falling (12 and 12 with optional signal board)
Memory card	SIMATIC Memory Card (optional)
Real time clock accuracy	+/- 60 seconds/month
Real time clock retention time	20 days typ./12 days min. at 40 °C (maintenance-free Super Capacitor)

1 The slower speed is applicable when the HSC is configured for quadrature mode of operation.

2 For CPU models with relay outputs, you must install a digital signal board (SB) to use the pulse outputs.

Table A- 31 Performance

Type of instruction	Execution speed
Boolean	0.08 μs/instruction
Move Word	1.7 μs/instruction
Real math	2.3 μs/instruction

A.3.2 Timers, counters, and code blocks supported by CPU 1212C

Table A- 32 Blocks, timers and counters supported by CPU 1212C

Element		Description
Blocks	Type	OB, FB, FC, DB
	Size	50 Kbytes
	Quantity	Up to 1024 blocks total (OBs + FBs + FCs + DBs)
	Address range for FBs, FCs, and DBs	FB and FC: 1 to 65535 (such as FB 1 to FB 65535) DB: 1 to 59999
	Nesting depth	16 from the program cycle or startup OB 6 from any interrupt event OB
	Monitoring	Status of 2 code blocks can be monitored simultaneously
OBs	Program cycle	Multiple
	Startup	Multiple
	Time-delay interrupt	4 (1 per event)
	Cyclic interrupts	4 (1 per event)
	Hardware interrupts	50 (1 per event)
	Time error interrupts	1
	Diagnostic error interrupts	1
	Pull or plug of modules	1

Element		Description
	Rack or station failure	1
	Time of day	Multiple
	Status	1
	Update	1
	Profile	1
Timers	Type	IEC
	Quantity	Limited only by memory size
	Storage	Structure in DB, 16 bytes per timer
Counters	Type	IEC
	Quantity	Limited only by memory size
	Storage	Structure in DB, size dependent upon count type • SInt, USInt: 3 bytes • Int, UInt: 6 bytes • DInt, UDInt: 12 bytes

Table A- 33 Communication

Technical data	Description
Number of ports	1
Type	Ethernet
HMI device	4
Programming device (PG)	1
Connections	• 8 for Open User Communication (active or passive): TSEND_C, TRCV_C, TCON, TDISCON, TSEND, and TRCV • 3 for server GET/PUT (CPU-to-CPU) S7 communication • 8 for client GET/PUT (CPU-to-CPU) S7 communication
Data rates	10/100 Mb/s
Isolation (external signal to PLC logic)	Transformer isolated, 1500 VAC, for short term event safety only
Cable type	CAT5e shielded

Table A- 34 Power supply

Technical data		CPU 1212C AC/DC/Relay	CPU 1212C DC/DC/Relay	CPU 1212C DC/DC/DC
Voltage range		85 to 264 VAC	20.4 VDC to 28.8 VDC	
Line frequency		47 to 63 Hz	--	
Input current (max. load)	CPU only	80 mA at 120 VAC 40 mA at 240 VAC	400 mA at 24 VDC	
	CPU with all expansion accessories	240 mA at 120 VAC 120 mA at 240 VAC	1200 mA at 24 VDC	
Inrush current (max.)		20 A at 264 VAC	12 A at 28.8 VDC	
Isolation (input power to logic)		1500 VAC	Not isolated	
Ground leakage, AC line to functional earth		0.5 mA max.	--	

Technical data	CPU 1212C AC/DC/Relay	CPU 1212C DC/DC/Relay	CPU 1212C DC/DC/DC
Hold up time (loss of power)	20 ms at 120 VAC 80 ms at 240 VAC	10 ms at 24 VDC	
Internal fuse, not user replaceable	3 A, 250 V, slow blow		

Table A- 35 Sensor power

Technical data	CPU 1212C AC/DC/Relay	CPU 1212C DC/DC/Relay	CPU 1212C DC/DC/DC
Voltage range	20.4 to 28.8 VDC	L+ minus 4 VDC min.	
Output current rating (max.)	300 mA (short-circuit protected		
Maximum ripple noise (<10 MHz)	< 1 V peak to peak	Same as input line	
Isolation (CPU logic to sensor power)	Not isolated		

A.3.3 Digital inputs and outputs

Table A- 36 Digital inputs

Technical data	CPU 1212C AC/DC/Relay, DC/DC/Relay, and DC/DC/DC
Number of inputs	8
Type	Sink/Source (IEC Type 1 sink)
Rated voltage	24 VDC at 4 mA, nominal
Continuous permissible voltage	30 VDC, max.
Surge voltage	35 VDC for 0.5 sec.
Logic 1 signal (min.)	15 VDC at 2.5 mA
Logic 0 signal (max.)	5 VDC at 1 mA
Isolation (field side to logic)	500 VAC for 1 minute
Isolation groups	1
Filter times	us settings: 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 10.0, 12.8, 20.0 ms settings: 0.05, 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 10.0, 12.8, 20.0
HSC clock input rates (max.) (Logic 1 Level = 15 to 26 VDC)	100/80 kHz (Ia.0 to Ia.5) 30 /20 kHz (Ia.6 to Ia.7)
Number of inputs on simultaneously	4 (no adjacent points) at 60 °C horizontal or 50 °C vertical 8 at 55 °C horizontal or 45 °C vertical
Cable length (meters)	500 m shielded, 300 m unshielded, 50 m shielded for HSC inputs

Table A- 37 Digital outputs

Technical data	CPU 1212C AC/DC/Relay and DC/DC/Relay	CPU 1212C DC/DC/DC
Number of outputs	6	
Type	Relay, mechanical	Solid state - MOSFET (sourcing)
Voltage range	5 to 30 VDC or 5 to 250 VAC	20.4 to 28.8 VDC
Logic 1 signal at max. current	--	20 VDC min.
Logic 0 signal with 10 KΩ load	--	0.1 VDC max.
Current (max.)	2.0 A	0.5 A
Lamp load	30 W DC / 200 W AC	5 W
ON state resistance	0.2 Ω max. when new	0.6 Ω max.
Leakage current per point	--	10 μA max.
Surge current	7 A with contacts closed	8 A for 100 ms max.
Overload protection	No	
Isolation (field side to logic)	1500 VAC for 1 minute (coil to contact) None (coil to logic)	500 VAC for 1 minute
Isolation resistance	100 MΩ min. when new	--
Isolation between open contacts	750 VAC for 1 minute	--
Isolation groups	2	1
Inductive clamp voltage	--	L+ minus 48 VDC, 1 W dissipation
Switching delay (Qa.0 to Qa.3)	10 ms max.	1.0 μs max., off to on 3.0 μs max., on to off
Switching delay (Qa.4 to Qa.5)	10 ms max.	50 μs max., off to on 200 μs max., on to off
Maximum relay switching frequency	1 Hz	--
Pulse Train Output rate	Not recommended [1]	100 kHz (Qa.0 to Qa.3)[2], 2 Hz min. 20 kHz (Qa.4 to Qa.5)[2]
Lifetime mechanical (no load)	10,000,000 open/close cycles	--
Lifetime contacts at rated load	100,000 open/close cycles	--
Behavior on RUN to STOP	Last value or substitute value (default value 0)	Last value or substitute value (default value 0)
Number of outputs on simultaneously	3 (no adjacent points) at 60 °C horizontal or 50 °C vertical 6 at 55 °C horizontal, or 45 °C vertical	
Cable length (meters)	500 m shielded, 150 m unshielded	

[1] For CPU models with relay outputs, you must install a digital signal board (SB) to use the pulse outputs.

[2] Depending on your pulse receiver and cable, an additional load resistor (at least 10% of rated current) may improve pulse signal quality and noise immunity.

A.3.4 Analog inputs

Table A- 38 Analog inputs

Technical data	Description
Number of inputs	2
Type	Voltage (single-ended)
Full-scale range	0 to 10 V
Full-scale range (data word)	0 to 27648
Overshoot range	10.001 to 11.759 V
Overshoot range (data word)	27649 to 32511
Overflow range	11.760 to 11.852 V
Overflow range (data word)	32512 to 32767
Resolution	10 bits
Maximum withstand voltage	35 VDC
Smoothing	None, Weak, Medium, or Strong See the table for Step response (ms) for the analog inputs of the CPU (Page 1124).
Noise rejection	10, 50, or 60 Hz
Impedance	≥100 KΩ
Isolation (field side to logic)	None
Accuracy (25 °C / 0 to 55 °C)	3.0% / 3.5% of full-scale
Cable length (meters)	100 m, shielded twisted pair

A.3.4.1 Step response of the built-in analog inputs of the CPU

Table A- 39 Step Response (ms), 0 V to 10 V measured at 95%

Smoothing selection (sample averaging)	Rejection frequency (Integration time)		
	60 Hz	50 Hz	10 Hz
None (1 cycle): No averaging	50 ms	50 ms	100 ms
Weak (4 cycles): 4 samples	60ms	70 ms	200 ms
Medium (16 cycles): 16 samples	200 ms	240 ms	1150 ms
Strong (32 cycles): 32 samples	400 ms	480 ms	2300 ms
Sample time	**4.17 ms**	**5 ms**	**25 ms**

A.3.4.2 Sample time for the built-in analog ports of the CPU

Table A- 40 Sample time for built-in analog inputs of the CPU

Rejection frequency (Integration time selection)	Sample time
60 Hz (16.6 ms)	4.17 ms
50 Hz (20 ms)	5 ms
10 Hz (100 ms)	25 ms

A.3.4.3 Measurement ranges of the analog inputs for voltage (CPUs)

Table A- 41 Analog input representation for voltage (CPUs)

System		**Voltage Measuring Range**	
Decimal	**Hexadecimal**	**0 to 10 V**	
32767	7FFF	11.851 V	Overflow
32512	7F00		
32511	7EFF	11.759 V	Overshoot range
27649	6C01		
27648	6C00	10 V	Rated range
20736	5100	7.5 V	
34	22	12 mV	
0	0	0 V	
Negative values		Negative values are not supported	

A.3.5 CPU 1212C wiring diagrams

Table A- 42 CPU 1212C AC/DC/Relay (6ES7 212-1BE40-0XB0)

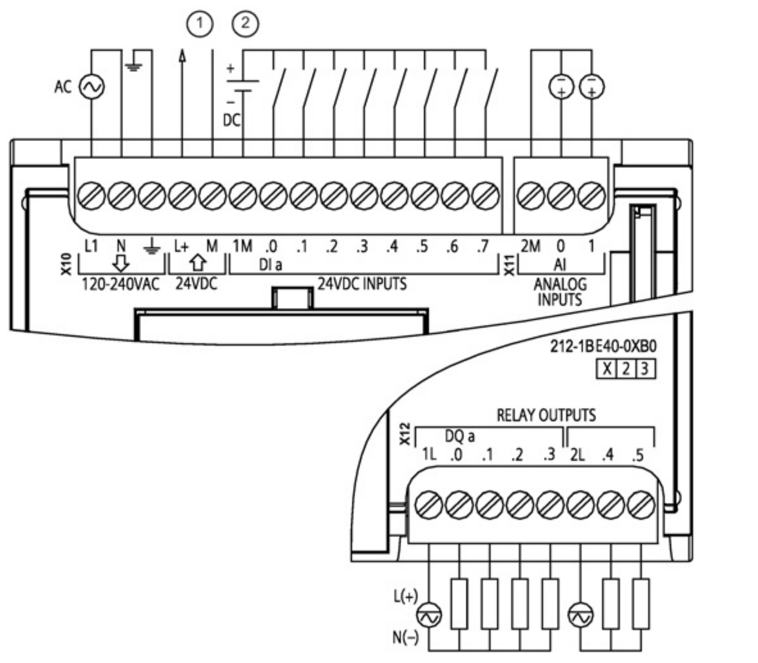
 ① ② AC DC L1 N L+ M 1M .0 .1 .2 .3 .4 .5 .6 .7 DI a X10 120-240VAC 24VDC 24VDC INPUTS X11 2M 0 1 AI ANALOG INPUTS 212-1BE40-0XB0 X 2 3 RELAY OUTPUTS X12 DQ a 1L .0 .1 .2 .3 2L .4 .5 L(+) N(-)	①	24 VDC Sensor Power Out For additional noise immunity, connect "M" to chassis ground even if not using sensor supply.
	②	For sinking inputs, connect "-" to "M" (shown). For sourcing inputs, connect "+" to "M".
	Note 1: X11 connectors must be gold. See Appendix C, Spare Parts for article number.	
	Note 2: Either the L1 or N (L2) terminal can be connected to a voltage source up to 240 VAC. The N terminal can be considered L2 and is not required to be grounded. No polarization is required for L1 and N (L2) terminals.	
	Note 3: See Device Configuration (Page 145) for information about the Ethernet port of the CPU.	

Table A- 43 Connector pin locations for CPU 1212C AC/DC/Relay (6ES7 212-1BE40-0XB0)

Pin	X10	X11 (gold)	X12
1	L1 / 120-240 VAC	2 M	1L
2	N / 120-240 VAC	AI 0	DQ a.0
3	Functional Earth	AI 1	DQ a.1
4	L+ / 24 VDC Sensor Out	--	DQ a.2
5	M / 24 VDC Sensor Out	--	DQ a.3
6	1M	--	2L
7	DI a.0	--	DQ a.4
8	DI a.1	--	DQ a.5
9	DI a.2	--	--
10	DI a.3	--	--
11	DI a.4	--	--
12	DI a.5	--	--

Pin	X10	X11 (gold)	X12
13	DI a.6	--	--
14	DI a.7	--	--

Table A- 44 CPU 1212C DC/DC/Relay (6ES7 212-1HE40-0XB0)

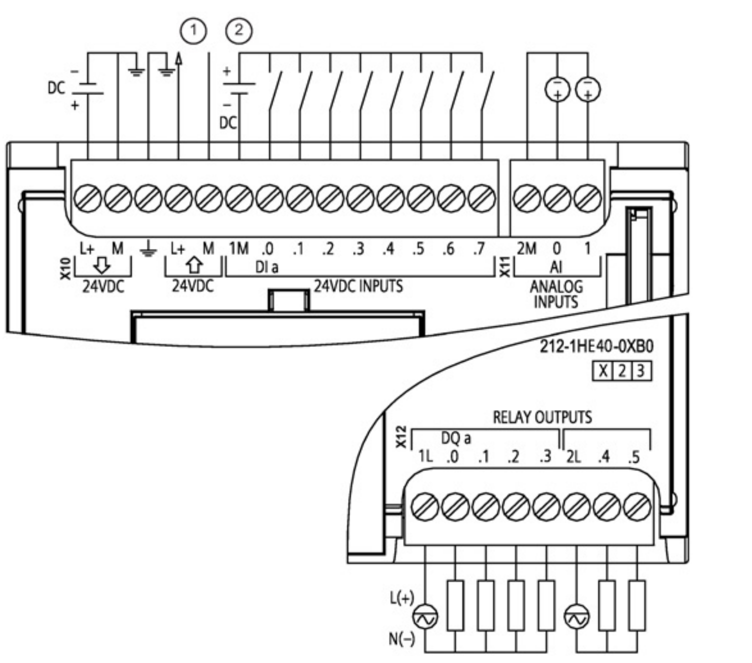
 ① ② DC - + + - DC L+ M ⏚ L+ M 1M .0 .1 .2 .3 .4 .5 .6 .7 2M 0 1 X10 24VDC 24VDC DI a 24VDC INPUTS X11 AI ANALOG INPUTS 212-1HE40-0XB0 X 2 3 RELAY OUTPUTS X12 DQ a 1L .0 .1 .2 .3 2L .4 .5 L(+) N(-)	①	24 VDC Sensor Power Out For additional noise immunity, connect "M" to chassis ground even if not using sensor supply.
	②	For sinking inputs, connect "-" to "M" (shown). For sourcing inputs, connect "+" to "M".
	Note 1: X11 connectors must be gold. See Appendix C, Spare Parts for article number.	
	Note 2: See Device Configuration (Page 145) for information about the Ethernet port of the CPU.	

Table A- 45 Connector pin locations for CPU 1212C DC/DC/Relay (6ES7 212-1HE40-0XB0)

Pin	X10	X11 (gold)	X12
1	L+ / 24 VDC	2 M	1L
2	M / 24 VDC	AI 0	DQ a.0
3	Functional Earth	AI 1	DQ a.1
4	L+ / 24 VDC Sensor Out	--	DQ a.2
5	M / 24 VDC Sensor Out	--	DQ a.3
6	1M	--	2L
7	DI a.0	--	DQ a.4
8	DI a.1	--	DQ a.5
9	DI a.2	--	--
10	DI a.3	--	--
11	DI a.4	--	--
12	DI a.5	--	--
13	DI a.6	--	--
14	DI a.7	--	--

Table A- 46 CPU 1212C DC/DC/DC (6ES7-212-1AE40-0XB0)

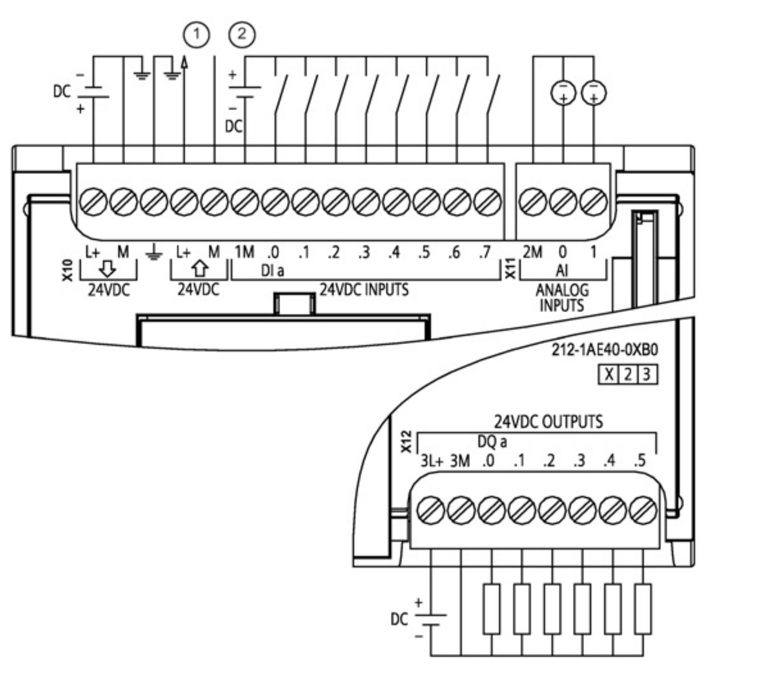
 ① ② DC DC X10 L+ M ⏚ L+ M 1M .0 .1 .2 .3 .4 .5 .6 .7 24VDC 24VDC DI a 24VDC INPUTS X11 2M 0 1 AI ANALOG INPUTS 212-1AE40-0XB0 X 2 3 24VDC OUTPUTS X12 DQ a 3L+ 3M .0 .1 .2 .3 .4 .5 DC	①	24 VDC Sensor Power Out For additional noise immunity, connect "M" to chassis ground even if not using sensor supply.
	②	For sinking inputs, connect "-" to "M" (shown). For sourcing inputs, connect "+" to "M".
	Note 1: X11 connectors must be gold. See Appendix C, Spare Parts for article number.	
	Note 2: See Device Configuration (Page 145) for information about the Ethernet port of the CPU.	

Table A- 47 Connector pin locations for CPU 1212C DC/DC/DC (6ES7 212-1AE40-0XB0)

Pin	X10	X11 (gold)	X12
1	L+ / 24 VDC	2 M	3L+
2	M / 24 VDC	AI 0	3M
3	Functional Earth	AI 1	DQ a.0
4	L+ / 24 VDC Sensor Out	--	DQ a.1
5	M / 24 VDC Sensor Out	--	DQ a.2
6	1M	--	DQ a.3
7	DI a.0	--	DQ a.4
8	DI a.1	--	DQ a.5
9	DI a.2	--	--
10	DI a.3	--	--
11	DI a.4	--	--
12	DI a.5	--	--
13	DI a.6	--	--
14	DI a.7	--	--

Note

Unused analog inputs should be shorted.

A.4 CPU 1214C

A.4.1 General specifications and features

Table A- 48 General

Technical data	**CPU 1214C AC/DC/Relay**	**CPU 1214C DC/DC/Relay**	**CPU 1214C DC/DC/DC**
Article number	6ES7 214-1BG40-0XB0	6ES7 214-1HG40-0XB0	6ES7 214-1AG40-0XB0
Dimensions W x H x D (mm)	110 x 100 x 75		
Shipping weight	475 grams	435 grams	415 grams
Power dissipation	14 W	12 W	
Current available (SM and CM bus)	1600 mA max. (5 VDC)		
Current available (24 VDC)	400 mA max. (sensor power)		
Digital input current consumption (24 VDC)	4 mA/input used		

Table A- 49 CPU features

Technical data		**Description**
User memory (Refer to "General technical specifications", (Page 1099) "Internal CPU memory retention".)	Work	100 Kbytes
	Load	4 Mbytes internal, expandable up to SD card size
	Retentive	10 Kbytes
On-board digital I/O		14 inputs/10 outputs
On-board analog I/O		2 inputs
Process image size		1024 bytes of inputs (I)/1024 bytes of outputs (Q)
Bit memory (M)		8192 bytes
Temporary (local) memory		• 16 Kbytes for startup and program cycle (including associated FBs and FCs) • 6 Kbytes for each of the other interrupt priority levels (including FBs and FCs)
Signal modules expansion		8 SMs max.
SB, CB, BB expansion		1 max.
Communication module expansion		3 CMs max.
High-speed counters		Up to 6 configured to use any built-in or SB inputs. See table, CPU1214C: HSC default address assignments (Page 463) • 100/[1]80 kHz (Ia.0 to Ia.5) • 30/[1]20 kHz (Ia.6 to Ib.5)

Technical data	Description
Pulse outputs[2]	Up to 4 configured to use any built-in or SB outputs • 100 kHz (Qa.0 to Qa.3) • 30 kHz (Qa.4 to Qb.1)
Pulse catch inputs	14
Time delay interrupts	4 total with 1 ms resolution
Cyclic interrupts	4 total with 1 ms resolution
Edge interrupts	12 rising and 12 falling (16 and 16 with optional signal board)
Memory card	SIMATIC Memory Card (optional)
Real time clock accuracy	+/- 60 seconds/month
Real time clock retention time	20 days typ./12 days min. at 40 °C (maintenance-free Super Capacitor)

[1] The slower speed is applicable when the HSC is configured for quadrature mode of operation.

[2] For CPU models with relay outputs, you must install a digital signal board (SB) to use the pulse outputs.

Table A- 50 Performance

Type of instruction	Execution speed
Boolean	0.08 μs/instruction
Move Word	1.7 μs/instruction
Real math	2.3 μs/instruction

A.4.2 Timers, counters and code blocks supported by CPU 1214C

Table A- 51 Blocks, timers and counters supported by CPU 1214C

Element		Description
Blocks	Type	OB, FB, FC, DB
	Size	64 Kbytes
	Quantity	Up to 1024 blocks total (OBs + FBs + FCs + DBs)
	Address range for FBs, FCs, and DBs	FB and FC: 1 to 65535 (such as FB 1 to FB 65535) DB: 1 to 59999
	Nesting depth	16 from the program cycle or startup OB 6 from any interrupt event OB
	Monitoring	Status of 2 code blocks can be monitored simultaneously
OBs	Program cycle	Multiple
	Startup	Multiple
	Time-delay interrupts	4 (1 per event)
	Cyclic interrupts	4 (1 per event)
	Hardware interrupts	50 (1 per event)
	Time error interrupts	1
	Diagnostic error interrupts	1
	Pull or plug of modules	1

Element		Description
	Rack or station failure	1
	Time of day	Multiple
	Status	1
	Update	1
	Profile	1
Timers	Type	IEC
	Quantity	Limited only by memory size
	Storage	Structure in DB, 16 bytes per timer
Counters	Type	IEC
	Quantity	Limited only by memory size
	Storage	Structure in DB, size dependent upon count type • SInt, USInt: 3 bytes • Int, UInt: 6 bytes • DInt, UDInt: 12 bytes

Table A- 52 Communication

Technical data	Description
Number of ports	1
Type	Ethernet
HMI device	4
Programming device (PG)	1
Connections	• 8 for Open User Communication (active or passive): TSEND_C, TRCV_C, TCON, TDISCON, TSEND, and TRCV • 3 for server GET/PUT (CPU-to-CPU) S7 communication • 8 for client GET/PUT (CPU-to-CPU) S7 communication
Data rates	10/100 Mb/s
Isolation (external signal to PLC logic)	Transformer isolated, 1500 VAC, for short term event safety only
Cable type	CAT5e shielded

Table A- 53 Power supply

Technical data		CPU 1214C AC/DC/Relay	CPU 1214C DC/DC/Relay	CPU 1214C DC/DC/DC
Voltage range		85 to 264 VAC	20.4 VDC to 28.8 VDC	
Line frequency		47 to 63 Hz	--	
Input current (max. load)	CPU only	100 mA at 120 VAC 50 mA at 240 VAC	500 mA at 24 VDC	
	CPU with all expansion accessories	300 mA at 120 VAC 150 mA at 240 VAC	1500 mA at 24 VDC	
Inrush current (max.)		20 A at 264 VAC	12 A at 28.8 VDC	
Isolation (input power to logic)		1500 VAC	Not isolated	
Ground leakage, AC line to functional earth		0.5 mA max.	-	
Hold up time (loss of power)		20 ms at 120 VAC 80 ms at 240 VAC	10 ms at 24 VDC	
Internal fuse, not user replaceable		3 A, 250 V, slow blow		

Table A- 54 Sensor power

Technical data	CPU 1214C AC/DC/Relay	CPU 1214C DC/DC/Relay	CPU 1214C DC/DC/DC
Voltage range	20.4 to 28.8 VDC	L+ minus 4 VDC min.	
Output current rating (max.)	400 mA (short-circuit protected)		
Maximum ripple noise (<10 MHz)	< 1 V peak to peak	Same as input line	
Isolation (CPU logic to sensor power)	Not isolated		

A.4.3 Digital inputs and outputs

Table A- 55 Digital inputs

Technical data	CPU 1214C AC/DC/Relay	CPU 1214C DC/DC/Relay	CPU 1214C DC/DC/DC
Number of inputs	14		
Type	Sink/Source (IEC Type 1 sink)		
Rated voltage	24 VDC at 4 mA, nominal		
Continuous permissible voltage	30 VDC, max.		
Surge voltage	35 VDC for 0.5 sec.		
Logic 1 signal (min.)	15 VDC at 2.5 mA		
Logic 0 signal (max.)	5 VDC at 1 mA		
Isolation (field side to logic)	500 VAC for 1 minute		
Isolation groups	1		
Filter times	us settings: 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 10.0, 12.8, 20.0 ms settings: 0.05, 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 10.0, 12.8, 20.0		
HSC clock input rates (max.) (Logic 1 Level = 15 to 26 VDC)	100/80 kHz (Ia.0 to Ia.5) 30/20 kHz (Ia.6 to Ib.5		
Number of inputs on simultaneously	• 7 (no adjacent points) at 60 °C horizontal or 50 °C vertical • 14 at 55 °C horizontal or 45 °C vertical		
Cable length (meters)	500 m shielded, 300 m unshielded, 50 m shielded for HSC inputs		

Table A- 56 Digital outputs

Technical data	CPU 1214C AC/DC/Relay and DC/DC/Relay	CPU 1214C DC/DC/DC
Number of outputs	10	
Type	Relay, mechanical	Solid state - MOSFET (sourcing)
Voltage range	5 to 30 VDC or 5 to 250 VAC	20.4 to 28.8 VDC
Logic 1 signal at max. current	--	20 VDC min.
Logic 0 signal with 10 KΩ load	--	0.1 VDC max.
Current (max.)	2.0 A	0.5 A
Lamp load	30 W DC / 200 W AC	5 W
ON state resistance	0.2 Ω max. when new	0.6 Ω max.
Leakage current per point	--	10 μA max.
Surge current	7 A with contacts closed	8 A for 100 ms max.
Overload protection	No	
Isolation (field side to logic)	1500 VAC for 1 minute (coil to contact) None (coil to logic)	500 VAC for 1 minute
Isolation resistance	100 MΩ min. when new	--
Isolation between open contacts	750 VAC for 1 minute	--
Isolation groups	2	1

Technical data	**CPU 1214C AC/DC/Relay and DC/DC/Relay**	**CPU 1214C DC/DC/DC**
Isolation (group-to-group)	1500 VAC[1]	--
Inductive clamp voltage	--	L+ minus 48 VDC, 1 W dissipation
Switching delay (Qa.0 to Qa.3)	10 ms max.	1.0 µs max., off to on 3.0 µs max., on to off
Switching delay (Qa.4 to Qb.1)	10 ms max.	50 µs max., off to on 200 µs max., on to off
Maximum relay switching frequency	1 Hz	--
Pulse Train Output rate	Not recommended [2]	100 kHz (Qa.0 to Qa.3)[3], 2 Hz min. 20 kHz (Qa.4 to Qb.1)[3]
Lifetime mechanical (no load)	10,000,000 open/close cycles	--
Lifetime contacts at rated load	100,000 open/close cycles	--
Behavior on RUN to STOP	Last value or substitute value (default value 0)	
Number of outputs on simultaneously	• 5 (no adjacent points) at 60 °C horizontal or 50 °C vertical • 10 at 55 °C horizontal or 45 °C vertical	
Cable length (meters)	500 m shielded, 150 m unshielded	

[1] Relay group-to-group isolation separates line voltage from SELVPELV and separates different phases up to 250 VAC line to ground.

[2] For CPU models with relay outputs, you must install a digital signal board (SB) to use the pulse outputs.

[3] Depending on your pulse receiver and cable, an additional load resistor (at least 10% of rated current) may improve pulse signal quality and noise immunity.

A.4.4 Analog inputs

Table A- 57 Analog inputs

Technical data	Description
Number of inputs	2
Type	Voltage (single-ended)
Full-scale range	0 to 10 V
Full-scale range (data word)	0 to 27648
Overshoot range	10.001 to 11.759 V
Overshoot range (data word)	27649 to 32511
Overflow range	11.760 to 11.852 V
Overflow range (data word)	32512 to 32767
Resolution	10 bits
Maximum withstand voltage	35 VDC
Smoothing	None, Weak, Medium, or Strong See the table for Step response (ms) for the analog inputs of the CPU (Page 1135).
Noise rejection	10, 50, or 60 Hz
Impedance	≥100 KΩ
Isolation (field side to logic)	None
Accuracy (25 °C / 0 to 55 °C)	3.0% / 3.5% of full-scale
Cable length (meters)	100 m, shielded twisted pair

A.4.4.1 Step response of the built-in analog inputs of the CPU

Table A- 58 Step Response (ms), 0 V to 10 V measured at 95%

Smoothing selection (sample averaging)	Rejection frequency (Integration time)		
	60 Hz	50 Hz	10 Hz
None (1 cycle): No averaging	50 ms	50 ms	100 ms
Weak (4 cycles): 4 samples	60ms	70 ms	200 ms
Medium (16 cycles): 16 samples	200 ms	240 ms	1150 ms
Strong (32 cycles): 32 samples	400 ms	480 ms	2300 ms
Sample time	**4.17 ms**	**5 ms**	**25 ms**

A.4.4.2 Sample time for the built-in analog ports of the CPU

Table A- 59 Sample time for built-in analog inputs of the CPU

Rejection frequency (Integration time selection)	Sample time
60 Hz (16.6 ms)	4.17 ms
50 Hz (20 ms)	5 ms
10 Hz (100 ms)	25 ms

A.4.4.3 Measurement ranges of the analog inputs for voltage (CPUs)

Table A- 60 Analog input representation for voltage (CPUs)

System		Voltage Measuring Range	
Decimal	**Hexadecimal**	**0 to 10 V**	
32767	7FFF	11.851 V	Overflow
32512	7F00		
32511	7EFF	11.759 V	Overshoot range
27649	6C01		
27648	6C00	10 V	Rated range
20736	5100	7.5 V	
34	22	12 mV	
0	0	0 V	
Negative values		Negative values are not supported	

A.4.5 CPU 1214C wiring diagrams

Table A- 61 CPU 1214C AC/DC/Relay (6ES7 214-1BG40-0XB0)

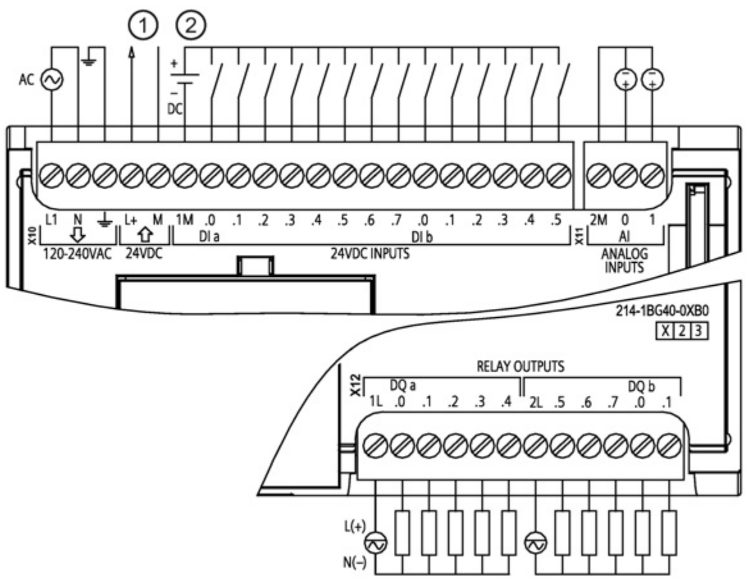
	①	24 VDC Sensor Power Out For additional noise immunity, connect "M" to chassis ground even if not using sensor supply.
	②	For sinking inputs, connect "-" to "M" (shown). For sourcing inputs, connect "+" to "M".
	Note 1: X11 connectors must be gold. See Appendix C, Spare Parts for article number.	
	Note 2: Either the L1 or N (L2) terminal can be connected to a voltage source up to 240 VAC. The N terminal can be considered L2 and is not required to be grounded. No polarization is required for L1 and N (L2) terminals.	
	Note 3: See Device Configuratio (Page 145)n for information about the Ethernet port of the CPU.	

Table A- 62 Connector pin locations for CPU 1214C AC/DC/Relay (6ES7 214-1BG40-0XB0)

Pin	X10	X11 (gold)	X12
1	L1 / 120-240 VAC	2 M	1L
2	N / 120-240 VAC	AI 0	DQ a.0
3	Functional Earth	AI 1	DQ a.1
4	L+ / 24 VDC Sensor Out	--	DQ a.2
5	M / 24 VDC Sensor Out	--	DQ a.3
6	1M	--	DQ a.4
7	DI a.0	--	2L
8	DI a.1	--	DQ a.5
9	DI a.2	--	DQ a.6
10	DI a.3	--	DQ a.7
11	DI a.4	--	DQ b.0
12	DI a.5	--	DQ b.1
13	DI a.6	--	--

Pin	X10	X11 (gold)	X12
14	DI a.7	--	--
15	DI b.0	--	--
16	DI b.1	--	--
17	DI b.2	--	--
18	DI b.3	--	--
19	DI b.4	--	--
20	DI b.5	--	--

Table A- 63 CPU 1214C DC/DC/Relay (6ES7 214-1HG40-0XB0)

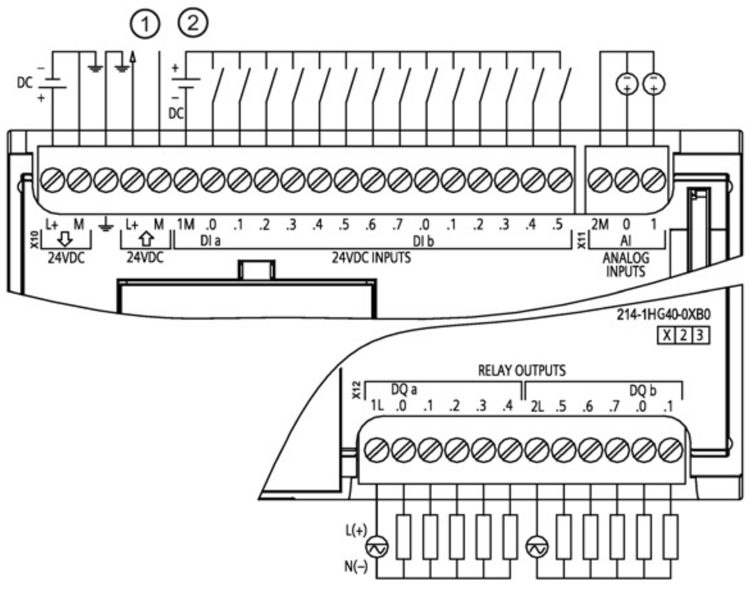
 ① ② DC DC L+ M L+ M 1M .0 .1 .2 .3 .4 .5 .6 .7 .0 .1 .2 .3 .4 .5 X10 24VDC 24VDC DI a DI b 24VDC INPUTS 2M 0 1 X11 AI ANALOG INPUTS 214-1HG40-0XB0 X 2 3 RELAY OUTPUTS X12 DQ a DQ b 1L .0 .1 .2 .3 .4 2L .5 .6 .7 .0 .1 L(+) N(-)	①	24 VDC Sensor Power Out For additional noise immunity, connect "M" to chassis ground even if not using sensor supply.
	②	For sinking inputs, connect "-" to "M" (shown). For sourcing inputs, connect "+" to "M".
	Note 1: X11 connectors must be gold. See Appendix C, Spare Parts for article number.	
	Note 2: See Device Configuration (Page 145) for information about the Ethernet port of the CPU.	

Table A- 64 Connector pin locations for CPU 1214C DC/DC/Relay (6ES7 214-1HG40-0XB0)

Pin	X10	X11 (gold)	X12
1	L+ / 24 VDC	2 M	1L
2	M / 24 VDC	AI 0	DQ a.0
3	Functional Earth	AI 1	DQ a.1
4	L+ / 24 VDC Sensor Out	--	DQ a.2
5	M / 24 VDC Sensor Out	--	DQ a.3
6	1M	--	DQ a.4
7	DI a.0	--	2L
8	DI a.1	--	DQ a.5
9	DI a.2	--	DQ a.6
10	DI a.3	--	DQ a.7
11	DI a.4	--	DQ b.0

Pin	X10	X11 (gold)	X12
12	DI a.5	--	DQ b.1
13	DI a.6	--	--
14	DI a.7	--	--
15	DI b.0	--	--
16	DI b.1	--	--
17	DI b.2	--	--
18	DI b.3	--	--
19	DI b.4	--	--
20	DI b.5	--	--

Table A- 65 CPU 1214C DC/DC/DC (6ES7 214-1AG40-0XB0)

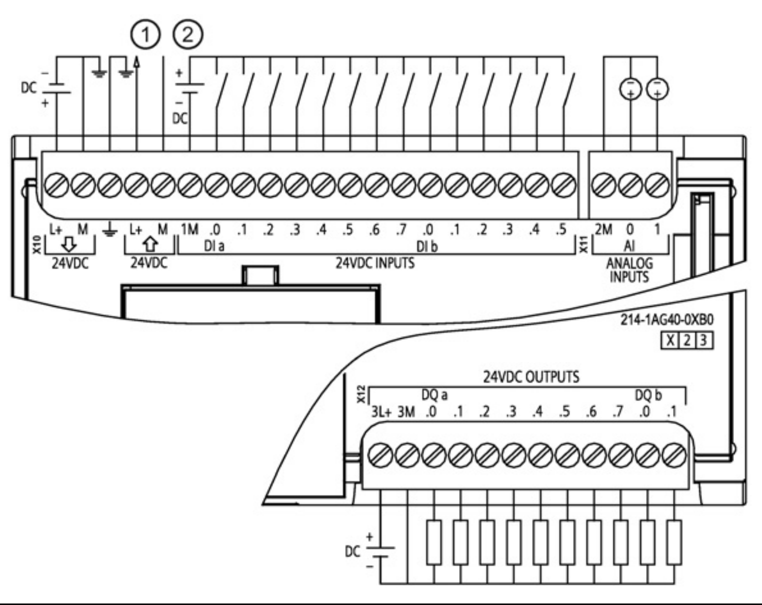
 ① ② DC X10 L+ M ⏚ L+ M 1M .0 .1 .2 .3 .4 .5 .6 .7 .0 .1 .2 .3 .4 .5 24VDC 24VDC DI a DI b 24VDC INPUTS X11 2M 0 1 AI ANALOG INPUTS 214-1AG40-0XB0 X 2 3 24VDC OUTPUTS X12 DQ a DQ b 3L+ 3M .0 .1 .2 .3 .4 .5 .6 .7 .0 .1	①	24 VDC Sensor Power Out For additional noise immunity, connect "M" to chassis ground even if not using sensor supply.
	②	For sinking inputs, connect "-" to "M" (shown). For sourcing inputs, connect "+" to "M".
	Note 1: X11 connectors must be gold. See Appendix C, Spare Parts for article number.	
	Note 2: See Device Configuration (Page 145) for information about the Ethernet port of the CPU.	

Table A- 66 Connector pin locations for CPU 1214C DC/DC/DC (6ES7 214-1AG40-0XB0)

Pin	X10	X11 (gold)	X12
1	L+ / 24 VDC	2 M	3L+
2	M / 24 VDC	AI 0	3M
3	Functional Earth	AI 1	DQ a.0
4	L+ / 24 VDC Sensor Out	--	DQ a.1
5	M / 24 VDC Sensor Out	--	DQ a.2
6	1M	--	DQ a.3
7	DI a.0	--	DQ a.4
8	DI a.1	--	DQ a.5
9	DI a.2	--	DQ a.6

Pin	X10	X11 (gold)	X12
10	DI a.3	--	DQ a.7
11	DI a.4	--	DQ b.0
12	DI a.5	--	DQ b.1
13	DI a.6	--	--
14	DI a.7	--	-
15	DI b.0	--	--
16	DI b.1	--	--
17	DI b.2	--	--
18	DI b.3	--	--
19	DI b.4	--	--
20	DI b.5	--	--

Note

Unused analog inputs should be shorted.

A.5 CPU 1215C

A.5.1 General specifications and features

Table A- 67 General

Technical data	CPU 1215C AC/DC/Relay	CPU 1215C DC/DC/Relay	CPU 1215C DC/DC/DC
Article number	6ES7 215-1BG40-0XB0	6ES7 215-1HG40-0XB0	6ES7 215-1AG40-0XB0
Dimensions W x H x D (mm)	130 x 100 x 75		
Shipping weight	585 grams	550 grams	520 grams
Power dissipation	14 W	12 W	
Current available (SM and CM bus)	1600 mA max. (5 VDC)		
Current available (24 VDC)	400 mA max. (sensor power)		
Digital input current consumption (24 VDC)	4 mA/input used		

Table A- 68 CPU features

Technical data		Description
User memory (Refer to "General technical specifications (Page 1099)", "Internal CPU memory retention".)	Work	125 Kbytes
	Load	4 Mbytes, internal, expandable up to SD card size
	Retentive	10 Kbytes
On-board digital I/O		14 inputs/10 outputs
On-board analog I/O		2 inputs/2 outputs
Process image size		1024 bytes of inputs (I)/1024 bytes of outputs (Q)
Bit memory (M)		8192 bytes
Temporary (local) memory		• 16 Kbytes for startup and program cycle (including associated FBs and FCs) • 6 Kbytes for each of the other interrupt priority levels (including FBs and FCs)
Signal modules expansion		8 SMs max.
SB, CB, BB expansion		1 max.
Communication module expansion		3 CMs max.
High-speed counters		Up to 6 configured to use any built-in or SB inputs. See table CPU 1215C: HSC default address assignments (Page 463) • 100/^{1}80 kHz (Ia.0 to Ia.5) • 30/^{1}20 kHz (Ia.6 to Ib.5)

Technical data	Description
Pulse outputs[2]	Up to 4 configured to use any built-in or SB outputs • 100 kHz (Qa.0 to Qa.3) • 30 kHz (Qa.4 to Qb.1)
Pulse catch inputs	14
Time delay interrupts	4 total with 1 ms resolution
Cyclic interrupts	4 total with 1 ms resolution
Edge interrupts	12 rising and 12 falling (16 and 16 with optional signal board)
Memory card	SIMATIC Memory Card (optional)
Real time clock accuracy	+/- 60 seconds/month
Real time clock retention time	20 days typ./12 days min. at 40 °C (maintenance-free Super Capacitor)

1 The slower speed is applicable when the HSC is configured for quadrature mode of operation.

2 For CPU models with relay outputs, you must install a digital signal board (SB) to use the pulse outputs.

Table A- 69 Performance

Type of instruction	Execution speed
Boolean	0.08 μs/instruction
Move Word	1.7 μs/instruction
Real math	2.3 μs/instruction

A.5.2 Timers, counters and code blocks supported by CPU 1215C

Table A- 70 Blocks, timers and counters supported by CPU 1215C

Element		Description
Blocks	Type	OB, FB, FC, DB
	Size	64 Kbytes
	Quantity	Up to 1024 blocks total (OBs + FBs + FCs + DBs)
	Address range for FBs, FCs, and DBs	FB and FC: 1 to 65535 (such as FB 1 to FB 65535) DB: 1 to 59999
	Nesting depth	16 from the program cycle or startup OB 6 from any interrupt event OB
	Monitoring	Status of 2 code blocks can be monitored simultaneously
OBs	Program cycle	Multiple
	Startup	Multiple
	Time-delay interrupts	4 (1 per event)
	Cyclic interrupts	4 (1 per event)
	Hardware interrupts	50 (1 per event)
	Time error interrupts	1
	Diagnostic error interrupts	1
	Pull or plug of modules	1

Element		Description
	Rack or station failure	1
	Time of day	Multiple
	Status	1
	Update	1
	Profile	1
Timers	Type	IEC
	Quantity	Limited only by memory size
	Storage	Structure in DB, 16 bytes per timer
Counters	Type	IEC
	Quantity	Limited only by memory size
	Storage	Structure in DB, size dependent upon count type • SInt, USInt: 3 bytes • Int, UInt: 6 bytes • DInt, UDInt: 12 bytes

Table A- 71 Communication

Technical data	Description
Number of ports	2
Type	Ethernet
HMI device	4
Programming device (PG)	1
Connections	• 8 for Open User Communication (active or passive): TSEND_C, TRCV_C, TCON, TDISCON, TSEND, and TRCV • 3 for server GET/PUT (CPU-to-CPU) S7 communication • 8 for client GET/PUT (CPU-to-CPU) S7 communication
Data rates	10/100 Mb/s
Isolation (external signal to PLC logic)	Transformer isolated, 1500 VAC, for short term event safety only
Cable type	CAT5e shielded

Table A- 72 Power supply

Technical data		CPU 1215C AC/DC/Relay	CPU 1215C DC/DC/Relay	CPU 1215C DC/DC/DC
Voltage range		85 to 264 VAC	20.4 VDC to 28.8 VDC	
Line frequency		47 to 63 Hz	--	
Input current (max. load)	CPU only	100 mA at 120 VAC 50 mA at 240 VAC	500 mA at 24 VDC	
	CPU with all expansion accessories	300 mA at 120 VAC 150 mA at 240 VAC	1500 mA at 24 VDC	
Inrush current (max.)		20 A at 264 VAC	12 A at 28.8 VDC	
Isolation (input power to logic)		1500 VAC	Not isolated	
Ground leakage, AC line to functional earth		0.5 mA max.	-	
Hold up time (loss of power)		20 ms at 120 VAC 80 ms at 240 VAC	10 ms at 24 VDC	
Internal fuse, not user replaceable		3 A, 250 V, slow blow		

Table A- 73 Sensor power

Technical data	CPU 1215C AC/DC/Relay	CPU 1215C DC/DC/Relay	CPU 1215C DC/DC/DC
Voltage range	20.4 to 28.8 VDC	L+ minus 4 VDC min.	
Output current rating (max.)	400 mA (short-circuit protected)		
Maximum ripple noise (<10 MHz)	< 1 V peak to peak	Same as input line	
Isolation (CPU logic to sensor power)	Not isolated		

A.5.3 Digital inputs and outputs

Table A- 74 Digital inputs

Technical data	CPU 1215C AC/DC/Relay	CPU 1215C DC/DC/Relay	CPU 1215C DC/DC/DC
Number of inputs	14		
Type	Sink/Source (IEC Type 1 sink)		
Rated voltage	24 VDC at 4 mA, nominal		
Continuous permissible voltage	30 VDC, max.		
Surge voltage	35 VDC for 0.5 sec.		
Logic 1 signal (min.)	15 VDC at 2.5 mA		
Logic 0 signal (max.)	5 VDC at 1 mA		
Isolation (field side to logic)	500 VAC for 1 minute		
Isolation groups	1		
Filter times	us settings: 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 10.0, 12.8, 20.0 ms settings: 0.05, 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 10.0, 12.8, 20.0		
HSC clock input rates (max.) (Logic 1 Level = 15 to 26 VDC)	100/80 kHz (Ia.0 to Ia.5) 30/20 kHz (Ia.6 to Ib.5)		
Number of inputs on simultaneously	• 7 (no adjacent points) at 60 °C horizontal or 50 °C vertical • 14 at 55 °C horizontal or 45 °C vertical		
Cable length (meters)	500 m shielded, 300 m unshielded, 50 m shielded for HSC inputs		

Table A- 75 Digital outputs

Technical data	CPU 1215C AC/DC/Relay and CPU 1215C DC/DC/Relay	CPU 1215C DC/DC/DC
Number of outputs	10	
Type	Relay, mechanical	Solid state - MOSFET (sourcing)
Voltage range	5 to 30 VDC or 5 to 250 VAC	20.4 to 28.8 VDC
Logic 1 signal at max. current	--	20 VDC min.
Logic 0 signal with 10 KΩ load	--	0.1 VDC max.
Current (max.)	2.0 A	0.5 A
Lamp load	30 W DC / 200 W AC	5 W
ON state resistance	0.2 Ω max. when new	0.6 Ω max.
Leakage current per point	--	10 μA max.
Surge current	7 A with contacts closed	8 A for 100 ms max.
Overload protection	No	
Isolation (field side to logic)	1500 VAC for 1 minute (coil to contact) None (coil to logic)	500 VAC for 1 minute
Isolation resistance	100 MΩ min. when new	--
Isolation between open contacts	750 VAC for 1 minute	--
Isolation groups	2	1

Technical data	CPU 1215C AC/DC/Relay and CPU 1215C DC/DC/Relay	CPU 1215C DC/DC/DC
Isolation (group-to-group)	1500 VAC[1]	--
Inductive clamp voltage	--	L+ minus 48 VDC, 1 W dissipation
Switching delay (Qa.0 to Qa.3)	10 ms max.	1.0 µs max., off to on 3.0 µs max., on to off
Switching delay (Qa.4 to Qb.1)	10 ms max.	50 µs max., off to on 200 µs max., on to off
Maximum relay switching frequency	1 Hz	--
Pulse Train Output rate	Not recommended [2]	100 kHz (Qa.0 to Qa.3)[3], 2 Hz min. 20 kHz (Qa.4 to Qb.1)[3]
Lifetime mechanical (no load)	10,000,000 open/close cycles	--
Lifetime contacts at rated load	100,000 open/close cycles	--
Behavior on RUN to STOP	Last value or substitute value (default value 0)	
Number of outputs on simultaneously	• 5 (no adjacent points) at 60 °C horizontal or 50 °C vertical • 10 at 55 °C horizontal or 45 °C vertical	
Cable length (meters)	500 m shielded, 150 m unshielded	

[1] Relay group-to-group isolation separates line voltage from SELV/PELV and separates different phases up to 250 VAC line to ground.

[2] For CPU models with relay outputs, you must install a digital signal board (SB) to use the pulse outputs.

[3] Depending on your pulse receiver and cable, an additional load resistor (at least 10% of rated current) may improve pulse signal quality and noise immunity.

A.5.4 Analog inputs and outputs

Table A- 76 Analog inputs

Technical data	Description
Number of inputs	2
Type	Voltage (single-ended)
Full-scale range	0 to 10 V
Full-scale range (data word)	0 to 27648
Overshoot range	10.001 to 11.759 V
Overshoot range (data word)	27649 to 32511
Overflow range	11.760 to 11.852 V
Overflow range (data word)	32512 to 32767
Resolution	10 bits
Maximum withstand voltage	35 VDC
Smoothing	None, Weak, Medium, or Strong See the table for Step response (ms) for the analog inputs of the CPU (Page 1147).
Noise rejection	10, 50, or 60 Hz

Technical data	Description
Impedance	≥100 KΩ
Isolation (field side to logic)	None
Accuracy (25 °C / 0 to 55 °C)	3.0% / 3.5% of full-scale
Cable length (meters)	100 m, shielded twisted pair

A.5.4.1 Step response of built-in analog inputs of the CPU

Table A- 77 Step Response (ms), 0 V to 10 V measured at 95%

Smoothing selection (sample averaging)	Rejection frequency (Integration time)		
	60 Hz	50 Hz	10 Hz
None (1 cycle): No averaging	50 ms	50 ms	100 ms
Weak (4 cycles): 4 samples	60ms	70 ms	200 ms
Medium (16 cycles): 16 samples	200 ms	240 ms	1150 ms
Strong (32 cycles): 32 samples	400 ms	480 ms	2300 ms
Sample time	**4.17 ms**	**5 ms**	**25 ms**

A.5.4.2 Sample time for the built-in analog ports of the CPU

Table A- 78 Sample time for built-in analog inputs of the CPU

Rejection frequency (Integration time selection)	Sample time
60 Hz (16.6 ms)	4.17 ms
50 Hz (20 ms)	5 ms
10 Hz (100 ms)	25 ms

A.5.4.3 Measurement ranges of the analog inputs for voltage (CPUs)

Table A- 79 Analog input representation for voltage (CPUs)

System		**Voltage Measuring Range**	
Decimal	**Hexadecimal**	**0 to 10 V**	
32767	7FFF	11.851 V	Overflow
32512	7F00		
32511	7EFF	11.759 V	Overshoot range
27649	6C01		
27648	6C00	10 V	Rated range
20736	5100	7.5 V	
34	22	12 mV	
0	0	0 V	
Negative values		Negative values are not supported	

A.5.4.4 Analog output specifications

Table A- 80 Analog outputs

Technical data	**Description**
Number of outputs	2
Type	Current
Full-scale range	0 to 20 mA
Full-scale range (data word)	0 to 27648
Overshoot range	20.01 to 23.52 mA
Overshoot range (data word)	27649 to 32511
Overflow range	see footnote [1]
Overflow range data word	32512 to 32767
Resolution	10 bits
Output drive impedance	≤500 Ω max.
Isolation (field side to logic)	None
Accuracy (25 °C / -20 to 60 °C)	3.0% / 3.5% of full-scale
Settling time	2 ms
Cable length (meters)	100 m, shielded twisted pair

[1] In an overflow condition, analog outputs will behave according to the device configuration properties settings. In the "Reaction to CPU STOP" parameter, select either: "Use substitute value" or "Keep last value".

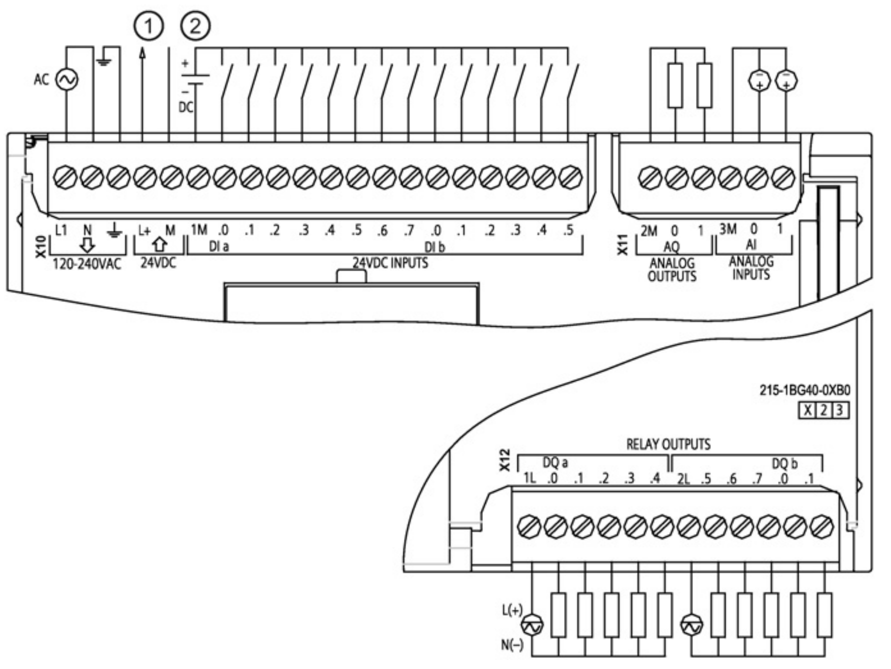
Table A- 81 Analog output representation for current (CPU 1215C and CPU 1217C)

System		Current output range	
Decimal	**Hexadecimal**	**0 mA to 20 mA**	
32767	7FFF	See note 1	Overflow
32512	7F00	See note 1	
32511	7EFF	23.52 mA	Overshoot range
27649	6C01		
27648	6C00	20 mA	Rated range
20736	5100	15 mA	
34	22	0.0247 mA	
0	0	0 mA	
Negative values		Negative values are not supported	

1 In an overflow condition, analog outputs will behave according to the device configuration properties settings. In the "Reaction to CPU STOP" parameter, select either: "Use substitute value" or "Keep last value".

A.5.5 CPU 1215C wiring diagrams

Table A- 82 CPU 1215C AC/DC/Relay (6ES7 215-1BG40-0XB0)



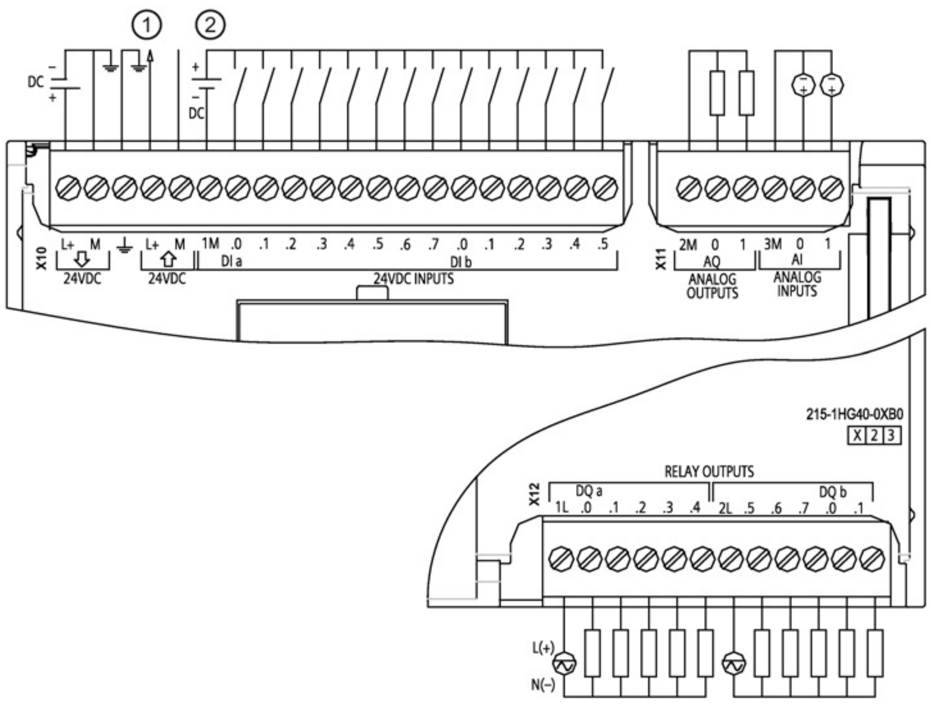
①	24 VDC Sensor Power Out For additional noise immunity, connect "M" to chassis ground even if not using sensor supply.
②	For sinking inputs, connect "-" to "M" (shown). For sourcing inputs, connect "+" to "M".
Note 1: X11 connectors must be gold. See Appendix C, Spare Parts for article number.	
Note 2: Either the L1 or N (L2) terminal can be connected to a voltage source up to 240 VAC. The N terminal can be considered L2 and is not required to be grounded. No polarization is required for L1 and N (L2) terminals.	
Note 3: See Device Configuration (Page 145) for information about the Ethernet port of the CPU.	

Table A- 83 Connector pin locations for CPU 1215C AC/DC/Relay (6ES7 215-1BG40-0XB0)

Pin	X10	X11 (gold)	X12
1	L1 /120-240 VAC	2 M	1L
2	N / 120 - 240 VAC	AQ 0	DQ a.0
3	Functional Earth	AQ 1	DQ a.1
4	L+ / 24 VDC Sensor Out	3M	DQ a.2
5	M / 24 VDC Sensor Out	AI 0	DQ a.3
6	1M	AI 1	DQ a.4
7	DI a.0	--	2L
8	DI a.1	--	DQ a.5
9	DI a.2	--	DQ a.6

Pin	X10	X11 (gold)	X12
10	DI a.3	--	DQ a.7
11	DI a.4	--	DQ b.0
12	DI a.5	--	DQ b.1
13	DI a.6	--	--
14	DI a.7	--	--
15	DI b.0	--	--
16	DI b.1	--	--
17	DI b.2	--	--
18	DI b.3	--	--
19	DI b.4	--	--
20	DI b.5	--	--

Table A- 84 CPU 1215C DC/DC/Relay (6ES7 215-1HG40-0XB0)



①	24 VDC Sensor Power Out For additional noise immunity, connect "M" to chassis ground even if not using sensor supply.
②	For sinking inputs, connect "-" to "M" (shown). For sourcing inputs, connect "+" to "M".
Note 1: X11 connectors must be gold. See Appendix C, Spare Parts for article number.	
Note 2: See Device Configuration (Page 145) for information about the Ethernet port of the CPU.	

Table A- 85 Connector pin locations for CPU 1215C DC/DC/Relay (6ES7 215-1HG40-0XB0)

Pin	X10	X11 (gold)	X12
1	L+ / 24 VDC	2 M	1L
2	M / 24 VDC	AQ 0	DQ a.0
3	Functional Earth	AQ 1	DQ a.1
4	L+ / 24 VDC Sensor Out	3M	DQ a.2
5	M / 24 VDC Sensor Out	AI 0	DQ a.3
6	1M	AI 1	DQ a.4
7	DI a.0	--	2L
8	DI a.1	--	DQ a.5
9	DI a.2	--	DQ a.6
10	DI a.3	--	DQ a.7
11	DI a.4	--	DQ b.0
12	DI a.5	--	DQ b.1
13	DI a.6	--	--
14	DI a.7	--	--
15	DI b.0	--	--
16	DI b.1	--	--
17	DI b.2	--	--
18	DI b.3	--	--
19	DI b.4	--	--
20	DI b.5	--	--

Table A- 86 CPU 1215C DC/DC/DC (6ES7 215-1AG40-0XB0)

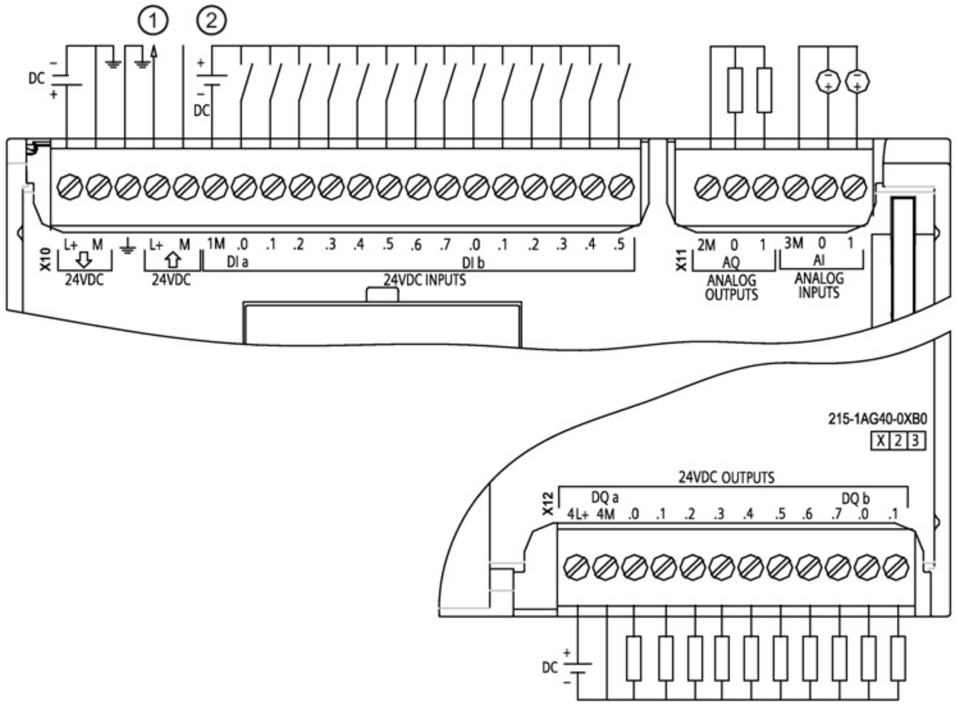
 ① ② DC DC X10 L+ M 24VDC L+ M 24VDC 1M .0 .1 .2 .3 .4 .5 .6 .7 .0 .1 .2 .3 .4 .5 DI a DI b 24VDC INPUTS X11 2M 0 1 AQ ANALOG OUTPUTS 3M 0 1 AI ANALOG INPUTS 215-1AG40-0XB0 X 2 3 24VDC OUTPUTS X12 DQ a DQ b 4L+ 4M .0 .1 .2 .3 .4 .5 .6 .7 .0 .1 DC	
①	24 VDC Sensor Power Out For additional noise immunity, connect "M" to chassis ground even if not using sensor supply.
②	For sinking inputs, connect "-" to "M" (shown). For sourcing inputs, connect "+" to "M".
Note 1: X11 connectors must be gold. See Appendix C, Spare Parts for article number.	
Note 2: See Device Configuration (Page 145) for information about the Ethernet port of the CPU.	

Table A- 87 Connector pin locations for CPU 1215C DC/DC/DC (6ES7 215-1AG40-0XB0)

Pin	X10	X11 (gold)	X12
1	L1 / 24 VDC	2 M	4L+
2	M / 24 VDC	AQ 0	4M
3	Functional Earth	AQ 1	DQ a.0
4	L+ / 24 VDC Sensor Out	3M	DQ a.1
5	M / 24 VDC Sensor Out	AI 0	DQ a.2
6	1M	AI 1	DQ a.3
7	DI a.0	--	DQ a.4
8	DI a.1	--	DQ a.5
9	DI a.2	--	DQ a.6
10	DI a.3	--	DQ a.7
11	DI a.4	--	DQ b.0
12	DI a.5	--	DQ b.1
13	DI a.6	--	--

Pin	X10	X11 (gold)	X12
14	DI a.7	--	--
15	DI b.0	--	--
16	DI b.1	--	--
17	DI b.2	--	--
18	DI b.3	--	--
19	DI b.4	--	--
20	DI b.5	--	--

Note

Unused analog inputs should be shorted.

A.6 CPU 1217C

A.6.1 General specifications and features

Table A- 88 General

Technical data	CPU 1217C DC/DC/DC
Article number	6ES7 217-1AG40-0XB0
Dimensions W x H x D (mm)	150 x 100 x 75
Shipping weight	530 grams
Power dissipation	12 W
Current available (SM and CM bus)	1600 mA max. (5 VDC)
Current available (24 VDC)	400 mA max. (sensor power)
Digital input current consumption (24 VDC)	4 mA/input used

Table A- 89 CPU features

Technical data		Description
User memory (Refer to "General technical specifications (Page 1099)", Internal CPU memory retention".)	Work	150 Kbytes
	Load	4 Mbytes, internal, expandable up to SD card size
	Retentive	10 Kbytes
On-board digital I/O		14 inputs/ 10 outputs
On-board analog I/O		2 inputs/ 2 outputs
Process image size		1024 bytes of inputs (I) / 1024 bytes of outputs (Q)
Bit memory (M)		8192 bytes
Temporary (local) memory		• 16 Kbytes for startup and program cycle (including associated FBs and FCs) • 6 Kbytes for each of the other interrupt priority levels (including FBs and FCs)
Signal modules expansion		8 SMs max.
SB, CB, BB expansion		1 max.
Communication module expansion		3 CMs max.
High-speed counters		Up to 6 configured to use any built-in or SB inputs (refer to CPU 1217C Digital input (DI) H/W configuration table) (Page 1158) • 1 MHz (Ib.2 to Ib.5) • 100/[1]80 kHz (Ia.0 to Ia.5) • 30/[1]20 kHz (Ia.6 to Ib.1)

Technical data	Description
Pulse outputs	Up to 4 configured to use any built-in or SB outputs (refer to CPU 1217C Digital output (DQ) H/W configuration table) (Page 1158) • 1 MHz (Qa.0 to Qa.3) • 100 kHz (Qa.4 to Qb.1)
Pulse catch inputs	14
Time delay interrupts	4 total with 1 ms resolution
Cyclic interrupts	4 total with 1 ms resolution
Edge interrupts	12 rising and 12 falling (16 and 16 with optional signal board)
Memory card	SIMATIC Memory Card (optional)
Real time clock accuracy	+/- 60 seconds/month
Real time clock retention time	20 days typ./12 days min. at 40 °C (maintenance-free Super Capacitor)

[1] The slower speed is applicable when the HSC is configured for quadrature mode of operation.

Table A- 90 Performance

Type of instruction	Execution speed
Boolean	0.08 μs/instruction
Move Word	1.7 μs/instruction
Real math	2.3 μs/instruction

A.6.2 Timers, counters and code blocks supported by CPU 1217C

Table A- 91 Blocks, timers and counters supported by CPU 1217C

Element		Description
Blocks	Type	OB, FB, FC, DB
	Size	64 Kbytes
	Quantity	Up to 1024 blocks total (OBs + FBs + FCs + DBs)
	Address range for FBs, FCs, and DBs	FB and FC: 1 to 65535 (such as FB 1 to FB 65535) DB: 1 to 59999
	Nesting depth	16 from the program cycle or startup OB 6 from any interrupt event OB
	Monitoring	Status of 2 code blocks can be monitored simultaneously
OBs	Program cycle	Multiple
	Startup	Multiple
	Time-delay interrupts	4 (1 per event)
	Cyclic interrupts	4 (1 per event)
	Hardware interrupts	50 (1 per event)
	Time error interrupts	1
	Diagnostic error interrupts	1
	Pull or plug of modules	1

Element		Description
	Rack or station failure	1
	Time of day	Multiple
	Status	1
	Update	1
	Profile	1
Timers	Type	IEC
	Quantity	Limited only by memory size
	Storage	Structure in DB, 16 bytes per timer
Counters	Type	IEC
	Quantity	Limited only by memory size
	Storage	Structure in DB, size dependent upon count type • SInt, USInt: 3 bytes • Int, UInt: 6 bytes • DInt, UDInt: 12 bytes

Table A- 92 Communication

Technical data	Description
Number of ports	2
Type	Ethernet
HMI device	4
Programming device (PG)	1
Connections	• 8 for Open User Communication (active or passive): TSEND_C, TRCV_C, TCON, TDISCON, TSEND, and TRCV • 3 for server GET/PUT (CPU-to-CPU) S7 communication • 8 for client GET/PUT (CPU-to-CPU) S7 communication
Data rates	10/100 Mb/s
Isolation (external signal to PLC logic)	Transformer isolated, 1500 VAC, for short term event safety only
Cable type	CAT5e shielded

Table A- 93 Power supply

Technical data		CPU 1217C DC/DC/DC
Voltage range		20.4 VDC to 28.8 VDC
Line frequency		--
Input current (max. load)	CPU only	600 mA at 24 VDC
	CPU with all expansion accessories	1600 mA at 24 VDC
Inrush current (max.)		12 A at 28.8 VDC
Isolation (input power to logic)		Not isolated
Hold up time (from loss of power)		10 ms at 24 VDC
Internal fuse, not user replaceable		3 A, 250 V, slow blow

Table A- 94 Sensor power

Technical data	CPU 1217C DC/DC/DC
Voltage range	L+ minus 4 VDC min.
Output current rating (max.)	400 mA (short-circuit protected)
Maximum ripple noise (<10 MHz)	Same as input line
Isolation (CPU logic to sensor power)	Not isolated

A.6.3 Digital inputs and outputs

Table A- 95 Digital inputs

Technical data	CPU 1217C DC/DC/DC
Number of inputs	14: total: 10: Sink/source (IEC Type 1 sink) 4: Differential (RS422/RS485)
Type: Sink/source (IEC Type 1 sink)	Ia.0 to Ia.7, Ib.0 to Ib.1
Rated voltage	24 VDC at 4 mA, nominal
Continuous permissible voltage	30 VDC, max.
Surge voltage	35 VDC for 0.5 sec.
Logic 1 signal (min.)	15 VDC at 2.5 mA
Logic 0 signal (max.)	5 VDC at 1 mA
Isolation (field side to logic)	500 VAC for 1 minute (functional isolation)
Isolation groups	1
Filter times	us settings: 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 10.0, 12.8, 20.0 ms settings: 0.05, 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 10.0, 12.8, 20.0
HSC clock input rates (max.) (Logic 1 Level = 15 to 26 VDC)	100/80 kHz (Ia.0 to Ia.5) 30/20 kHz (Ia.6 to Ib.1)
Type: Differential input (RS422/RS485)	Ib.2 to Ib.5 (.2+ .2- to .5+ .5-)
Common mode voltage range	-7 V to +12 V, 1 second, 3 VRMS continuous (RS422/RS485 characteristics)
Built-in termination and bias	390 Ω to 2M on Ib'-', 390 Ω to +5 V on Ib'-', (biased OFF when T/B open-circuit) 220 Ω between Ib'+' and Ib'-'
Receiver input impedance	100 Ω including bias and termination
Differential receiver threshold/sensitivity	+/- 0.2 V min., 60 mV typical hysteresis (RS422/RS485 characteristics)
Isolation (field side to logic)	500 VAC for 1 minute (functional isolation)
Isolation groups	1
Filter times	us settings: 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 10.0, 12.8, 20.0 ms settings: 0.05, 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 10.0, 12.8, 20.0

Technical data	CPU 1217C DC/DC/DC
HSC clock input rates (max.)	Single phase: 1 MHz (Ib.2 to Ib.5) Quadrature phase: 1 MHz (Ib.2 to Ib.5)
Differential input channel-to-channel skew	40 ns max.
General specifications (all digital inputs)	
Number of inputs on simultaneously	5 Sink/source inputs (no adjacent points) and 4 differential inputs at 60 °C horizontal or 50 °C vertical 14 at 55 °C horizontal or 45 °C vertical
Cable length (meters)	500 m shielded, 300 m unshielded 50 m shielded for HSC inputs (sink/source) 50 m shielded, twisted pair for all differential inputs

Table A- 96 CPU 1217C Digital input (DI) H/W configuration table

Input	Type and rate
DIa.0	Type: 24 V, source-sink Type 1 input High-speed counter input rate: 100 kHz max.
DIa.1	Type: 24 V, source-sink Type 1 input High-speed counter input rate: 100 kHz max.
DIa.2	Type: 24 V, source-sink Type 1 input High-speed counter input rate: 100 kHz max.
DIa.3	Type: 24 V, source-sink Type 1 input High-speed counter input rate: 100 kHz max.
DIa.4	Type: 24 V, source-sink Type 1 input High-speed counter input rate: 100 kHz max.
DIa.5	Type: 24 V, source-sink Type 1 input High-speed counter input rate: 100 kHz max.
DIa.6	Type: 24 V, source-sink Type 1 input High-speed counter input rate: 30 kHz max.
DIa.7	Type: 24 V, source-sink Type 1 input High-speed counter input rate: 30 kHz max.
DIb.0	Type: 24 V, source-sink Type 1 input High-speed counter input rate: 30 kHz max.
DIb.1	Type: 24 V, source-sink Type 1 input High-speed counter input rate: 30 kHz max.
DIb.2+ .2-	Type: RS422/RS485 differential input High-speed counter input rate: 1 MHz max.
DIb.3+ .3-	Type: RS422/RS485 differential input High-speed counter input rate: 1 MHz max.

Input	Type and rate
DIb.4+ .4-	Type: RS422/RS485 differential input High-speed counter input rate: 1 MHz max.
DIb.5+ .5-	Type: RS422/RS485 differential input High-speed counter input rate: 1 MHz max.

Table A- 97 Digital outputs

Technical data	CPU 1217C DC/DC/DC
Number of outputs	10 total 6: Solid state - MOSFET (sourcing) 4: Differential (RS422/RS485)
Type: Solid state - MOSFET (sourcing output)	Qa.4 to Qb.1
Voltage range	20.4 to 28.8 VDC
Logic 1 signal at max. current	20 VDC min.
Logic 0 signal with 10 KΩ load	0.1 VDC max.
Current (max.)	0.5 A
Lamp load	5 W
ON state resistance	0.6 Ω max.
Leakage current per point	10 μA max.
Surge current	8 A for 100 ms max.
Overload protection	No
Isolation (field side to logic)	500 VAC for 1 minute (functional isolation)
Isolation groups	1
Inductive clamp voltage	L+ minus 48 VDC, 1 W dissipation
Switching delay (Qa.0 to Qa.3)	1.0 μs max., off to on 3.0 μs max., on to off
Switching delay (Qa.4 to Qb.1)	50 μs max., off to on 200 μs max., on to off
Maximum relay switching frequency	--
Pulse Train Output rate	100 kHz max. (Qa.4 to Qb.1)[1], 2 Hz min.
Type: Differential output (RS422/RS485)	Qa.0 to Qa.3 (.0+ 0- to .3+ .3-)
Common mode voltage range	-7 V to +12 V, 1 second, 3 VRMS continuous (RS422/RS485 characteristics)
Transmitter differential output voltage	2 V min. at RL = 100 Ω, 1.5 V min. at RL = 54 Ω (RS422/RS485 characteristics)
Built-in termination	100 Ω between Qa'+' and Qa'-'
Driver output impedance	100 Ω including termination
Isolation	500 VAC, 1 minute (functional isolation)
Isolation groups	1
Switching delay (DQa.0 to DQa.3)	100 ns max.

Technical data	CPU 1217C DC/DC/DC
Differential output channel-to-channel skew	40 ns max.
Pulse train output rate	1 MHz (Qa.0 to Qa.3), 2 Hz min.
General specifications (all digital outputs)	
Behavior on RUN to STOP	Last value or substitute value (default value 0)
Number of outputs on simultaneously	3 Solid state - MOSFET (sourcing) outputs (no adjacent points) and 4 differential outputs at 60 °C horizontal or 50 °C vertical 10 at 55 °C horizontal or 45 °C vertical
Cable length (meters)	500 m shielded, 150 m unshielded

1 Depending on your pulse receiver and cable, an additional load resistor (at least 10% of rated current) may improve pulse signal quality and noise immunity.

Table A- 98 CPU 1217C Digital output (DQ) H/W configuration table

Output	Type and rate
DQa.0+ .0-	Type: RS422/RS485 differential output Pulse train output rate: 1 MHz max., 2 Hz min.
DQa.1+ .1-	Type: RS422/RS485 differential output Pulse train output rate: 1 MHz max., 2 Hz min.
DQa.2+ .2-	Type: RS422/RS485 differential output Pulse train output rate: 1 MHz max., 2 Hz min.
DQa.3+ .3-	Type: RS422/RS485 differential output Pulse train output rate: 1 MHz max., 2 Hz min.
DQa.4	Type: 24 V Sourcing output Pulse train output rate: 100 kHz max., 2 Hz min.
DQa.5	Type: 24 V Sourcing output Pulse train output rate: 100 kHz max., 2 Hz min.
DQa.6	Type: 24 V Sourcing output Pulse train output rate: 100 kHz max., 2 Hz min.
DQa.7	Type: 24 V Sourcing output Pulse train output rate: 100 kHz max., 2 Hz min.
DQb.0	Type: 24 V Sourcing output Pulse train output rate: 100 kHz max., 2 Hz min.
DQb.1	Type: 24 V Sourcing output Pulse train output rate: 100 kHz max., 2 Hz min.

A.6.4 Analog inputs and outputs

A.6.4.1 Analog input specifications

Table A- 99 Analog inputs

Technical data	Description
Number of inputs	2
Type	Voltage (single-ended)
Full-scale range	0 to 10 V
Full-scale range (data word)	0 to 27648
Overshoot range	10.001 to 11.759 V
Overshoot range (data word)	27649 to 32511
Overflow range	11.760 to 11.852 V
Overflow range (data word)	32512 to 32767
Resolution	10 bits
Maximum withstand voltage	35 VDC
Smoothing	None, Weak, Medium, or Strong See the table for Step response (ms) for the analog inputs of the CPU (Page 1162).
Noise rejection	10, 50, or 60 Hz
Impedance	≥100 KΩ
Isolation (field side to logic)	None
Accuracy (25 °C / 0 to 55 °C)	3.0% / 3.5% of full-scale
Cable length (meters)	100 m, shielded twisted pair

A.6.4.2 Step response of built-in analog inputs of the CPU

Table A- 100 Step Response (ms), 0 V to 10 V measured at 95%

Smoothing selection (sample averaging)	Rejection frequency (Integration time)		
	60 Hz	50 Hz	10 Hz
None (1 cycle): No averaging	50 ms	50 ms	100 ms
Weak (4 cycles): 4 samples	60ms	70 ms	200 ms
Medium (16 cycles): 16 samples	200 ms	240 ms	1150 ms
Strong (32 cycles): 32 samples	400 ms	480 ms	2300 ms
Sample time	**4.17 ms**	**5 ms**	**25 ms**

A.6.4.3 Sample time for the built-in analog ports of the CPU

Table A- 101 Sample time for built-in analog inputs of the CPU

Rejection frequency (Integration time selection)	Sample time
60 Hz (16.6 ms)	4.17 ms
50 Hz (20 ms)	5 ms
10 Hz (100 ms)	25 ms

A.6.4.4 Measurement ranges of the analog inputs for voltage (CPUs)

Table A- 102 Analog input representation for voltage (CPUs)

System		Voltage Measuring Range	
Decimal	**Hexadecimal**	**0 to 10 V**	
32767	7FFF	11.851 V	Overflow
32512	7F00		
32511	7EFF	11.759 V	Overshoot range
27649	6C01		
27648	6C00	10 V	Rated range
20736	5100	7.5 V	
34	22	12 mV	
0	0	0 V	
Negative values		Negative values are not sup-ported	

A.6.4.5 Analog output specifications

Table A- 103 Analog outputs

Technical data	Description
Number of outputs	2
Type	Current
Full-scale range	0 to 20 mA
Full-scale range (data word)	0 to 27648
Overshoot range	20.01 to 23.52 mA
Overshoot range (data word)	27649 to 32511
Overflow range	see footnote [1]
Overflow range data word	32512 to 32767
Resolution	10 bits
Output drive impedance	≤500 Ω max.
Isolation (field side to logic)	None
Accuracy (25 °C / -20 to 60 °C)	3.0% / 3.5% of full-scale

Technical data	Description
Settling time	2 ms
Cable length (meters)	100 m, shielded twisted pair

1 In an overflow condition, analog outputs will behave according to the device configuration properties settings. In the "Reaction to CPU STOP" parameter, select either: "Use substitute value" or "Keep last value".

Table A- 104 Analog output representation for current (CPU 1215C and CPU 1217C)

System		Current output range	
Decimal	**Hexadecimal**	**0 mA to 20 mA**	
32767	7FFF	See note 1	Overflow
32512	7F00	See note 1	
32511	7EFF	23.52 mA	Overshoot range
27649	6C01		
27648	6C00	20 mA	Rated range
20736	5100	15 mA	
34	22	0.0247 mA	
0	0	0 mA	
Negative values		Negative values are not supported	

1 In an overflow condition, analog outputs will behave according to the device configuration properties settings. In the "Reaction to CPU STOP" parameter, select either: "Use substitute value" or "Keep last value".

A.6.5 CPU 1217C wiring diagrams

Table A- 105 CPU 1217C DC/DC/DC (6ES7 217-1AG40-0XB0)

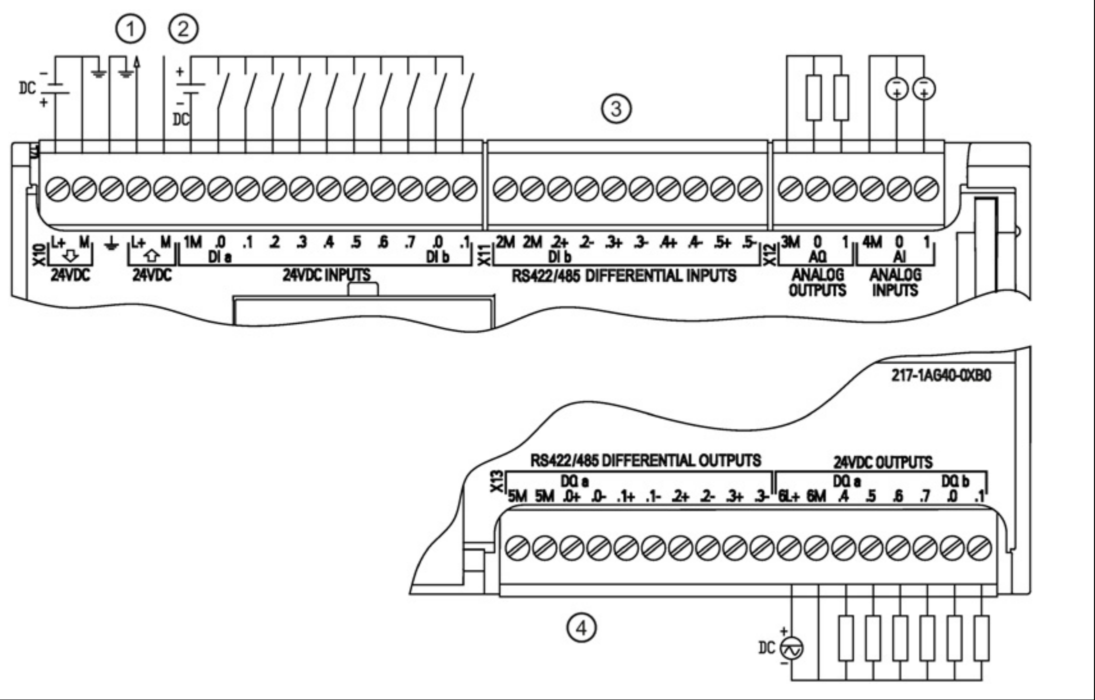
 ① ② ③ ④ DC X10 L+ M 24VDC L+ M 24VDC 1M .0 .1 .2 .3 .4 .5 .6 .7 .0 .1 DI a DI b 24VDC INPUTS X11 2M 2M 2+ 2- 3+ 3- 4+ 4- 5+ 5- DI b RS422/485 DIFFERENTIAL INPUTS X12 3M 0 1 AQ ANALOG OUTPUTS 4M 0 1 AI ANALOG INPUTS 217-1AG40-0XB0 RS422/485 DIFFERENTIAL OUTPUTS 24VDC OUTPUTS X13 DQ a DQ b 5M 5M .0+ .0- .1+ .1- .2+ .2- .3+ .3- 6L+ 6M .4 .5 .6 .7 .0 .1	
①	24 VDC Sensor Power Out For additional noise immunity, connect "M" to chassis ground even if not using sensor supply.
②	For sinking inputs, connect "-" to "M" (shown). For sourcing inputs, connect "+" to "M".
③	See CPU 1217C Differential input (DI) detail and application example (Page 1167).
④	See CPU 1217C Differential output (DQ) detail and application example (Page 1168).
Note 1: X12 connectors must be gold. See Appendix C, Spare Parts (Page 1282) for article number.	
Note 2: See Device configuration (Page 145) for information about the Ethernet port of the CPU.	

Table A- 106 Connector pin locations for CPU 1217C DC/DC/DC (6ES7 217-1AG40-0XB0)

Pin	X10	X11	X12 (gold)	X13
1	L+ / 24 VDC	2M	3M	5M
2	M / 24 VDC	2M	AQ 0	5M
3	Functional Earth	DI b.2+	AQ 1	DQ a.0+
4	L+ / 24 VDC Sensor Out	DI b.2-	4M	DQ a.0-
5	M / 24 VDC Sensor Out	DI b.3+	AI 0	DQ a.1+
6	1M	DI b.3-	AI 1	DQ a.1-
7	DI a.0	DI b.4+	--	DQ a.2+
8	DI a.1	DI b.4-	--	DQ a.2-
9	DI a.2	DI b.5+	--	DQ a.3+

Pin	X10	X11	X12 (gold)	X13
10	DI a.3	DI b.5-	--	DQ a.3-
11	DI a.4	--	--	6L+
12	DI a.5	--	--	6M
13	DI a.6	--	--	DQ a.4
14	DI a.7	--	--	DQ a.5
15	DI b.0	--	--	DQ a.6
16	DI b.1	--	--	DQ a.7
17	--	--	--	DQ b.0
18	--	--	--	DQ b.1

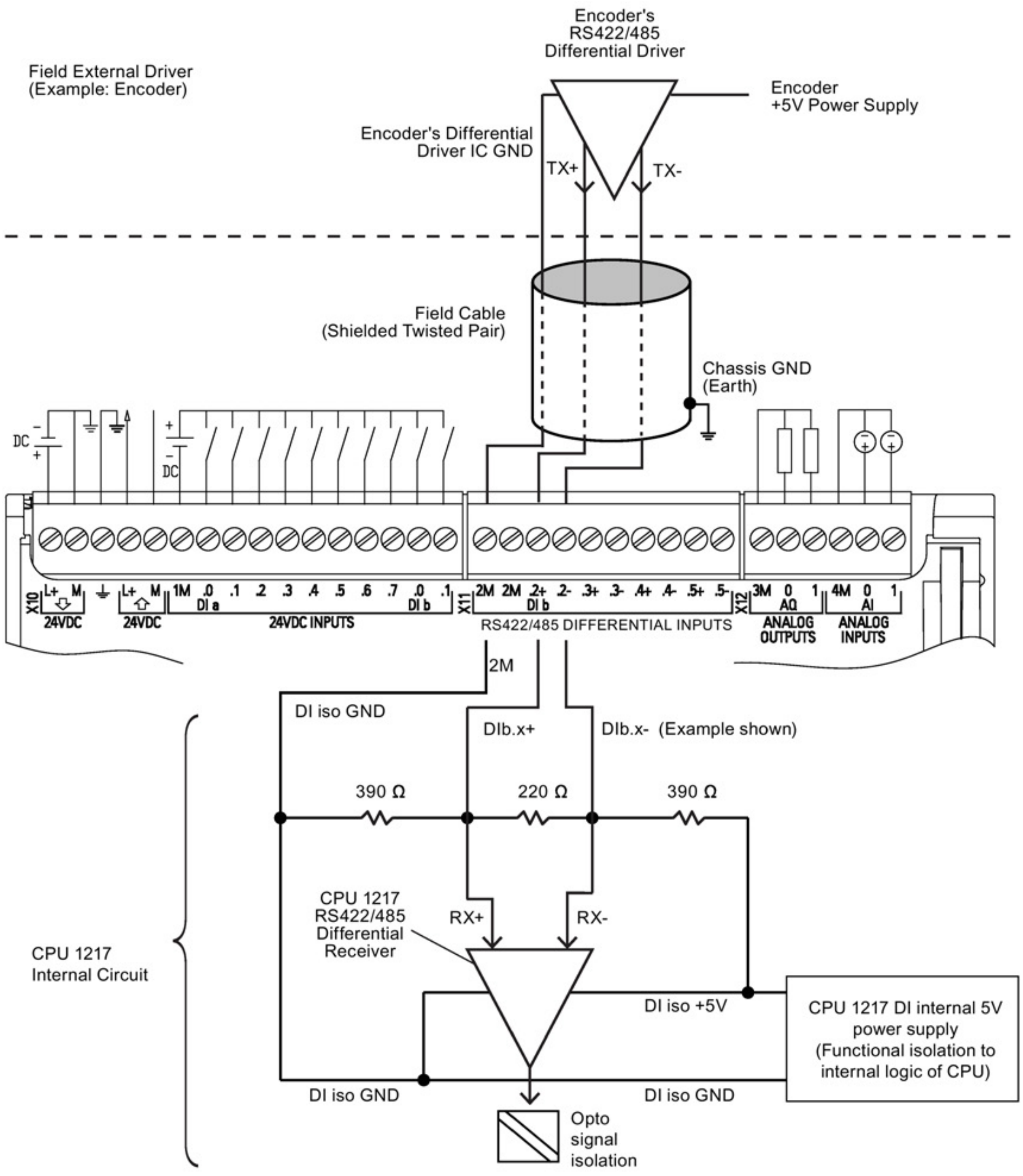
Note

Unused analog inputs should be shorted.

See also

Analog inputs and outputs (Page 1146)

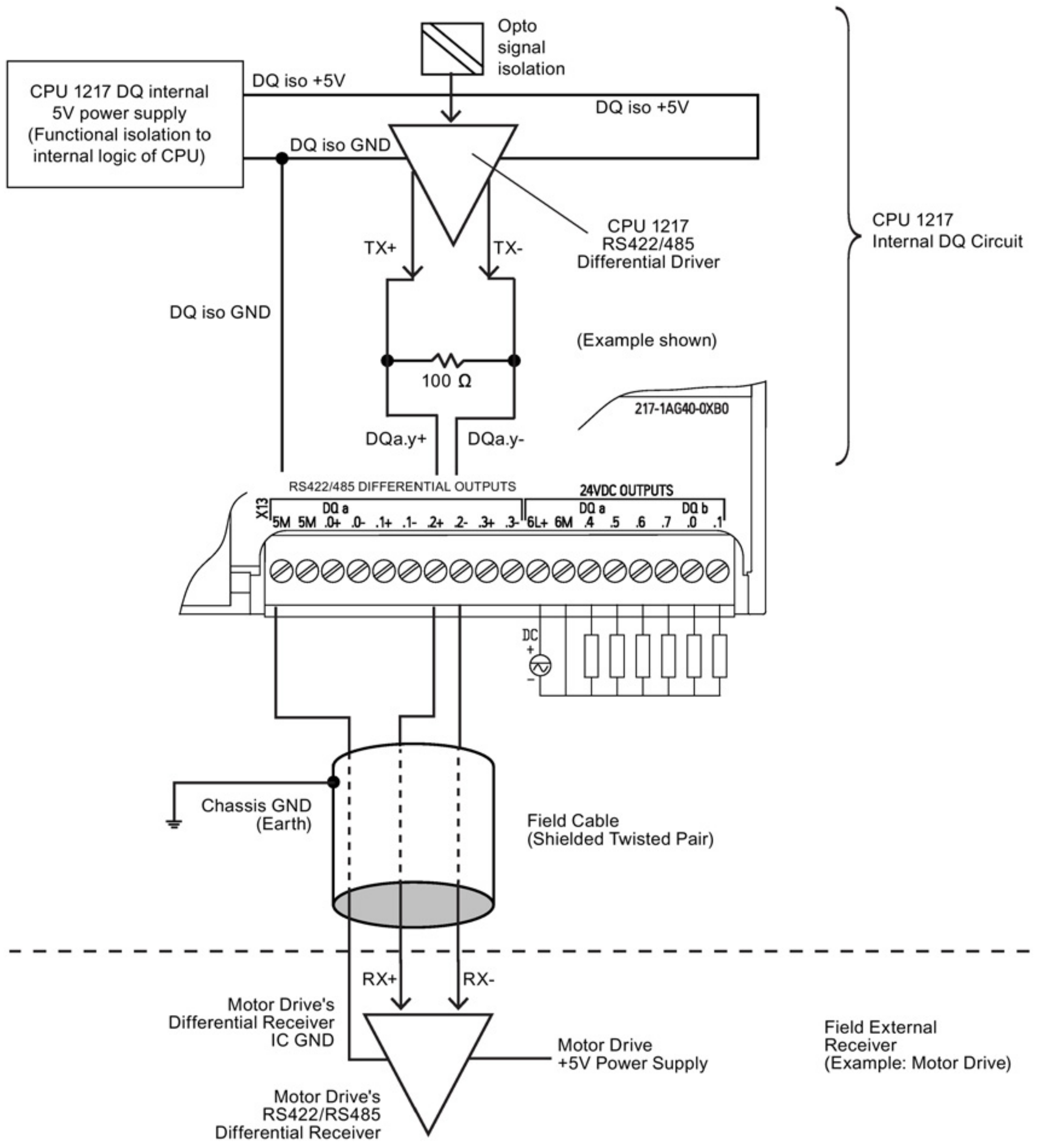
A.6.6 CPU 1217C Differential Input (DI) detail and application example


Encoder's RS422/485 Differential Driver
Field External Driver (Example: Encoder)
Encoder +5V Power Supply
Encoder's Differential Driver IC GND
TX+
TX-
Field Cable (Shielded Twisted Pair)
Chassis GND (Earth)
DC
24VDC
24VDC INPUTS
DI a
DI b
RS422/485 DIFFERENTIAL INPUTS
ANALOG OUTPUTS
ANALOG INPUTS
2M
DI iso GND
DIb.x+
DIb.x- (Example shown)
390 Ω
220 Ω
390 Ω
CPU 1217 RS422/485 Differential Receiver
RX+
RX-
CPU 1217 Internal Circuit
DI iso +5V
CPU 1217 DI internal 5V power supply (Functional isolation to internal logic of CPU)
DI iso GND
DI iso GND
Opto signal isolation

Notes

- Each differential DI is biased "OFF" when terminal block screws are open-circuit.
- Built-in DI Termination and Bias = 100 Ω equivalent impedance.
- Built-in DI Termination and Bias resistors limit the continuous common mode voltage range. See electrical specifications for detail.

A.6.7 CPU 1217C Differential Output (DQ) detail and application example


Opto signal isolation
CPU 1217 DQ internal 5V power supply (Functional isolation to internal logic of CPU)
DQ iso +5V
DQ iso +5V
DQ iso GND
CPU 1217 RS422/485 Differential Driver
CPU 1217 Internal DQ Circuit
TX+
TX-
DQ iso GND
(Example shown)
100 Ω
217-1AG40-0XB0
DQa.y+
DQa.y-
RS422/485 DIFFERENTIAL OUTPUTS
24VDC OUTPUTS
X13
DQ a
DQ a
DQ b
5M 5M .0+ .0- .1+ .1- .2+ .2- .3+ .3- 6L+ 6M .4 .5 .6 .7 .0 .1
DC
Chassis GND (Earth)
Field Cable (Shielded Twisted Pair)
RX+
RX-
Motor Drive's Differential Receiver IC GND
Motor Drive +5V Power Supply
Field External Receiver (Example: Motor Drive)
Motor Drive's RS422/RS485 Differential Receiver

Note

- Built-in DQ Termination resistor limits the continuous common mode voltage range. See electrical specifications for detail.

A.7 Digital signal modules (SMs)

A.7.1 SM 1221 digital input specifications

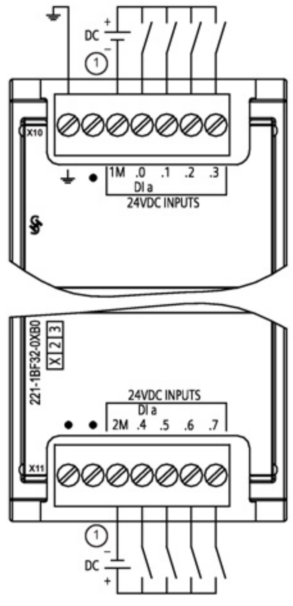
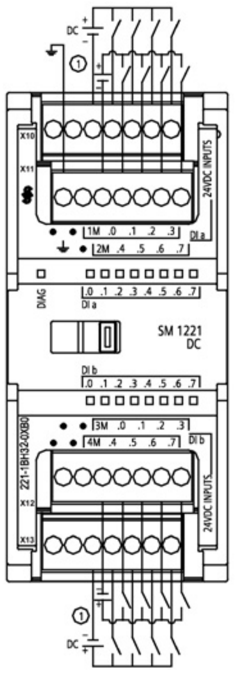
Table A- 107 General specifications

Model	SM 1221 DI 8 x 24 VDC	SM 1221 DI 16 x 24 VDC
Article number	6ES7 221-1BF32-0XB0	6ES7 221-1BH32-0XB0
Dimensions W x H x D (mm)	45 x 100 x 75	
Weight	170 grams	210 grams
Power dissipation	1.5 W	2.5 W
Current consumption (SM Bus)	105 mA	130 mA
Current consumption (24 VDC)	4 mA / input used	

Table A- 108 Digital inputs

Model	SM 1221 DI 8 x 24 VDC	SM 1221 DI 16 x 24 VDC
Number of inputs	8	16
Type	Sink/Source (IEC Type 1 sink)	
Rated voltage	24 VDC at 4 mA, nominal	
Continuous permissible voltage	30 VDC, max.	
Surge voltage	35 VDC for 0.5 sec.	
Logic 1 signal (min.)	15 VDC at 2.5 mA	
Logic 0 signal (max.)	5 VDC at 1 mA	
Isolation (field side to logic)	500 VAC for 1 minute	
Isolation groups	2	4
Filter times	0.2, 0.4, 0.8, 1.6, 3.2, 6.4, and 12.8 ms (selectable in groups of 4)	0.2, 0.4, 0.8, 1.6, 3.2, 6.4, and 12.8 ms (selectable in groups of 4)
Number of inputs on simultaneously	8	16
Cable length (meters)	500 m shielded, 300 unshielded	

Table A- 109 Wiring diagrams for the digital input SMs

SM 1221 DI 8 x 24 VDC (6ES7 221-1BF32-0XB0)	SM 1221 DI 16 x 24 VDC (6ES7 221-1BH32-0XB0)
	

① For sinking inputs, connect "-" to "M" (shown). For sourcing inputs, connect "+" to "M".

Table A- 110 Connector pin locations for SM 1221 DI 8 x 24 VDC (6ES7 221-1BF32-0XB0)

Pin	X10	X11
1	Functional Earth	No connection
2	No connection	No connection
3	1M	2M
4	DI a.0	DI a.4
5	DI a.1	DI a.5
6	DI a.2	DI a.6
7	DI a.3	DI a.7

Table A- 111 Connector pin locations for SM 1221 DI 16 x 24 VDC (6ES7 221-1BH32-0XB0)

Pin	X10	X11	X12	X13
1	No connection	Functional Earth	No connection	No connection
2	No connection	No connection	No connection	No connection
3	1M	2M	3 M	4 M

Pin	X10	X11	X12	X13
4	DI a.0	DI a.4	DI b.0	DI b.4
5	DI a.1	DI a.5	DI b.1	DI b.5
6	DI a.2	DI a.6	DI b.2	DI b.6
7	DI a.3	DI a.7	DI b.3	DI b.7

A.7.2 SM 1222 8-point digital output specifications

Table A- 112 General specifications

Model	SM 1222 DQ 8 x Relay	SM 1222 DQ 8 RLY Changeover	SM 1222 DQ 8 x 24 VDC
Article number	6ES7 222-1HF32-0XB0	6ES7 222-1XF32-0XB0	6ES7 222-1BF32-0XB0
Dimensions W x H x D (mm)	45 x 100 x 75	70 x 100 x 75	45 x 100 x 75
Weight	190 grams	310 grams	180 grams
Power dissipation	4.5 W	5 W	1.5 W
Current consumption (SM Bus)	120 mA	140 mA	120 mA
Current consumption (24 VDC)	11 mA / Relay coil used	16.7 mA/Relay coil used	50 mA

Table A- 113 Digital outputs

Model	SM 1222 DQ 8 x Relay	SM 1222 DQ8 RLY Changeover	SM 1222 DQ 8 x 24 VDC
Number of outputs	8		
Type	Relay, mechanical	Relay change over contact	Solid state - MOSFET (sourcing)
Voltage range	5 to 30 VDC or 5 to 250 VAC		20.4 to 28.8 VDC
Logic 1 signal at max. current	--		20 VDC min.
Logic 0 signal with 10K Ω load	--		0.1 VDC max
Current (max.)	2.0 A		0.5 A
Lamp load	30 W DC/200 W AC		5 W
ON state contact resistance	0.2 Ω max. when new		0.6 Ω max.
Leakage current per point	--		10 μA max.
Surge current	7 A with contacts closed		8 A for 100 ms max.
Overload protection	No		
Isolation (field side to logic)	1500 VAC for 1 minute (coil to contact) None (coil to logic)	1500 VAC for 1 minute (coil to contact)	500 VAC for 1 minute
Isolation resistance	100 MΩ min. when new		--
Isolation between open contacts	750 VAC for 1 minute		--
Isolation groups	2	8	1
Current per common (max.)	10 A	2 A	4 A

Model	SM 1222 DQ 8 x Relay	SM 1222 DQ8 RLY Changeover	SM 1222 DQ 8 x 24 VDC
Inductive clamp voltage	--		L+ minus 48 V, 1 W dissipation
Switching delay	10 ms max.		50 µs max. off to on 200 µs max. on to off
Maximum relay switching frequency	1 Hz		--
Lifetime mechanical (no load)	10,000,000 open/close cycles		--
Lifetime contacts at rated load (N.O. contact)	100,000 open/close cycles		--
Behavior on RUN to STOP	Last value or substitute value (default value 0)		
Number of outputs on simultaneously	8	• 4 (no adjacent points) at 60 °C horizontal or 50 °C vertical • 8 at 55 °C horizontal or 45 °C vertical	8
Cable length (meters)	500 m shielded, 150 m unshielded		

A.7.3 SM 1222 16-point digital output specifications

Table A- 114 General specifications

Model	SM 1222 DQ 16 x Relay	SM 1222 DQ 16 x 24 VDC
Article number	6ES7 222-1HH32-0XB0	6ES7 222-1BH32-0XB0
Dimensions W x H x D (mm)	45 x 100 x 75	
Weight	260 grams	220 grams
Power dissipation	8.5 W	2.5 W
Current consumption (SM Bus)	135 mA	140 mA
Current consumption (24 VDC)	11 mA / Relay coil used	100 mA

Table A- 115 Digital outputs

Model	SM1222 DQ 16 x Relay	SM1222 DQ 16 x 24 VDC
Number of outputs	16	
Type	Relay, mechanical	Solid state - MOSFET (sourcing)
Voltage range	5 to 30 VDC or 5 to 250 VAC	20.4 to 28.8 VDC
Logic 1 signal at max. current	-	20 VDC min.
Logic 0 signal with 10K Ω load	-	0.1 VDC max.
Current (max.)	2.0 A	0.5 A
Lamp load	30 W DC/200 W AC	5 W
ON state contact resistance	0.2 Ω max. when new	0.6 Ω max.

Model	SM1222 DQ 16 x Relay	SM1222 DQ 16 x 24 VDC
Leakage current per point	--	10 μA max.
Surge current	7 A with contacts closed	8 A for 100 ms max.
Overload protection	No	
Isolation (field side to logic)	1500 VAC for 1 minute (coil to contact) None (coil to logic)	500 VAC for 1 minute
Isolation resistance	100 MΩ min. when new	-
Isolation between open contacts	750 VAC for 1 minute	-
Isolation groups	4	1
Current per common (max.)	10 A	8 A
Inductive clamp voltage	-	L+ minus 48 V, 1 W dissipation
Switching delay	10 ms max.	50 μs max. off to on 200 μs max. on to off
Maximum relay switching frequency	1 Hz	-
Lifetime mechanical (no load)	10,000,000 open/close cycles	-
Lifetime contacts at rated load (N.O. contact)	100,000 open/close cycles	-
Behavior on RUN to STOP	Last value or substitute value (default value 0)	Last value or substitute value (default value 0)
Number of outputs on simultaneously	• 8 (no adjacent points) at 60 °C horizontal or 50 °C vertical • 16 at 55 °C horizontal or 45 °C vertical	16
Cable length (meters)	500 m shielded, 150 m unshielded	

Table A- 116 Wiring diagrams for the 8-point digital output SMs

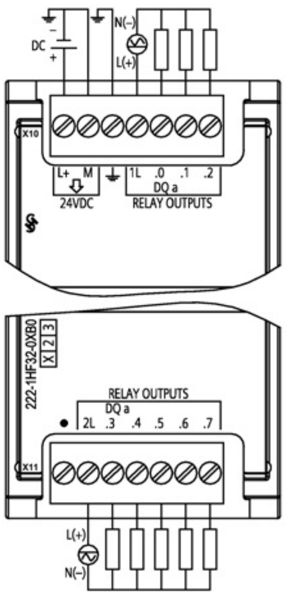
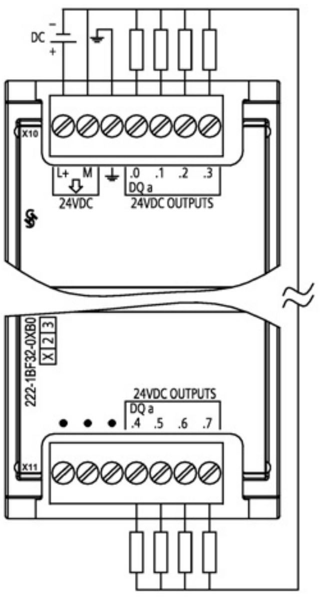
SM 1222 DQ 8 x Relay (6ES7 222-1HF32-0XB0)	SM 1222 DQ 8 x 24 VDC (6ES7 222-1BF32-0XB0)
 DC N(-) L(+) X10 L+ M 1L .0 .1 .2 DQ a 24VDC RELAY OUTPUTS 222-1HF32-0XB0 X 2 3 RELAY OUTPUTS DQ a 2L .3 .4 .5 .6 .7 X11 L(+) N(-)	 DC X10 L+ M .0 .1 .2 .3 DQ a 24VDC 24VDC OUTPUTS 222-1BF32-0XB0 X 2 3 24VDC OUTPUTS DQ a .4 .5 .6 .7 X11

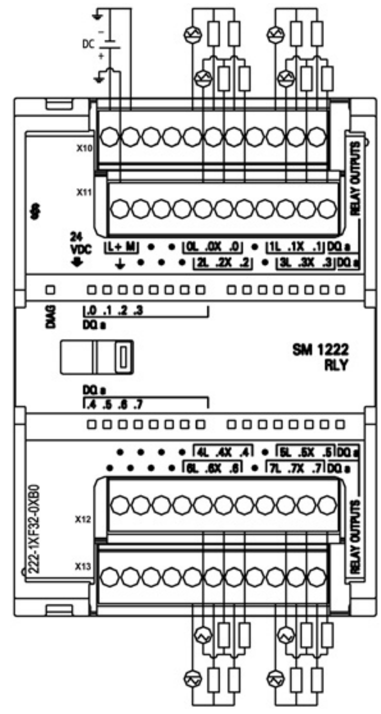
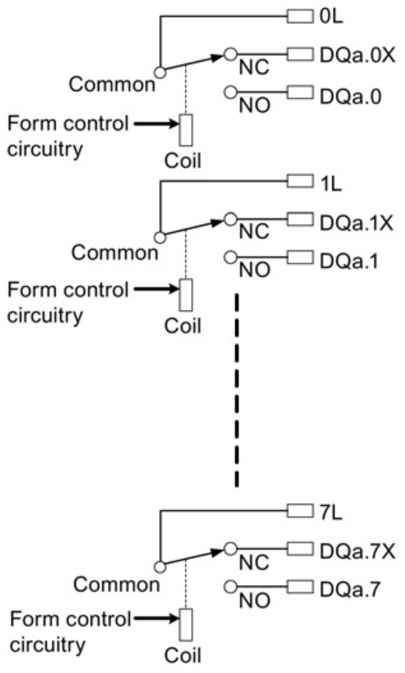
Table A- 117 Connector pin locations for SM 1222 DQ 8 x Relay (6ES7 222-1HF32-0XB0)

Pin	X10	X11
1	L+ / 24 VDC	No connection
2	M / 24 VDC	2L
3	Functional Earth	DQ a.3
4	1L	DQ a.4
5	DQ a.0	DQ a.5
6	DQ a.1	DQ a.6
7	DQ a.2	DQ a.7

Table A- 118 Connector pin locations for SM 1222 DQ 8 x 24 VDC (6ES7 222-1BF32-0XB0)

Pin	X10	X11
1	L+ / 24 VDC	No connection
2	M / 24 VDC	No connection
3	Functional Earth	No connection
4	DQ a.0	DQ a.4
5	DQ a.1	DQ a.5
6	DQ a.2	DQ a.6
7	DQ a.2	DQ a.7

Table A- 119 Wiring diagram for the 8-point digital output relay changeover SM

SM 1222 DQ 8 x Relay Changeover (6ES7 222-1XF32-0XB0)	
	 0L DQa.0X DQa.0 NC NO Common Form control circuitry Coil 1L DQa.1X DQa.1 7L DQa.7X DQa.7

A changeover relay output controls two circuits using a common terminal: one normally closed contact, and one normally open contact. Using output "0" as an example, when the output point is OFF, the common (0L) is connected to the normally closed contact (.0X) and disconnected from the normally open contact (.0). When the output point is ON, the common (0L) is disconnected from the normally closed contact (.0X) and connected to the normally open contact (.0).

Table A- 120 Connector pin locations for SM 1222 DQ 8 x Relay Changeover (6ES7 222-1XF32-0XB0)

Pin	X10	X11	X12	X13
1	L+ / 24 VDC	Functional Earth	No connection	No connection
2	M / 24 VDC	No connection	No connection	No connection
3	No connection	No connection	No connection	No connection
4	No connection	No connection	No connection	No connection
5	0L	2L	4L	6L
6	DQ a.0X	DQ a.2X	DQ a.4X	DQ a.6X
7	DQ a.0	DQ a.2	DQ a.4	DQ a.6
8	No connection	No connection	No connection	No connection
9	1L	3L	5L	7L
10	DQ a.1X	DQ a.3X	DQ a.5X	DQ a.7X
11	DQ a.1	DQ a.3	DQ a.5	DQ a.7

Table A- 121 Wiring diagrams for the 16-point digital output SMs

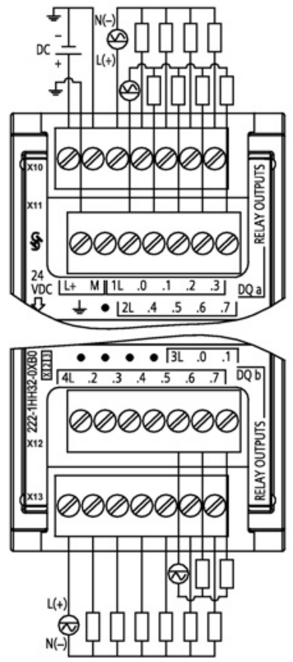
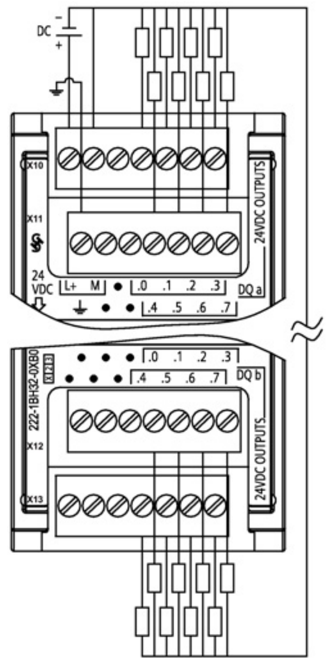
SM 1222 DQ 16 x Relay (6ES7 222-1HH32-0XB0)	SM 1222 DQ 16 x 24 VDC (6ES7 222-1BH32-0XB0)
 DC N(-) L(+) X10 X11 24 VDC L+ M 1L .0 .1 .2 .3 DQ a 2L .4 .5 .6 .7 RELAY OUTPUTS 222-1HH32-0XB0 3L .0 .1 4L .2 .3 .4 .5 .6 .7 DQ b X12 X13 RELAY OUTPUTS L(+) N(-)	 DC X10 X11 24 VDC L+ M .0 .1 .2 .3 DQ a .4 .5 .6 .7 24VDC OUTPUTS 222-1BH32-0XB0 .0 .1 .2 .3 .4 .5 .6 .7 DQ b X12 X13 24VDC OUTPUTS

Table A- 122 Connector pin locations for SM 1222 DQ 16 x Relay (6ES7 222-1HH32-0XB0)

Pin	X10	X11	X12	X13
1	L+ / 24 VDC	Functional Earth	No connection	4L
2	M / 24 VDC	No connection	No connection	DQ b.2
3	1L	2L	No connection	DQ b.3
4	DQ a.0	DQ a.4	No connection	DQ b.4
5	DQ a.1	DQ a.5	3L	DQ b.5
6	DQ a.2	DQ a.6	DQ b.0	DQ b.6
7	DQ a.3	DQ a.7	DQ b.1	DQ b.7

Table A- 123 Connector pin locations for SM 1222 DQ 16 x 24 VDC (6ES7 222-1BH32-0XB0)

Pin	X10	X11	X12	X13
1	L+ / 24 VDC	Functional Earth	No connection	No connection
2	M / 24 VDC	No connection	No connection	No connection
3	No connection	No connection	No connection	No connection
4	DQ a.0	DQ a.4	DQ b.0	DQ b.4
5	DQ a.1	DQ a.5	DQ b.1	DQ b.5

Pin	X10	X11	X12	X13
6	DQ a.2	DQ a.6	DQ b.2	DQ b.6
7	DQ a.3	DQ a.7	DQ b.3	DQ b.7

A.7.4 SM 1223 digital input/output VDC specifications

Table A- 124 General specifications

Model	SM 1223 DI 8 x 24 VDC, DQ 8 x Relay	SM 1223 DI 16 x 24 VDC, DQ 16 x Relay	SM 1223 DI 8 x 24 VDC, DQ 8 x 24 VDC	SM 1223 DI 16 x 24 VDC, DQ 16 x 24 VDC
Article number	6ES7 223-1PH32-0XB0	6ES7 223-1PL32-0XB0	6ES7 223-1BH32-0XB0	6ES7 223-1BL32-0XB0
Dimensions W x H x D (mm)	45 x 100 x 75	70 x 100 x 75	45 x 100 x 75	70 x 100 x 75
Weight	230 grams	350 grams	210 grams	310 grams
Power dissipation	5.5 W	10 W	2.5 W	4.5 W
Current consumption (SM Bus)	145 mA	180 mA	145 mA	185 mA
Current consumption (24 VDC)	4 mA / Input used 11 mA / Relay coil used		150 mA	200 mA

Table A- 125 Digital inputs

Model	SM 1223 DI 8 x 24 VDC, DQ 8 x Relay	SM 1223 DI 16 x 24 VDC, DQ 16 x Relay	SM 1223 DI 8 x 24 VDC, DQ 8 x 24 VDC	SM 1223 DI 16 x 24 VDC, DQ 16 x 24 VDC
Number of inputs	8	16	8	16
Type	Sink/Source (IEC Type 1 sink)			
Rated voltage	24 VDC at 4 mA, nominal			
Continuous permissible voltage	30 VDC max.			
Surge voltage	35 VDC for 0.5 sec.			
Logic 1 signal (min.)	15 VDC at 2.5 mA			
Logic 0 signal (max.)	5 VDC at 1 mA			
Isolation (field side to logic)	500 VAC 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	8	• 8 (no adjacent points) at 60 °C horizontal or 50 °C vertical • 16 at 55 °C horizontal or 45 °C vertical	8	16
Cable length (meters)	500 m shielded, 300 m unshielded			

Table A- 126 Digital outputs

Model	**SM 1223 DI 8 x 24 VDC, DQ 8 x Relay**	**SM 1223 DI 16 x 24 VDC, DQ 16 x Relay**	**SM 1223 DI 8 x 24 VDC, DQ 8 x 24 VDC**	**SM 1223 DI 16 x 24 VDC, DQ 16 x 24 VDC**
Number of outputs	8	16	8	16
Type	Relay, mechanical		Solid state - MOSFET (sourcing)	
Voltage range	5 to 30 VDC or 5 to 250 VAC		20.4 to 28.8 VDC	
Logic 1 signal at max. current	--		20 VDC, min.	
Logic 0 signal with 10 KΩ load	--		0.1 VDC, max.	
Current (max.)	2.0 A		0.5 A	
Lamp load	30 W DC / 200 W AC		5 W	
ON state contact resistance	0.2 Ω max. when new		0.6 Ω max.	
Leakage current per point	--		10 μA max.	
Surge current	7 A with contacts closed		8 A for 100 ms max.	
Overload protection	No			
Isolation (field side to logic)	1500 VAC for 1 minute (coil to contact) None (coil to logic)		500 VAC for 1 minute	
Isolation resistance	100 MΩ min. when new		--	
Isolation between open contacts	750 VAC for 1 minute		--	
Isolation groups	2	4	1	
Current per common	10A	8 A	4 A	8 A
Inductive clamp voltage	--		L+ minus 48 V, 1 W dissipation	
Switching delay	10 ms max.		50 μs max. off to on 200 μs max. on to off	
Maximum relay switching frequency	1 Hz		--	
Lifetime mechanical (no load)	10,000,000 open/close cycles		--	
Lifetime contacts at rated load (N.O. contact)	100,000 open/close cycles		--	
Behavior on RUN to STOP	Last value or substitute value (default value 0)			
Number of outputs on simultaneously	8	• 8 (no adjacent points) at 60 °C horizontal or 50 °C vertical • 16 at 55 °C horizontal or 45 °C vertical	8	16
Cable length (meters)	500 m shielded, 150 m unshielded			

Table A- 127 Wiring diagrams for the digital input VDC/output relay SMs

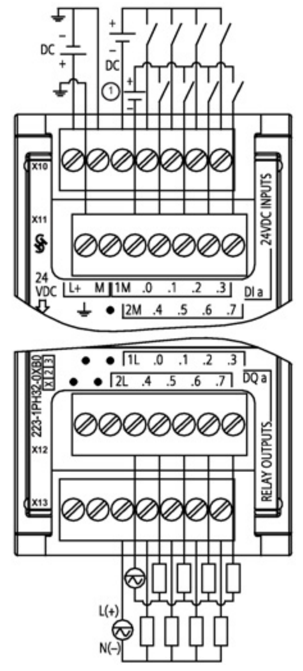
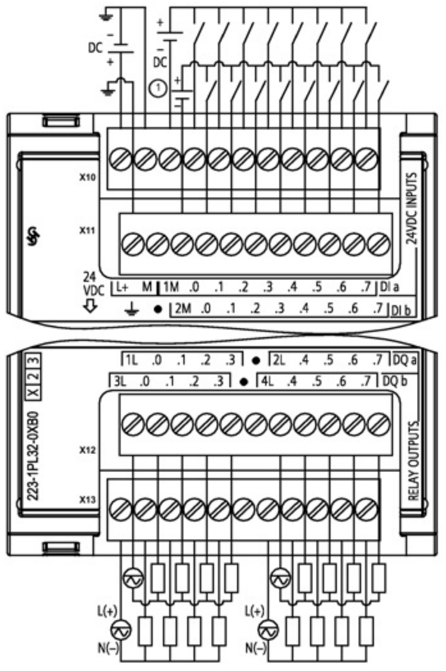
SM 1223 DI 8 x 24 VDC, DQ 8 x Relay (6ES7 223-1PH32-0XB0)	SM 1223 DI 16 x 24 VDC, DQ 16 x Relay (6ES7 223-1PL32-0XB0)	Notes
		① For sinking inputs, connect "-" to "M" (shown). For sourcing inputs, connect "+" to "M".

Table A- 128 Connector Pin Locations for SM 1223 DI 8 x 24 VDC, DQ 8 x Relay (6ES7 223-1PH32-0XB0)

Pin	X10	X11	X12	X13
1	L+ / 24 VDC	Functional Earth	No connection	No connection
2	M / 24 VDC	No connection	No connection	No connection
3	1M	2M	1L	2L
4	DI a.0	DI a.4	DQ a.0	DQ a.4
5	DI a.1	DI a.5	DQ a.1	DQ a.5
6	DI a.2	DI a.6	DQ a.2	DQ a.6
7	DI a.3	DI a.7	DQ a.3	DQ a.7

Table A- 129 Connector Pin Locations for SM 1223 DI 16 x 24 VDC, DQ 16 x Relay (6ES7 223-1PL32-0XB0)

Pin	X10	X11	X12	X13
1	L+ / 24 VDC	Functional Earth	1L	3L
2	M / 24 VDC	No connection	DQ a.0	DQ b.0
3	1M	2M	DQ a.1	DQ b.1
4	DI a.0	DI b.0	DQ a.2	DQ b.2
5	DI a.1	DI b.1	DQ a.3	DQ b.3
6	DI a.2	DI b.2	No connection	No connection
7	DI a.3	DI b.3	2L	4L
8	DI a.4	DI b.4	DQ a.4	DQ b.4
9	DI a.5	DI b.5	DQ a.5	DQ b.5
10	DI a.6	DI b.6	DQ a.6	DQ b.6
11	DI a.7	DI b.7	DQ a.7	DQ b.7

Table A- 130 Wiring diagrams for the digital input VDC/output SMs

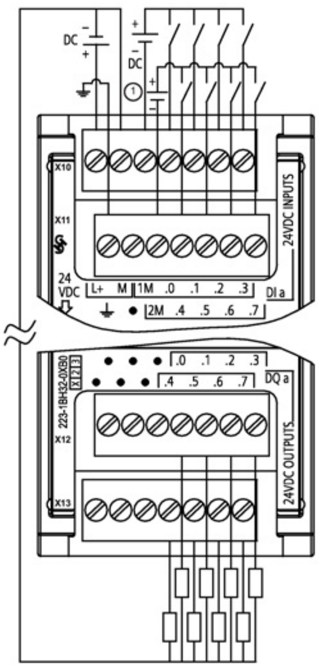
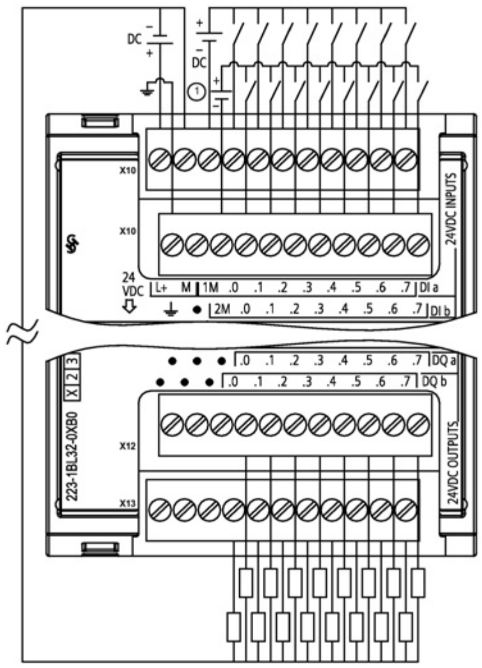
SM 1223 DI 8 x 24 VDC, DQ 8 x 24 VDC (6ES7 223-1BH32-0XB0)	SM 1223 DI 16 x 24 VDC, DQ 16 x 24 VDC (6ES7 223-1BL32-0XB0)	Notes
 DC DC ① X10 X11 24VDC INPUTS 24 VDC L+ M 1M .0 .1 .2 .3 DI a 2M .4 .5 .6 .7 .0 .1 .2 .3 DQ a .4 .5 .6 .7 223-1BH32-0XB0 X12 X13 24VDC OUTPUTS	 DC DC ① X10 X10 24VDC INPUTS 24 VDC L+ M 1M .0 .1 .2 .3 .4 .5 .6 .7 DI a 2M .0 .1 .2 .3 .4 .5 .6 .7 DI b .0 .1 .2 .3 .4 .5 .6 .7 DQ a .0 .1 .2 .3 .4 .5 .6 .7 DQ b 223-1BL32-0XB0 X12 X13 24VDC OUTPUTS	① For sinking inputs, connect "-" to "M" (shown). For sourcing inputs, connect "+" to "M".

Table A- 131 Connector Pin Locations for SM 1223 DI 8 x 24 VDC, DQ 8 x 24 VDC (6ES7 223-1BH32-0XB0)

Pin	X10	X11	X12	X13
1	L+ / 24 VDC	Functional Earth	No connection	No connection
2	M / 24 VDC	No connection	No connection	No connection
3	1M	2M	No connection	No connection
4	DI a.0	DI a.4	DQ a.0	DQ a.4
5	DI a.1	DI a.5	DQ a.1	DQ a.5
6	DI a.2	DI a.6	DQ a.2	DQ a.6
7	DI a.3	DI a.7	DQ a.3	DQ a.7

Table A- 132 Connector Pin Locations for SM 1223 DI 16 x 24 VDC, DQ 16 x 24 VDC (6ES7 223-1BL32-0XB0)

Pin	X10	X11	X12	X13
1	L+ / 24 VDC	Functional Earth	No connection	No connection
2	M / 24 VDC	No connection	No connection	No connection
3	1M	2M	No connection	No connection
4	DI a.0	DI b.0	DQ a.0	DQ b.0
5	DI a.1	DI b.1	DQ a.1	DQ b.1
6	DI a.2	DI b.2	DQ a.2	DQ b.2
7	DI a.3	DI b.3	DQ a.3	DQ b.3
8	DI a.4	DI b.4	DQ a.4	DQ b.4
9	DI a.5	DI b.5	DQ a.5	DQ b.5
10	DI a.6	DI b.6	DQ a.6	DQ b.6
11	DI a.7	DI b.7	DQ a.7	DQ b.7

A.7.5 SM 1223 digital input/output AC specifications

Table A- 133 General specifications

Model	SM 1223 DI 8 x120/230 VAC / DQ 8 x Relay
Article number	6ES7 223-1QH32-0XB0
Dimensions W x H x D (mm)	45 x 100 x 75 mm
Weight	190 grams
Power dissipation	7.5 W
Current consumption (SM Bus)	120 mA
Current consumption (24 VDC)	11 mA per output when on

Table A- 134 Digital inputs

Model	SM 1223 DI 8 x 120/230 VAC / DQ 8 x Relay
Number of inputs	8
Type	IEC Type 1
Rated voltage	120 VAC at 6 mA, 230 VAC at 9 mA
Continuous permissible voltage	264 VAC
Surge voltage	--
Logic 1 signal (min.)	79 VAC at 2.5 mA
Logic 0 signal (max.)	20 VAC at 1 mA
Leakage current (max.)	1 mA
Isolation (field side to logic)	1500 VAC for 1 minute
Isolation groups[1]	4
Input delay times	Typical: 0.2 to 12.8 ms, user selectable Maximum: -
Connection of 2 wire proximity sensor (Bero) (max.)	1 mA
Cable length	Unshielded: 300 meters Shielded: 500 meters
Number of inputs on simultaneously	8

[1] Channels within a group must be of the same phase.

Table A- 135 Digital outputs

Model	SM 1223 DI 8 x 120/230 VAC / DQ 8 x Relay
Number of outputs	8
Type	Relay, mechanical
Voltage range	5 to 30 VDC or 5 to 250 VAC
Logic 1 signal at max. current	--
Logic 0 signal with 10K Ω load	--
Current (max.)	2.0 A
Lamp load	30 W DC / 200 W AC
ON state contact resistance	0.2 Ω max. when new
Leakage current per point	--
Surge current	7 A with contacts closed
Overload protection	No
Isolation (field side to logic)	1500 VAC for 1 minute (coil to contact) None (coil to logic)
Isolation resistance	100 MΩ min. when new
Isolation between open contacts	750 VAC for 1 minute
Isolation groups	2
Current per common (max.)	10 A
Inductive clamp voltage	--
Switching delay (max.)	10 ms
Maximum relay switching frequency	1 Hz
Lifetime mechanical (no load)	10,000,000 open/close cycles
Lifetime contacts at rated load	1000,000 open/close cycles
Behavior on RUN to STOP	Last value or substitute value (default value 0)
Number of outputs on simultaneously	• 4 (no adjacent points) at 60 °C horizontal or 50 °C vertical • 8 at 55 °C horizontal or 45 °C vertical
Cable length (meters)	500 m shielded, 150 m unshielded

Table A- 136 SM 1223 DI 8 x 120/230 VAC, DQ 8 x Relay (6ES7 223-1QH32-0XB0)

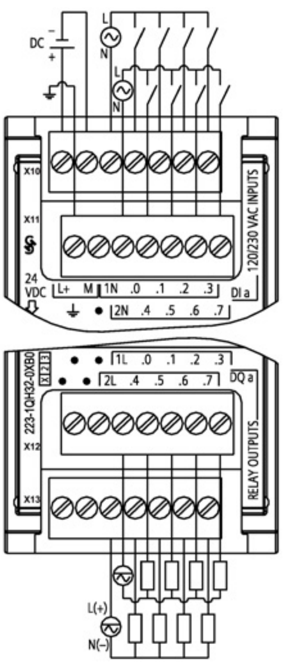
 DC L N L N X10 X11 24 VDC L+ M 1N .0 .1 .2 .3 DI a 2N .4 .5 .6 .7 120/230 VAC INPUTS 223-1QH32-0XB0 X12 X13 1L .0 .1 .2 .3 2L .4 .5 .6 .7 DQ a RELAY OUTPUTS L(+) N(-)	

Table A- 137 Connector Pin Locations for SM 1223 DI 8 x 120/240 VAC, DQ 8 x Relay (6ES7 223-1QH32-0XB0)

Pin	X10	X11	X12	X13
1	L+ / 24 VDC	Functional Earth	No connection	No connection
2	M / 24 VDC	No connection	No connection	No connection
3	1N	2N	1L	2L
4	DI a.0	DI a.4	DQ a.0	DQ a.4
5	DI a.1	DI a.5	DQ a.1	DQ a.5
6	DI a.2	DI a.6	DQ a.2	DQ a.6
7	DI a.3	DI a.7	DQ a.3	DQ a.7

A.8 Analog signal modules (SMs)

A.8.1 SM 1231 analog input module specifications

Table A- 138 General specifications

Model	SM 1231 AI 4 x 13 bit	SM 1231 AI 8 x 13 bit	SM 1231 AI 4 x 16 bit
Article number	6ES7 231-4HD32-0XB0	6ES7 231-4HF32-0XB0	6ES7 231-5ND30-0XB0
Dimensions W x H x D (mm)	45 x 100 x 75		
Weight	180 grams		
Power dissipation	2.2 W	2.3 W	2.0 W
Current consumption (SM Bus)	80 mA	90 mA	80 mA
Current consumption (24 VDC)	45 mA		65 mA

Table A- 139 Analog inputs

Model	SM 1231 AI 4 x 13 bit	SM 1231 AI 8 x 13 bit	SM 1231 AI 4 x 16 bit
Number of inputs	4	8	4
Type	Voltage or current (differential): Selectable in groups of 2		Voltage or current (differential)
Range	±10 V, ±5 V, ±2.5 V, 0 to 20 mA, or 4 mA to 20 mA		±10 V, ±5 V, ±2.5 V, ±1.25 V, 0 to 20 mA or 4 mA to 20 mA
Full scale range (data word)	-27648 to 27648 voltage / 0 to 27648 current		
Overshoot/undershoot range (data word) Refer to the section on analog input ranges for voltage and current (Page 1194).	Voltage: 32511 to 27649 / -27649 to -32,512 Current: 32511 to 27649 / 0 to -4864		
Overflow/underflow (data word) Refer to the section on input ranges for voltage and current (Page 1194).	Voltage: 32767 to 32512 / -32513 to -32768 Current 0 to 20 mA: 32767 to 32512 / -4865 to -32768 Current 4 to 20 mA: 32767 to 32512 (values below -4864 indicate open wire)		
Resolution1	12 bits + sign bit		15 bits + sign bit
Maximum withstand voltage/current	±35 V / ±40 mA		
Smoothing	None, weak, medium, or strong Refer to the section on step response times (Page 1194).		
Noise rejection	400, 60, 50, or 10 Hz Refer to the section on sample rates (Page 1194).		
Input impedance	≥ 9 MΩ (voltage) / 280 Ω (current)		≥ 1 MΩ (voltage) / <315 Ω, >280 Ω (current)

Model	SM 1231 AI 4 x 13 bit	SM 1231 AI 8 x 13 bit	SM 1231 AI 4 x 16 bit
Isolation Field side to logic Logic to 24 VDC Field side to 24 VDC Channel to channel	None		 500 VAC 500 VAC 500 VAC None
Accuracy (25 °C / -20 to 60 °C)	±0.1% / ±0.2% of full scale		±0.1% / ±0.3% of full scale
Measuring principle	Actual value conversion		
Common mode rejection	40 dB, DC to 60 Hz		
Operational signal range[1]	Signal plus common mode voltage must be less than +12 V and greater than -12 V		
Cable length (meters)	100 m, twisted and shielded		

[1] Voltages outside the operational range applied to one channel may cause interference on other channels.

Table A- 140 Diagnostics

Model	SM 1231 AI 4 x 13 bit	SM 1231 AI 8 x 13 bit	SM 1231 AI 4 x 16 bit
Overflow/underflow	Yes		
24 VDC low voltage	Yes		
Open wire	4 to 20 mA range only (if input is below -4164; 1.185 mA)		

Table A- 141 Wiring diagrams for the analog input SMs

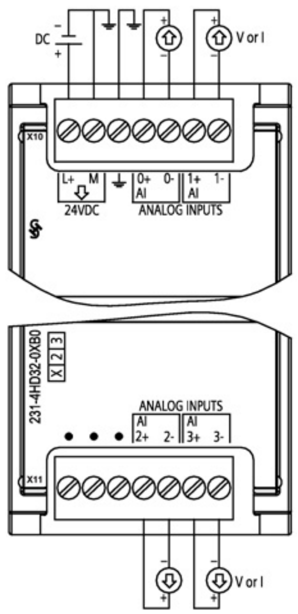
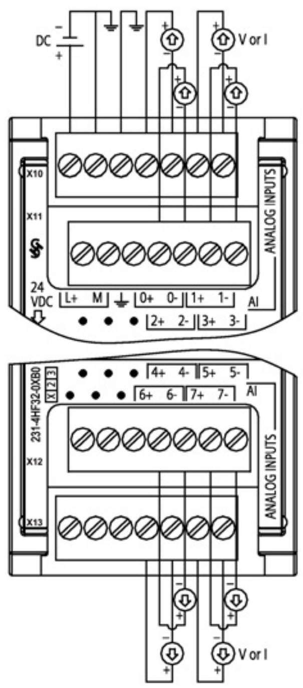
SM 1231 AI 4 x 13 bit (6ES7 231-4HD32-0XB0)	SM 1231 AI 8 x 13 bit (6ES7 231-4HF32-0XB0)
 DC V or I X10 L+ M 0+ 0- AI 1+ 1- AI 24VDC ANALOG INPUTS 231-4HD32-0XB0 X 2 3 ANALOG INPUTS AI 2+ 2- AI 3+ 3- X11 V or I	 DC V or I X10 X11 ANALOG INPUTS 24 VDC L+ M 0+ 0- 1+ 1- AI 2+ 2- 3+ 3- 231-4HF32-0XB0 X 2 3 4+ 4- 5+ 5- 6+ 6- 7+ 7- AI X12 X13 ANALOG INPUTS V or I
Note: Connectors must be gold. See Appendix C, Spare Parts for article number.	

Table A- 142 Connector pin locations for SM 1231 AI 4 x 13 bit (6ES7 231-4HD32-0XB0)

Pin	X10 (gold)	X11 (gold)
1	L+ / 24 VDC	No connection
2	M / 24 VDC	No connection
3	Functional Earth	No connection
4	AI 0+	AI 2+
5	AI 0-	AI 2-
6	AI 1+	AI 3+
7	AI 1-	AI 3-

Table A- 143 Connector pin locations for SM 1231 AI 8 x 13 bit (6ES7 231-4HF32-0XB0)

Pin	X10 (gold)	X11 (gold)	X12 (gold)	X13 (gold)
1	L+ / 24 VDC	No connection	No connection	No connection
2	M / 24 VDC	No connection	No connection	No connection
3	Functional Earth	No connection	No connection	No connection
4	AI 0+	AI 2+	AI 4+	AI 6+
5	AI 0-	AI 2-	AI 4-	AI 6-
6	AI 1+	AI 3+	AI 5+	AI 7+
7	AI 1-	AI 3-	AI 5-	AI 7-

Table A- 144 Wiring diagram for the analog input SM

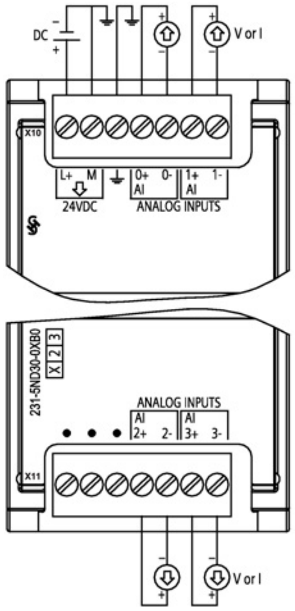
SM 1231 AI 4 x 16 bit (6ES7 231-5ND30-0XB0)	
 DC V or I X10 L+ M 24VDC 0+ 0- AI 1+ 1- AI ANALOG INPUTS 231-5ND30-0XB0 X 2 3 ANALOG INPUTS AI 2+ 2- AI 3+ 3- X11 V or I	
Note: Connectors must be gold. See Appendix C, Spare Parts for article number.	

Table A- 145 Connector pin locations for SM 1231 AI 4 x 16 bit (6ES7 231-5ND30-0XB0)

Pin	X10 (gold)	X11 (gold)
1	L+ / 24 VDC	No connection
2	M / 24 VDC	No connection
3	Functional Earth	No connection
4	AI 0+	AI 2+
5	AI 0-	AI 2-
6	AI 1+	AI 3+
7	AI 1-	AI 3-

Note

Unused voltage input channels should be shorted.

Unused curent input channels should be set to the 0 to 20 mA range and/or disable broken wire error reporting.

Inputs configured for current mode will not conduct loop current unless the module is powered and configured.

Current input channels will not operate unless external power is supplied to the transmitter.

A.8.2 SM 1232 analog output module specifications

Table A- 146 General specifications

Technical data	SM 1232 AQ 2 x 14 bit	SM 1232 AQ 4 x 14 bit
Article number	6ES7 232-4HB32-0XB0	6ES7 232-4HD32-0XB0
Dimensions W x H x D (mm)	45 x 100 x 75	
Weight	180 grams	
Power dissipation	1.8 W	2.0 W
Current consumption (SM Bus)	80 mA	
Current consumption (24 VDC)	45 mA (no load)	

Table A- 147 Analog outputs

Technical data	SM 1232 AQ 2 x 14 bit	SM 1232 AQ 4 x 14 bit
Number of outputs	2	4
Type	Voltage or current	
Range	±10 V, 0 to 20 mA, or 4 mA to 20 mA	
Resolution	Voltage: 14 bits Current: 13 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 1195).	
Accuracy (25 °C / -20 to 60 °C)	±0.3% / ±0.6% 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	
Behavior on RUN to STOP	Last value or substitute value (default value 0)	
Isolation (field side to logic)	none	
Cable length (meters)	100 m twisted and shielded	

Table A- 148 Diagnostics

Technical data	SM 1232 AQ 2 x 14 bit	SM 1232 AQ 4 x 14 bit
Overflow/underflow	Yes	
Short to ground (voltage mode only)	Yes	
Wire break (current mode only)	Yes	
24 VDC low voltage	Yes	

Table A- 149 Wiring diagrams for the analog output SMs

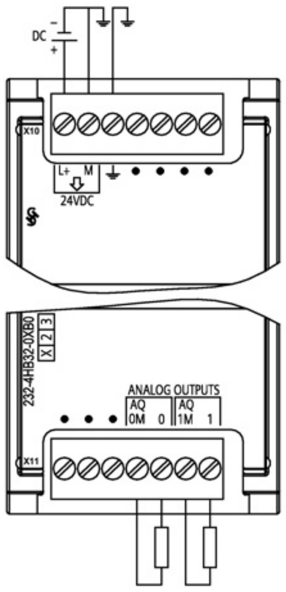
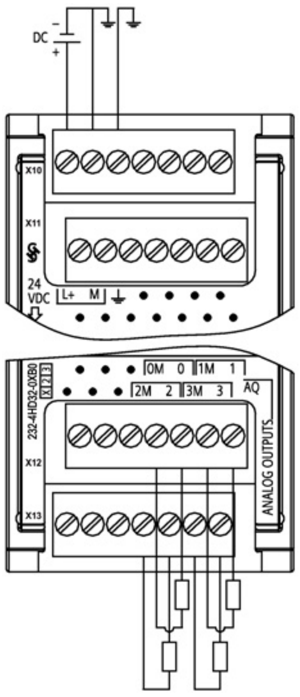
SM 1232 AQ 2 x 14 bit (6ES7 232-4HB32-0XB0)	SM 1232 AQ 4 x 14 bit (6ES7 232-4HD32-0XB0)
 DC X10 L+ M 24VDC 232-4HB32-0XB0 X 2 3 ANALOG OUTPUTS AQ 0M 0 AQ 1M 1 X11	 DC X10 X11 24 VDC L+ M 232-4HD32-0XB0 X 2 3 0M 0 1M 1 2M 2 3M 3 AQ X12 X13 ANALOG OUTPUTS
Note: Connectors must be gold. See Appendix C, Spare Parts for article number.	

Table A- 150 Connector pin locations for SM 1232 AQ 2 x 14 bit (6ES7 232-4HB32-0XB0)

Pin	X10 (gold)	X11 (gold)
1	L+ / 24 VDC	No connection
2	M / 24 VDC	No connection
3	Functional Earth	No connection
4	No connection	AQ 0M
5	No connection	AQ 0
6	No connection	AQ 1M
7	No connection	AQ 1

Table A- 151 Connector pin locations for SM 1232 AQ 4 x 14 bit (6ES7 232-4HD32-0XB0)

Pin	X10 (gold)	X11 (gold)	X12 (gold)	X13 (gold)
1	L+ / 24 VDC	No connection	No connection	No connection
2	M / 24 VDC	No connection	No connection	No connection
3	Functional Earth	No connection	No connection	No connection
4	No connection	No connection	AQ 0M	AQ 2M
5	No connection	No connection	AQ 0	AQ 2

Pin	X10 (gold)	X11 (gold)	X12 (gold)	X13 (gold)
6	No connection	No connection	AQ 1M	AQ 3M
7	No connection	No connection	AQ 1	AIQ 3

A.8.3 SM 1234 analog input/output module specifications

Table A- 152 General specifications

Technical data	SM 1234 AI 4 x 13 bit / AQ 2 x 14 bit
Article number	6ES7 234-4HE32-0XB0
Dimensions W x H x D (mm)	45 x 100 x 75
Weight	220 grams
Power dissipation	2.4 W
Current consumption (SM Bus)	80 mA
Current consumption (24 VDC)	60 mA (no load)

Table A- 153 Analog inputs

Model	SM 1234 AI 4 x 13 bit / AQ 2 x 14 bit
Number of inputs	4
Type	Voltage or Current (differential): Selectable in groups of 2
Range	±10 V, ±5 V, ±2.5 V, 0 to 20 mA, or 4 mA to 20 mA
Full scale range (data word)	-27648 to 27648
Overshoot/undershoot range (data word)	Voltage: 32511 to 27649 / -27649 to -32512 Current: 32511 to 27649 / 0 to -4864 Refer to the section on input ranges for voltage and current (Page 1194).
Overflow/underflow (data word)	Voltage: 32767 to 32512 / -32513 to -32768 Current: 32767 to 32512 / -4865 to -32768 Refer to the section on input ranges for voltage and current (Page 1194).
Resolution	12 bits + sign bit
Maximum withstand voltage/current	±35 V / ±40 mA
Smoothing	None, weak, medium, or strong Refer to the section on step response times (Page 1194).
Noise rejection	400, 60, 50, or 10 Hz Refer to the section on sample rates (Page 1194).
Input impedance	≥ 9 MΩ (voltage) / 280 Ω (current)
Isolation (field side to logic)	None
Accuracy (25 °C / -20 to 60 °C)	±0.1% / ±0.2% of full scale
Analog to digital conversion time	625 μs (400 Hz rejection)
Common mode rejection	40 dB, DC to 60 Hz

Model	SM 1234 AI 4 x 13 bit / AQ 2 x 14 bit
Operational signal range[1]	Signal plus common mode voltage must be less than +12 V and greater than -12 V
Cable length (meters)	100 m, twisted and shielded

[1] Voltages outside the operational range applied to one channel may cause interference on other channels.

Table A- 154 Analog outputs

Technical data	SM 1234 AI 4 x 13 bit / AQ 2 x 14 bit
Number of outputs	2
Type	Voltage or current
Range	±10 V or 0 to 20 mA
Resolution	Voltage: 14 bits ; Current: 13 bits
Full scale range (data word)	Voltage: -27648 to 27648; Current: 0 to 27648 Refer to the section on output ranges for voltage and current (Page 1195).
Accuracy (25 °C / -20 to 60 °C)	±0.3% / ±0.6% 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
Behavior on RUN to STOP	Last value or substitute value (default value 0)
Isolation (field side to logic)	none
Cable length (meters)	100 m twisted and shielded

Table A- 155 Diagnostics

Model	SM 1234 AI 4 x 13 bit / AQ 2 x 14 bit
Overflow/underflow	Yes
Short to ground (voltage mode only)	Yes on outputs
Wire break (current mode only)	Yes on outputs
24 VDC low voltage	Yes

Table A- 156 Wiring diagrams for the analog input/output SM

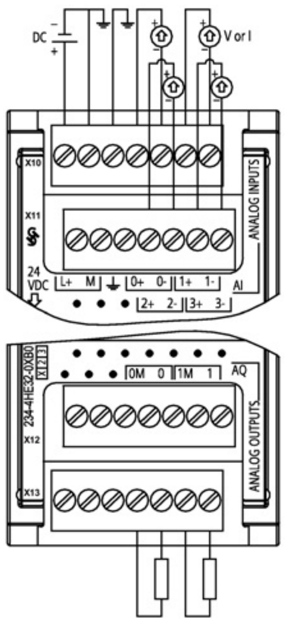
SM 1234 AI 4 x 13 Bit / AQ 2 x 14 bit (6ES7 234-4HE32-0XB0)	
 DC V or I X10 X11 X12 X13 ANALOG INPUTS ANALOG OUTPUTS 24 VDC L+ M ⏚ 0+ 0- 1+ 1- AI 2+ 2- 3+ 3- 0M 0 1M 1 AQ 234-4HE32-0XB0	
Note: Connectors must be gold. See Appendix C, Spare Parts for article number.	

Table A- 157 Connector pin locations for SM 1234 AI 4 x 13 Bit / AQ 2 x 14 bit (6ES7 234-4HE32-0XB0)

Pin	X10 (gold)	X11 (gold)	X12 (gold)	X13 (gold)
1	L+ / 24 VDC	No connection	No connection	No connection
2	M / 24 VDC	No connection	No connection	No connection
3	Functional Earth	No connection	No connection	No connection
4	AI 0+	AI 2+	No connection	AQ 0M
5	AI 0-	AI 2-	No connection	AQ 0
6	AI 1+	AI 3+	No connection	AQ 1M
7	AI 1-	AI 3-	No connection	AQ 1

Note

Unused voltage input channels should be shorted.

Unused curent input channels should be set to the 0 to 20 mA range and/or disable broken wire error reporting.

Inputs configured for current mode will not conduct loop current unless the module is powered and configured.

Current input channels will not operate unless external power is supplied to the transmitter.

A.8.4 Step response of the analog inputs

Table A- 158 Step response (ms), 0 to full-scale measured at 95%

Smoothing selection (sample averaging)	Noise reduction/rejection frequency (Integration time selection)			
	400 Hz (2.5 ms)	60 Hz (16.6 ms)	50 Hz (20 ms)	10 Hz (100 ms)
None (1 cycle): No averaging	4 ms	18 ms	22 ms	100 ms
Weak (4 cycles): 4 samples	9 ms	52 ms	63 ms	320 ms
Medium (16 cycles): 16 samples	32 ms	203 ms	241 ms	1200 ms
Strong (32 cycles): 32 samples	61 ms	400 ms	483 ms	2410 ms
Sample time • 4 AI x 13 bits • 8 AI x 13 bits • 4 AI4 x 16 bits	 • 0.625 ms • 1.25 ms • 0.417 ms	 • 4.17 ms • 4.17 ms • 0.397 ms	 • 5 ms • 5 ms • 0.400 ms	 • 25 ms • 25 ms • 0.400 ms

A.8.5 Sample time and update times for the analog inputs

Table A- 159 Sample time and update time

Rejection frequency (Integration time)	Sample time	Module update time for all channels	
		4-channel SM	8-channel SM
400 Hz (2.5 ms)	• 4-channel SM: 0.625 ms • 8-channel SM: 1.250 ms	0.625 ms	1.250 ms
60 Hz (16.6 ms)	4.170 ms	4.17 ms	4.17 ms
50 Hz (20 ms)	5.000 ms	5 ms	5 ms
10 Hz (100 ms)	25.000 ms	25 ms	25 ms

A.8.6 Measurement ranges of the analog inputs for voltage and current (SB and SM)

Table A- 160 Analog input representation for voltage (SB and SM)

System		Voltage Measuring Range				
Decimal	Hexadecimal	±10 V	±5 V	±2.5 V	±1.25 V	
32767	7FFF[1]	11.851 V	5.926 V	2.963 V	1.481 V	Overflow
32512	7F00					
32511	7EFF	11.759 V	5.879 V	2.940 V	1.470 V	Overshoot range
27649	6C01					
27648	6C00	10 V	5 V	2.5 V	1.250 V	Rated range
20736	5100	7.5 V	3.75 V	1.875 V	0.938 V	
1	1	361.7 μV	180.8 μV	90.4 μV	45.2 μV	

System		Voltage Measuring Range				
Decimal	**Hexadecimal**	**±10 V**	**±5 V**	**±2.5 V**	**±1.25 V**	
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- 161 Analog input representation for current (SB and SM)

System		Current measuring range		
Decimal	**Hexadecimal**	**0 mA to 20 mA**	**4 mA to 20 mA**	
32767	7FFF	23.70 mA	22.96 mA	Overflow
32512	7F00			
32511	7EFF	23.52 mA	22.81 mA	Overshoot range
27649	6C01			
27648	6C00	20 mA	20 mA	Nominal range
20736	5100	15 mA	16 mA	
1	1	723.4 nA	4 mA + 578.7 nA	
0	0	0 mA	4 mA	
-1	FFFF			Undershoot range
-4864	ED00	-3.52 mA	1.185 mA	
-4865	ECFF			Underflow
-32768	8000			

A.8.7 Measurement ranges of the analog outputs for voltage and current (SB and SM)

Table A- 162 Analog output representation for voltage (SB and SM)

System		Voltage Output Range	
Decimal	**Hexadecimal**	**± 10 V**	
32767	7FFF	See note 1	Overflow
32512	7F00	See note 1	
32511	7EFF	11.76 V	Overshoot range
27649	6C01		
27648	6C00	10 V	Rated range
20736	5100	7.5 V	

System		Voltage Output Range	
Decimal	**Hexadecimal**	**± 10 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- 163 Analog output representation for current (SB and SM)

System		Current output range		
Decimal	**Hexadecimal**	**0 mA to 20 mA**	**4 mA to 20 mA**	
32767	7FFF	See note 1	See note 1	Overflow
32512	7F00	See note 1	See note 1	
32511	7EFF	23.52 mA	22.81 mA	Overshoot range
27649	6C01			
27648	6C00	20 mA	20 mA	Rated range
20736	5100	15 mA	16 mA	
1	1	723.4 nA	4 mA + 578.7 nA	
0	0	0 mA	4mA	
-1	FFFF		4 mA to 578.7 nA	Undershoot range
-6912	E500		0 mA	
-6913	E4FF			Not possible. Output value limited to 0 mA.
-32512	8100			
-32513	80FF	See note 1	See note 1	Underflow
-32768	8000	See note 1	See note 1	

1 In an overflow or underflow condition, analog outputs will take on the substitute value of the STOP mode.

A.9 Thermocouple and RTD signal modules (SMs)

A.9.1 SM 1231 Thermocouple

Table A- 164 General specifications

Model	SM 1231 AI 4 x 16 bit TC	SM 1231 AI 8 x 16 bit TC
Article number	6ES7 231-5QD32-0XB0	6ES7 231-5QF32-0XB0
Dimensions W x H x D (mm)	45 x 100 x 75	
Weight	180 grams	190 grams
Power dissipation	1.5 W	
Current consumption (SM Bus)	80 mA	
Current consumption (24 VDC) [1]	40 mA	

[1] 20.4 to 28.8 VDC (Class 2, Limited Power, or sensor power from PLC)

Table A- 165 Analog inputs

Model		SM 1231 AI 4 x 16 bit TC	SM 1231 AI 8 x 16 bit TC
Number of inputs		4	8
Range Nominal range (data word) Overrange/underrange (data word) Overflow/underflow (data word)		See Thermocouple selection table (Page 1200).	
Resolution	Temperature	0.1 °C/0.1 °F	
	Voltage	15 bits plus sign	
Maximum withstand voltage		± 35 V	
Noise rejection		85 dB for selected filter setting (10 Hz, 50 Hz, 60 Hz or 400 Hz)	
Common mode rejection		> 120 dB at 120 VAC	
Impedance		≥ 10 MΩ	
Isolation	Field to logic	500 VAC	
	Field to 24 VDC	500 VAC	
	24 VDC to logic	500 VAC	
Channel to channel		120 VAC	
Accuracy		See Thermocouple selection table (Page 1200).	
Repeatability		±0.05% FS	
Measuring principle		Integrating	
Module update time		See Noise reduction selection table (Page 1200).	
Cold junction error		±1.5 °C	
Cable length (meters)		100 meters to sensor max.	
Wire resistance		100 Ω max.	

Table A- 166 Diagnostics

Model	SM 1231 AI 4 x 16 bit TC	SM 1231 AI 8 x 16 bit TC
Overflow/underflow [1]	Yes	
Wire break (current mode only) [2]	Yes	
24 VDC low voltage [1]	Yes	

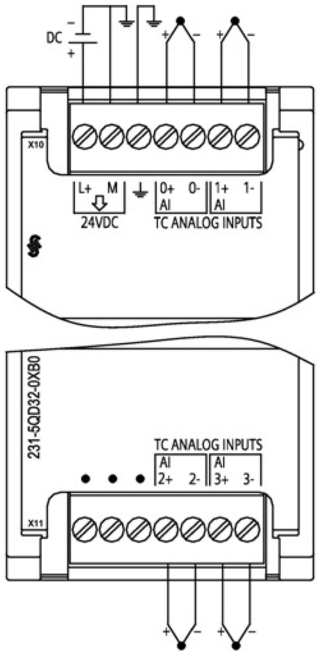
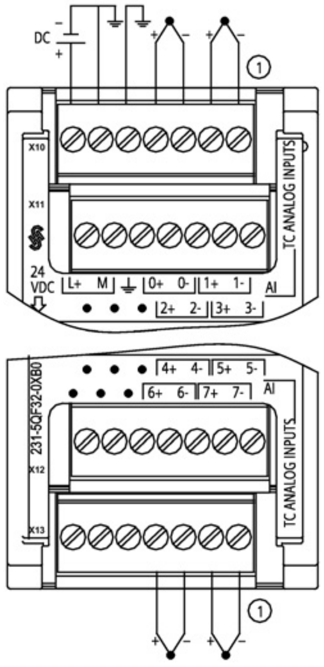
[1] The overflow, underflow and low voltage diagnostic alarm information will be reported in the analog data values even if the alarms are disabled in the module configuration.

[2] When wire break alarm is disabled and an open wire condition exists in the sensor wiring, the module may report random values.

The SM 1231 Thermocouple (TC) analog signal module measures the value of voltage connected to the module inputs. The temperature measurement type can be either "Thermocouple" or "Voltage".

- "Thermocouple": The value will be reported in degrees multiplied by ten (for example, 25.3 degrees will be reported as decimal 253).
- "Voltage": The nominal range full scale value will be decimal 27648.

Table A- 167 Wiring diagrams for the thermocouple SMs

SM 1231 AI 4 x TC 16 bit (6ES7 231-5QD32-0XB0)	SM 1231 AI 8 x TC bit (6ES7 231-5QF32-0XB0)
 DC X10 L+ M 24VDC 0+ 0- AI 1+ 1- AI TC ANALOG INPUTS 231-5QD32-0XB0 TC ANALOG INPUTS AI 2+ 2- AI 3+ 3- X11	 DC ① X10 X11 24 VDC L+ M 0+ 0- 1+ 1- 2+ 2- 3+ 3- AI TC ANALOG INPUTS 231-5QF32-0XB0 4+ 4- 5+ 5- 6+ 6- 7+ 7- AI X12 X13 TC ANALOG INPUTS ①
Note: Connectors must be gold. See Appendix C, Spare Parts for article number.	

① TC 2, 3, 4, and 5 not shown connected for clarity.

Table A- 168 Connector pin locations for SM 1231 AI 4 x TC 16 bit (6ES7 231-5QD32-0XB0)

Pin	X10 (gold)	X11 (gold)
1	L+ / 24 VDC	No connection
2	M / 24 VDC	No connection
3	Functional Earth	No connection
4	AI 0+ /TC	AI 2+ /TC
5	AI 0- /TC	AI 2- /TC
6	AI 1+ /TC	AI 3+ /TC
7	AI 1- /TC	AI 3- /TC

Table A- 169 Connector Pin Locations for SM 1231 AI 8 x TC bit (6ES7 231-5QF32-0XB0)

Pin	X10 (gold)	X11 (gold)	X12 (gold)	X13 (gold)
1	L+ / 24 VDC	No connection	No connection	No connection
2	M / 24 VDC	No connection	No connection	No connection
3	Functional Earth	No connection	No connection	No connection
4	AI 0+ /TC	AI 2+ /TC	AI 4 I- /TC	AI 6 I- /TC
5	AI 0- /TC	AI 2- /TC	AI 4 I+ /TC	AI 6 I+ /TC
6	AI 1+ /TC	AI 3+ /TC	AI 5 M- /TC	AI 7 M- /TC
7	AI 1- /TC	AI 3- /TC	AI 5 M+ /TC	AI 7 M+ /TC

Note

Unused analog inputs should be shorted.

The thermocouple unused channels can be deactivated. No error will occur if an unused channel is deactivated.

A.9.1.1 Basic operation for 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 SM 1231 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.

A.9.1.2 Selection tables for the SM 1231 thermocouple

The ranges and accuracy for the different thermocouple types supported by the SM 1231 Thermocouple signal module are shown in the table below.

Table A- 170 SM 1231 Thermocouple selection table

Type	Under-range minimum[1]	Nominal range low limit	Nominal range high limit	Over-range maximum[2]	Normal range [3, 4] accuracy @ 25 °C	Normal range [1, 2] accuracy -20 °C to 60 °C
J	-210.0 °C	-150.0 °C	1200.0 °C	1450.0 °C	±0.3 °C	±0.6 °C
K	-270.0 °C	-200.0 °C	1372.0 °C	1622.0 °C	±0.4 °C	±1.0 °C
T	-270.0 °C	-200.0 °C	400.0 °C	540.0 °C	±0.5 °C	±1.0 °C
E	-270.0 °C	-200.0 °C	1000.0 °C	1200.0 °C	±0.3 °C	±0.6 °C
R & S	-50.0 °C	100.0 °C	1768.0 °C	2019.0 °C	±1.0 °C	±2.5 °C
B	0.0 °C	200.0 °C	800.0 °C	--	±2.0 °C	±2.5 °C
	--	800.0 °C	1820.0 °C	1820.0 °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	1050.0 °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 minimum value are reported as 32767.

[3] Internal cold junction error is ±1.5 °C for all ranges. This adds to the error in this table. The module requires at least 30 minutes of warm-up time to meet this specification.

[4] In the presence of radiated radio frequency of 970 MHz to 990 MHz, the accuracy of the SM 1231 AI 4 x 16 bit TC may be degraded.

Note

Thermocouple channel

Each channel on the Thermocouple signal module can be configured with a different thermocouple type (selectable in the software during configuration of the module).

Table A- 171 Noise reduction and update times for the SM 1231 Thermocouple

Rejection frequency selection	Integration time	4 Channel module update time (seconds)	8 Channel module update time (seconds)
400 Hz (2.5 ms)	10 ms [1]	0.143	0.285
60 Hz (16.6 ms)	16.67 ms	0.223	0.445
50 Hz (20 ms)	20 ms	0.263	0.525
10 Hz (100 ms)	100 ms	1.225	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 of the module in your configuration. If required, you can include logic in your user program to accommodate the initialization time of the module.

Representation of analog values for Thermocouple Type J

A representation of the analog values of thermocouples type J is shown in the table below.

Table A- 172 Representation of analog values of thermocouples type J

Type J in °C	Units		Type J in °F	Units		Range
	Decimal	Hexadecimal		Decimal	Hexadecimal	
> 1450.0	32767	7FFF	> 2642.0	32767	7FFF	Overflow
1450.0 : 1200.1	14500 : 12001	38A4 : 2EE1	2642.0 : 2192.2	26420 : 21922	6734 : 55A2	Overrange
1200.0 : -150.0	12000 : -1500	2EE0 : FA24	2192.0 : -238.0	21920 : -2380	55A0 : F6B4	Rated range
< -150.0	-32768	8000	< -238.0	-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.9.2 SM 1231 RTD

SM 1231 RTD specifications

Table A- 173 General specifications

Technical data	SM 1231 AI 4 x RTD x 16 bit	SM 1231 AI 8 x RTD x 16 bit
Article number	6ES7 231-5PD32-0XB0	6ES7 231-5PF32-0XB0
Dimensions W x H x D (mm)	45 x 100 x 75	70 x 100 x 75
Weight	220 grams	270 grams
Power dissipation	1.5 W	
Current consumption (SM Bus)	80 mA	90 mA
Current consumption (24 VDC) [1]	40 mA	

[1] 20.4 to 28.8 VDC (Class 2, Limited Power, or sensor power from CPU)

Table A- 174 Analog inputs

Technical data	SM 1231 AI 4 x RTD x 16 bit	SM 1231 AI 8 x RTD x16 bit
Number of inputs	4	8
Type	Module referenced RTD and Ω	

Technical data		SM 1231 AI 4 x RTD x 16 bit	SM 1231 AI 8 x RTD x16 bit
Range Nominal range (data word) Overshoot/undershoot range (data word) Overflow/underflow (data word)		See RTD Sensor selection table (Page 1205).	
Resolution	Temperature	0.1 °C/0.1 °F	
	Resistance	15 bits plus sign	
Maximum withstand voltage		± 35 V	
Noise rejection		85 dB for the selected noise reduction (10 Hz, 50 Hz, 60 Hz or 400 Hz)	
Common mode rejection		> 120dB	
Impedance		≥ 10 MΩ	
Isolation	Field side to logic	500 VAC	
	Field to 2 VDC	500 VAC	
	24 VDC to logic	500 VAC	
Channel to channel isolation		none	
Accuracy		See RTD Sensor selection table (Page 1205).	
Repeatability		±0.05% FS	
Maximum sensor dissipation		0.5 m W	
Measuring principle		Integrating	
Module update time		See Noise reduction selection table (Page 1205).	
Cable length (meters)		100 meters to sensor max.	
Wire resistance		20 Ω, 2.7 Ω for 10 Ω RTD max.	

Table A- 175 Diagnostics

Technical data	SM 1231 AI 4 x RTD x 16 bit	SM 1231 AI 8 x RTD x16 bit
Overflow/underflow [1,2]	Yes	
Wire break [3]	Yes	
24 VDC low voltage [1]	Yes	

[1] The overflow, underflow and low voltage diagnostic alarm information will be reported in the analog data values even if the alarms are disabled in the module configuration.

[2] For resistance ranges underflow detection is never enabled.

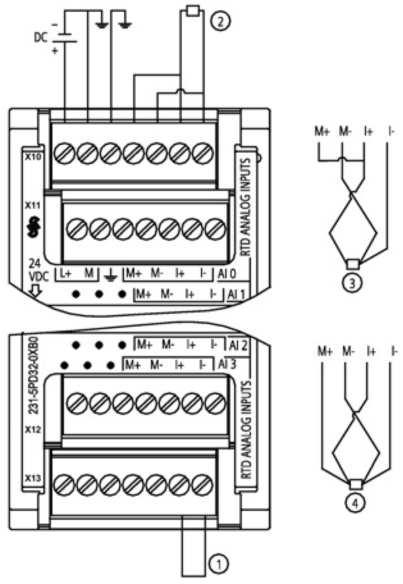
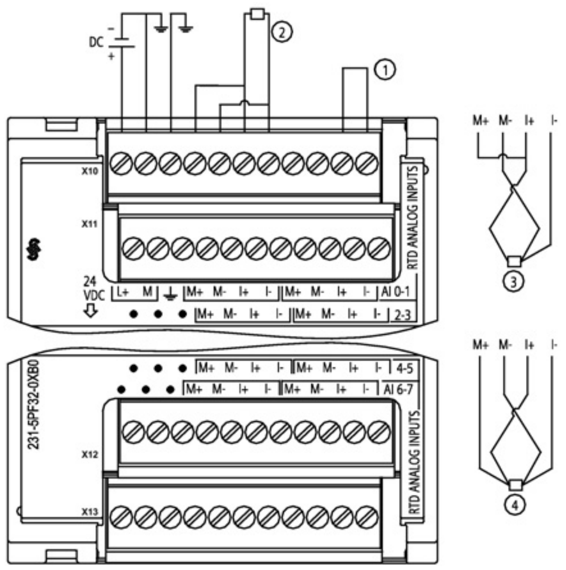
[3] When wire break alarm is disabled and an open wire condition exists in the sensor wiring, the module may report random values.

The SM 1231 RTD analog signal 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 climatic range values will be reported in degrees multiplied by one hundred (for example, 25.34 degrees will be reported as decimal 2534).

The SM 1231 RTD module supports measurements with 2-wire, 3-wire and 4-wire connections to the sensor resistor.

Table A- 176 Wiring diagrams for the RTD SMs

SM 1231 RTD 4 x 16 bit (6ES7 231-5PD32-0XB0)	SM 1231 RTD 8 x 16 bit (6ES7 231-5PF32-0XB0)
	

① Loop-back unused RTD inputs
② 2-wire RTD ③ 3-wire RTD ④ 4-wire RTD
NOTE: Connectors must be gold. See Appendix C, Spare Parts for article number.

Table A- 177 Connector Pin Locations for SM 1231 RTD 4 x 16 bit (6ES7 231-5PD32-0XB0)

Pin	X10 (gold)	X11 (gold)	X12 (gold)	X13 (gold)
1	L+ / 24 VDC	No connection	No connection	No connection
2	M / 24 VDC	No connection	No connection	No connection
3	Functional Earth	No connection	No connection	No connection
4	AI 0 M+ /RTD	AI 1 M+ /RTD	AI 2 M+ /RTD	AI 3 M+ /RTD
5	AI 0 M- /RTD	AI 1 M- /RTD	AI 2 M- /RTD	AI 3 M- /RTD
6	AI 0 I+ /RTD	AI 1 I+ /RTD	AI 2 I+ /RTD	AI 3 I+ /RTD
7	AI 0 I- /RTD	AI 1 I- /RTD	AI 2 I- /RTD	AI 3 I- /RTD

Table A- 178 Connector Pin Locations for SM 1231 RTD 8 x 16 bit (6ES7 231-5PF32-0XB0)

Pin	X10 (gold)	X11 (gold)	X12 (gold)	X13 (gold)
1	L+ / 24 VDC	No connection	No connection	No connection
2	M / 24 VDC	No connection	No connection	No connection
3	Functional Earth	No connection	No connection	No connection

Pin	X10 (gold)	X11 (gold)	X12 (gold)	X13 (gold)
4	AI 0 M+ /RTD	AI 2 M+ /RTD	AI 4 M+ /RTD	AI 6 M+ /RTD
5	AI 0 M- /RTD	AI 2 M- /RTD	AI 4 M- /RTD	AI 6 M- /RTD
6	AI 0 I+ /RTD	AI 2 I+ /RTD	AI 4 I+ /RTD	AI 6 I+ /RTD
7	AI 0 I- /RTD	AI 2 I- /RTD	AI 4 I- /RTD	AI 6 I- /RTD
8	AI 1 M+ /RTD	AI 3 M+ /RTD	AI 5 M+ /RTD	A7 M+ /RTD
9	AI 1 M- /RTD	AI 3 M- /RTD	AI 5 M- /RTD	AI 7 M- /RTD
10	AI 1 I+ /RTD	AI 3 I+ /RTD	AI 5 I+ /RTD	AI 7 I+ /RTD
11	AI 1 I- /RTD	AI 3 I- /RTD	AI 5 I- /RTD	AI 7 I- /RTD

Note

The RTD unused channels can be deactivated. No error will occur if an unused channel is deactivated.

The RTD module needs to have the current loop continuous to eliminate extra stabilization time which is automatically added to an unused channel that is not deactivated. For consistency the RTD module should have a resistor connected (like the 2-wire RTD connection).

A.9.2.1 Selection tables for the SM 1231 RTD

Table A- 179 Ranges and accuracy for the different sensors supported by the RTD modules

Temperature coefficient	RTD type	Under range minimum[1]	Nominal range low limit	Nominal range high limit	Over range maximum[2]	Normal range accuracy @ 25 °C	Normal range accuracy -20 °C to 60 °C
Pt 0.003850 ITS90 DIN EN 60751	Pt 100 climatic	-145.00 °C	-120.00 °C	145.00 °C	155.00 °C	±0.20 °C	±0.40 °C
	Pt 10	-243.0 °C	-200.0 °C	850.0 °C	1000.0 °C	±1.0 °C	±2.0 °C
	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

Temperature coefficient	RTD type	Under range minimum[1]	Nominal range low limit	Nominal range high limit	Over range maximum[2]	Normal range accuracy @ 25 °C	Normal range accuracy -20 °C to 60 °C
	Pt 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
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- 180 Resistance

Range	Under range minimum	Nominal range low limit	Nominal range high limit	Over range maximum[1]	Normal range accuracy @ 25 °C	Normal range accuracy -20 °C to 60 °C
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%

[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 500 Ω and 1000 Ω RTD ranges are used with other lower value resistors, the error may increase to two times the specified error.

Best accuracy will be achieved for the 10 Ω RTD ranges if 4 wire connections are used.

The resistance of the connection wires in 2 wire mode will cause an error in the sensor reading and therefore accuracy is not guaranteed.

Table A- 181 Noise reduction and update times for the RTD modules

Rejection frequency selection	Integration time	Update time (seconds)	
		4-channel module	8-channel module
400 Hz (2.5 ms)	10 ms [1]	4-/2-wire: 0.142 3-wire: 0.285	4-/2-wire: 0.285 3-wire: 0.525
60 Hz (16.6 ms)	16.67 ms	4-/2-wire: 0.222 3-wire: 0.445	4-/2-wire: 0.445 3-wire: 0.845
50 Hz (20 ms)	20 ms	4-/2-wire: 0.262 3-wire: .505	4-/2-wire: 0.524 3-wire: 1.015
10 Hz (100 ms)	100 ms	4-/2-wire: 1.222 3-wire: 2.445	4-/2-wire: 2.425 3-wire: 4.845

[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.

Representation of Analog values for RTDs

A representation of the digitized measured value for the RTD standard temperature range sensors are shown in the tables below.

Table A- 182 Representation of analog values for resistance thermometers PT 100, 200, 500, 1000 and PT 10, 50, 100, 500 GOST (0.003850) standard

Pt x00 standard in °C (1 digit = 0.1 °C)	Units		Pt x00 standard in °F (1 digit = 0.1 °F)	Units		Range
	Decimal	Hexadecimal		Decimal	Hexadecimal	
> 1000.0	32767	7FFF	> 1832.0	32767	7FFF	Overflow
1000.0 : 850.1	10000 : 8501	2710 : 2135	1832.0 : 1562.1	18320 : 15621	4790 : 3D05	Overrange
850.0 : -200.0	8500 : -2000	2134 : F830	1562.0 : -328.0	15620 : -3280	3D04 : F330	Rated range

Pt x00 standard in °C (1 digit = 0.1 °C)	Units		Pt x00 standard in °F (1 digit = 0.1 °F)	Units		Range
	Decimal	Hexadecimal		Decimal	Hexadecimal	
-200.1 : -243.0	-2001 : -2430	F82F : F682	-328.1 : -405.4	-3281 : -4054	F32F : F02A	Underrange
< -243.0	-32768	8000	< -405.4	-32768	8000	Underflow

A.10 Technology modules

A.10.1 SM 1278 4xIO-Link Master SM

Table A- 183 General specifications

Technical data		SM 1278 4xIO-Link Master signal module
Article number		6ES7 278-4BD32-0XB0
Dimensions W x H x D (mm)		45 x 100 x 75
Weight		150 grams
General information		
	I&M data	Yes; IM0 to IM3
Supply voltage		
	Rated voltage (DC)	24 VDC
	Valid range low limit (DC)	19.2 V; 20.5 V if IO-Link is used (the supply voltage for IO-Link devices on the master must be at least 20 V)
	Valid range high limit (DC)	28.8 VDC
	Polarity reversal protection	Yes
Input current		
	Current consumption	65 mA; without load
Encoder supply		
	Number of outputs	4
	Output current, rated value	200 mA
Power loss		
	Power loss, typ.	1 W, excluding port loading
Digital inputs/outputs		
	Cable length (meters)	20 m, unshielded, max.
SDLC		
	Cable length (meters)	20 m, unshielded, max.
IO-Link		
	Number of ports	4

Technical data			SM 1278 4xIO-Link Master signal module
	Number of ports which can be controlled at the same time		4
	IO-Link protocol 1.0		Yes
	IO-Link protocol 1.1		Yes
Operating mode			
	IO-Link		Yes
	DI		Yes
	DQ		Yes; max. 100 mA
Connection of IO-Link devices			
	Port type A		Yes
		Transmission rate	4.8 kBd (COM1)
			38.4 kBd (COM2)
			230.4 kBd (COM3)
	Cycle time, min.		2 ms, dynamic, dependent on the user data length
	Size of process data, input per port		32 bytes; max.
	Size of process data, input per module		32 bytes
	Size of process data, output per port		32 bytes; max.
	Size of process data, output per module		32 bytes
	Memory size for device parameters		2 Kbytes
	Cable length unshielded, max. (meters)		20 m
Interrupts/diagnostics/status information			
	Status display		Yes
Interrupts			
	Diagnostic interrupt		Yes; port diagnostics is only available in IO-Link mode
Diagnostic alarms			
	Diagnostics		
		Monitoring of supply voltage	Yes
		Short circuit	Yes
Diagostic indicator LED			
	Monitoring of supply voltage		Yes; flashing red DIAG LED
	Channel status display		Yes; per channel one green LED for channel status Qn (SIO mode) and PORT status Cn (IO-Link mode)
	For channel diagnostics		Yes; red Fn LED
	For module diagnostics		Yes; green/red DIAG LED
Electrical isolation			
	Electrical isolation channels		
		Between the channels	No
		Between the channels and the backplane bus	Yes
Permitted potential difference			
	Between the different circuits		75 VDC / 60 VAC (basic insulation)
Insulation			
	Insulation tested with		707 VDC (type test)

Technical data			SM 1278 4xIO-Link Master signal module
Ambient conditions			
	Operating temperature		
		Min.	-20 °C
		Max.	60 °C
		Horizontal installation, min.	-20 °C
		Horizontal installation, max.	60 °C
		Vertical installation, min.	-20 °C
		Vertical installation, max.	50 °C

Overview of the response time

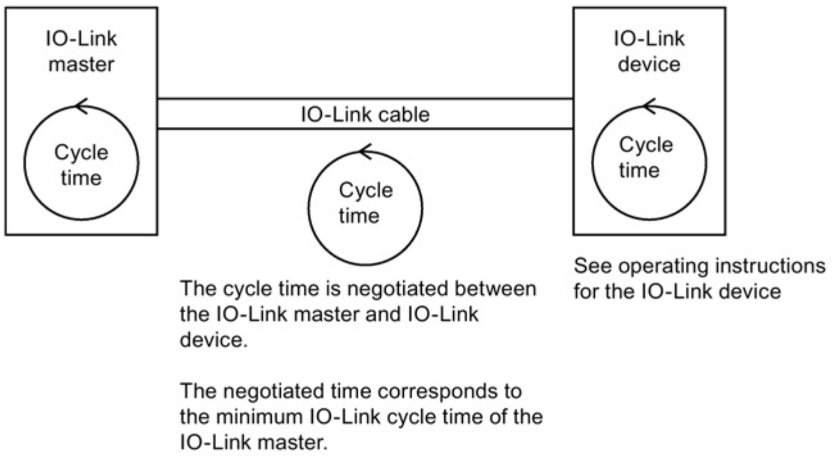

IO-Link master
Cycle time
IO-Link cable
Cycle time
IO-Link device
Cycle time
The cycle time is negotiated between the IO-Link master and IO-Link device.
The negotiated time corresponds to the minimum IO-Link cycle time of the IO-Link master.
See operating instructions for the IO-Link device

Table A- 184 Wiring diagram for the SM 1278 IO-Link Master

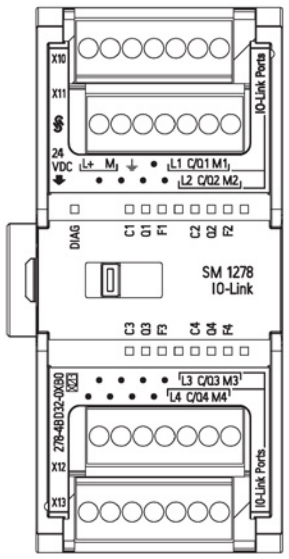
SM 1278 IO-Link Master (6ES7 278-4BD32-0XB0)	
 X10 X11 24 VDC L+ M L1 C/Q1 M1 L2 C/Q2 M2 IO-Link Ports DIAG C1 Q1 F1 C2 Q2 F2 SM 1278 IO-Link C3 Q3 F3 C4 Q4 F4 L3 C/Q3 M3 L4 C/Q4 M4 278-4BD32-0XB0 X12 X13	

Table A- 185 Connector pin locations for SM 1278 IO-Link Master (6ES7 278-4BD32-0XB0)

Pin	X10	X11	X12	X13
1	L+ / 24 VDC	No connection	No connection	No connection
2	M / 24 VDC	No connection	No connection	No connection
3	Functional Earth	No connection	No connection	No connection
4	No connection	No connection	No connection	No connection
5	L_1	L_2	L_3	L_4
6	C/Q_1	C/QL_2	C/Q_3	C/QL_4
7	ML_1	ML_2	M_3	ML_4

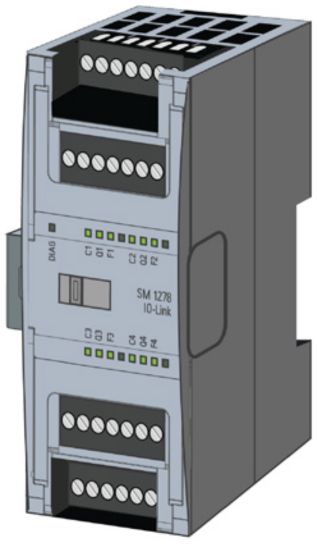
A.10.1.1 SM 1278 4xIO-Link Master overview

The SM 1278 4xIO-Link Master is a 4-port module that functions as both a signal module and a communication module. Each port can operate in the IO-Link mode, single 24 VDC digital input or 24 VDC digital output.

The IO-Link master programs acyclic communication with an IO-Link device using the IOL_CALL function block (FB) in your STEP 7 S7-1200 controller program. The IOL_CALL FB indicates the IO-Link master your program uses, and which ports the master uses for data exchange.

Visit the Siemens Industry Online Support website (http://support.automation.siemens.com) for details on working with the IOL_CALL FB. Enter "IO-Link" in the website's search box to access information about IO-Link products and their use.

View of the module



Properties

Technical properties

- IO-Link Master according to IO-Link specification V1.1 (see the IO-Link Consortium website (http://io-link.com/en/index.php) for details)
- Serial communication module with four ports (channels)
- Data transmission rate COM1 (4.8 kbaud), COM2 (38.4 kbaud), COM3 (230.4 kbaud)
- SIO mode (standard IO mode)
- Connection of up to four IO-Link devices (3-wire connection) or four standard actuators or standard encoders
- Programmable diagnostics function by port

Supported functions

- I&M (installation and maintenance) identification data
- Firmware update
- IO-Link parameter assignment by means of the S7-PCT port configuration tool, STEP 7 Professional, and an S7-1200 V4.0 or higher CPU

IO-Link is a point-to-point connection between a master and a device. Both conventional and intelligent sensors/actuators can be used as devices at the IO-Link via unshielded standard cables using proven 3-wire technology. IO-Link is backward compatible with conventional digital sensors and actuators. The circuit state and data channel are designed in proven 24 VDC technology.

For additional information about the SIMATIC IO-Link technology, refer to the "IO-Link system Function Manual" on the Siemens Industry Online Support website (http://support.automation.siemens.com).

Note

IO-Link parameter data

When you replace the SM 4xIO-Link Master, the parameter data is not automatically assigned to it.

CAUTION

Removal and insertion

If you insert the SM 4xIO-Link Master with the load switched on, this can lead to dangerous conditions in your plant.

Physical damage to the S7-1200 automation system may occur as a result.

Remove or insert the SM 4xIO-Link Master only when the load is switched off.

Effects of resetting to the factory settings

Use the function "Reset to factory settings" to restore the parameter assignments you made with S7-PCT to the delivery state.

After a "Reset to factory settings", the parameters of the SM 1278 4xIO-Link module are assigned as follows:

- The ports are in DI mode
- The ports are mapped to the relative addresses 0.0 to 0.3
- The PortQualifier is disabled
- Maintenance data 1 to 3 is deleted

Note

When you reset to factory settings, the device parameters are deleted and the delivery state is restored.

If you remove an SM 1278 4xIO-Link signal module, reset it to factory settings before you put it into storage.

Procedure

For "Reset to factory settings", proceed as described in the S7-PCT online help under "Master Configuration > 'Commands' tab".

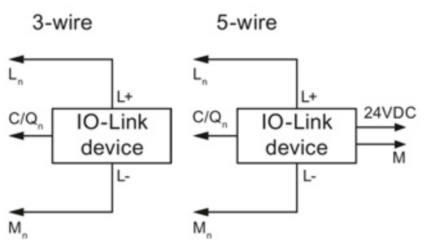
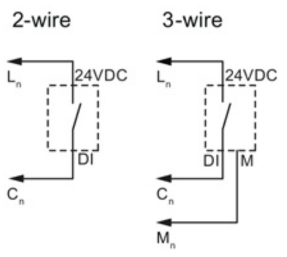
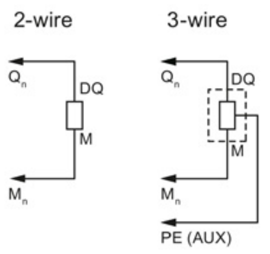
A.10.1.2 Connecting

For details about pin assignment, see table, Connector pin locations for SM 1278 I/O-Link Master (6ES 278-4BD32-0XB0). (Page 1208)

The following table shows the terminal assignments for the SM 1278 4xIO-Link Master:

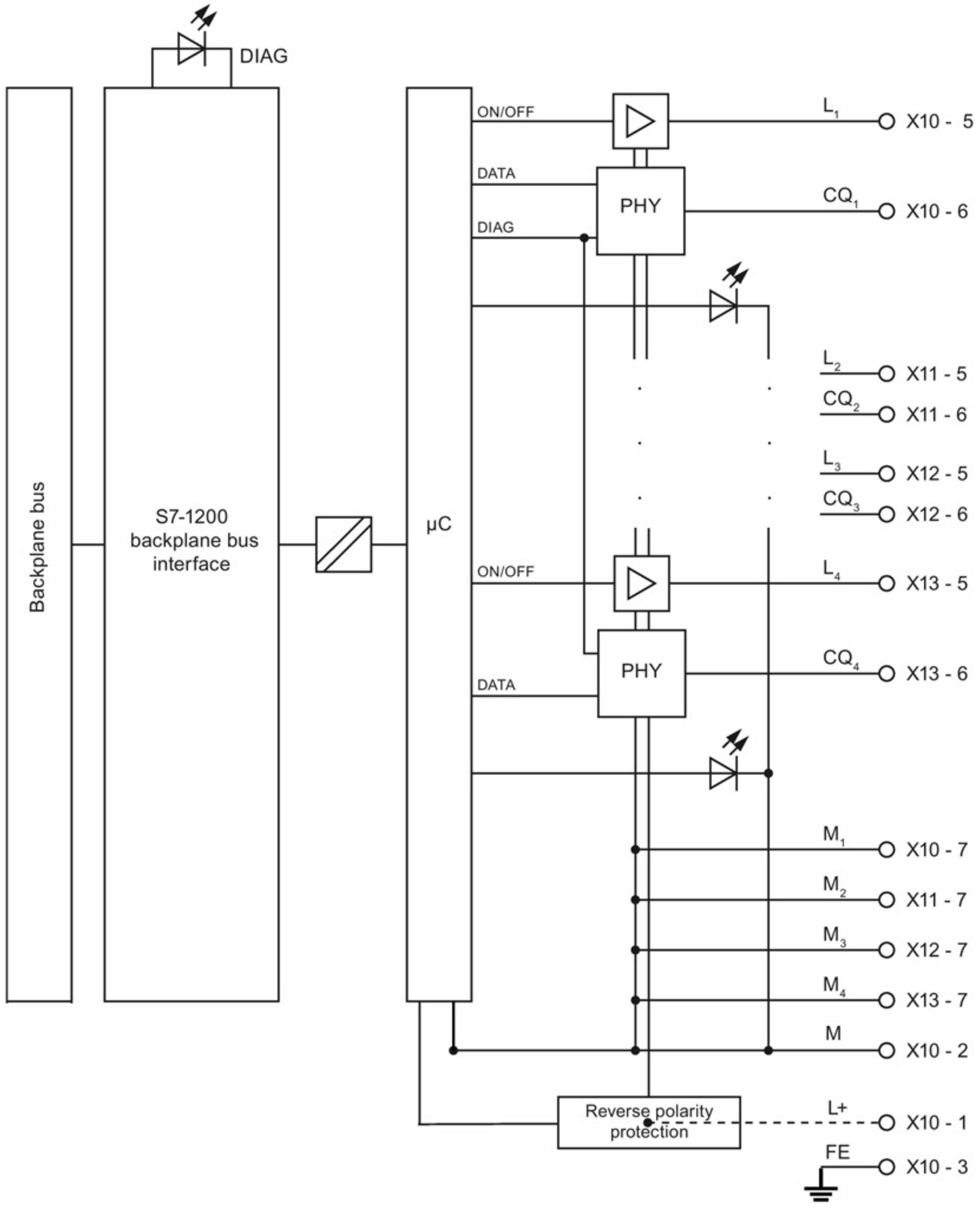
Pin	X10	X11	X12	X13	Notes	BaseUnits
7	M_1	M_2	M_3	M_4	• M_n: ground to slave • C/Q_n: SDLC, DI or DQ • L_n: 24 VDC to slave • M: ground • L+: 24 VDC to Master • RES: reserved; may not be assigned	A1
6	C/Q_1	C/Q_2	C/Q_3	C/Q_4		
5	L_1	L_2	L_3	L_4		
4	RES	RES	RES	RES		
3	⏚ (functional earth)	RES	RES	RES		
2	M	RES	RES	RES		
1	L+	RES	RES	RES		

The following table contains illustrations of connection examples, where n = port number:

IO-Link operating mode	Operating mode DI	Operating mode DQ
 3-wire 5-wire L_n L+ C/Q_n IO-Link device L- M_n 24VDC M	 2-wire 3-wire L_n 24VDC DI C_n M M_n	

Note

Connected sensors must use the device supply provided by the Master module L_n connection.


DIAG
Backplane bus
S7-1200 backplane bus interface
µC
ON/OFF
DATA
DIAG
PHY
ON/OFF
PHY
DATA
L1
X10 - 5
CQ1
X10 - 6
L2
X11 - 5
CQ2
X11 - 6
L3
X12 - 5
CQ3
X12 - 6
L4
X13 - 5
CQ4
X13 - 6
M1
X10 - 7
M2
X11 - 7
M3
X12 - 7
M4
X13 - 7
M
X10 - 2
Reverse polarity protection
L+
X10 - 1
FE
X10 - 3

A.10.1.3 Parameters/address space

Configuring the SM 1278 4xIO-Link Master

For the module integration, you need the Siemens engineering tool TIA Portal V13 or higher. You also need S7-PCT V3.2 or higher for the IO-Link integration.

For commissioning, you require both an engineering tool and S7-PCT V3.2 or higher for parameter assignment.

The following table shows the parameters for the SM 1278 4xIO-Link Master:

Parameters	Value range	Default	Configuration in RUN	Efficiency range
Diagnostics port 1	• Disable • Enable	Disable	Yes	Port (channel)
Diagnostics port 2	• Disable • Enable	Disable	Yes	Port (channel)
Diagnostics port 3	• Disable • Enable	Disable	Yes	Port (channel)
Diagnostics port 4	• Disable • Enable	Disable	Yes	Port (channel)

Enable diagnostics for port 1 to port 4 parameter

This parameter allows diagnostics to be enabled for specific ports of the four IO-Link ports.

The port assignments are as follows:

Port 1 → channel 1

Port 2 → channel 2

Port 3 → channel 3

Port 4 → channel 4

The maximum size of the input and output addresses of the SM 4xIO-Link Master is 32 bytes in each case. You assign address spaces using the S7-PCT port configuration tool.

Parameter data record

Parameter assignment in the user program

You can configure the device in runtime.

Changing parameters in runtime

The module parameters are included in data record 128. You can transmit the modifiable parameters to the module with the WRREC instruction.

When you reset (power cycle) the CPU, the CPU overwrites the parameters that were sent to the module by the WRREC instruction during the parameterization process.

Instruction for parameter assignment

The following instruction is provided for assigning parameters to the I/O module in the user program:

Instruction	Application
SFB 53 WRREC	Transfer of the alterable parameters to the module.

Error message

The following return value is reported in the event of an error:

Error code	Meaning
$80B1_H$	Error in data length
$80E0_H$	Error in header information
$80E1_H$	Parameter error

Data record structure

The following table shows the IO-Link parameters:

Offset	Label	Type	Default	Description
0	Version	1 byte	0x02	Shows the structure of the record 0x02 for the IO-Link Master in accordance with IO-Link V1.1
1	Parameter length	1 byte	0x02	Parameter length (2 bytes + 2 headers)
IO-Link start parameters				
2	Port diagnostics (Port1 1 to n)	1 byte	0x00	Activating the diagnostics for port 1 to n
3	IOL properties	1 byte	0x00	Module properties

The following table shows the data record version:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved		Major version (00)		Minor version (0010)			

The following table shows the data record port diagnostics:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved				EN_Port4	EN_Port3	EN_Port2	EN_Port1

EN_Portx:

0 = Diagnostics deactivated

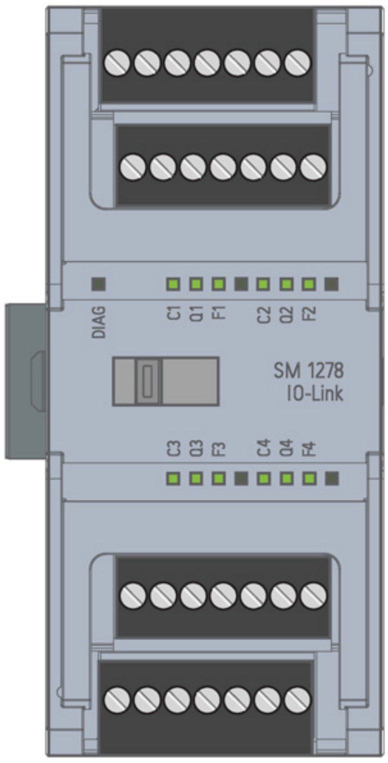
1 = Diagnostics activated

The following table shows the data record IOL properties:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved							

A.10.1.4 Interrupt, error, and system alarms

LED display



Meaning of the LED displays

The following table explains the meaning of the status and error displays. You can find remedial measures for diagnostic alarms in the "Diagnostic alarms" section.

LED DIAG

DIAG	Meaning
□ Off	Backplane bus supply of the S7-1200 not OK
☼ Flashes	Module is not configured
🟩 On	Module parameterized and no module diagnostics
☼ Flashes	Module parameterized and module diagnostics OR L+ power not connected

LED port status

Valid for IO-Link port which is in IO-Link port mode.

COM/1 ... COM/4	Meaning
□ Off	Port deactivated
☼ Flashes	Port activated, device not connected or Port is not connected to the configured device
🟩 On	Port activated, device connected

LED channel status

Valid for IO-Link port which is in DI/Q mode.

DI/Q1 ... DI/Q4	Meaning
□ Off	Process signal = 0
🟩 On	Process signal = 1

LED port error

F1 ... F4	Meaning
□ Off	No error
🟥 On	Error

Module errors are indicated as diagnostics (module status) only in IO-Link mode.

Diagnostic alarm	Error code (decimal)	STATUS (W#16#...)	Meaning (IO-Link error code)	IO-Link master	IO-Link device
Short-circuit	1	1804	Short-circuit at the process cables on the IO-Link device	X	
		7710	Short-circuit on IO device		X
Undervoltage	2	5111 5112	Supply voltage too low		X
Overvoltage	3	5110	Supply voltage too high		X
Overheating	5	1805	Temperature exceeded on master	X	
		4000 4210	Temperature exceeded on device		X
Wire break	6	1800	• No IO-Link device connected • There is a break on the signal line to the IO-Link device • IO-Link device cannot communicate due to a different error	X	
Overflow	7	8C10 8C20	Process tag range exceeded		X
		8C20	Measuring range exceeded		
Underflow	8	8C30	Process tag range too low		X
Error	9	---	All IO-Link error codes that are not listed here are mapped to this PROFIBUS DP error		X
Parameter assignment error	16	1882 1883	IO-Link master could not be configured	X	
		1802	Incorrect device		
		1886	Storage error		
		6320 6321 6350	Device was not configured correctly		X
Supply voltage missing	17	1806	L+ supply voltage for device missing	X	
		1807	L+ supply voltage for device too low (<20 V)		
Defective fuse	18	5101	Fuse on device is defective		X

Diagnostic alarm	Error code (decimal)	STATUS (W#16#...)	Meaning (IO-Link error code)	IO-Link master	IO-Link device
Safety shut-down	25	1880	Serious error (master has to be re-placed)	X	
External fault	26	1809 180A 180B 180C 180D	Error in data storage	X	
		1808	More than 6 errors are pending simul-taneously on the IO-Link device		

A.11 Digital signal boards (SBs)

A.11.1 SB 1221 200 kHz digital input specifications

Table A- 186 General specifications

Technical data	SB 1221 DI 4 x 24 VDC, 200 kHz	SB 1221 DI 4 x 5 VDC, 200 kHz
Article number	6ES7 221-3BD30-0XB0	6ES7 221-3AD30-0XB0
Dimensions W x H x D (mm)	38 x 62 x 21	
Weight	35 grams	
Power dissipation	1.5 W	1.0 W
Current consumption (SM Bus)	40 mA	
Current consumption (24 VDC)	7 mA / input + 20 mA	15 mA / input + 15 mA

Table A- 187 Digital inputs

Technical data		SB 1221 DI 4 x 24 VDC, 200 kHz	SB 1221 DI 4 x 5 VDC, 200 kHz
Number of inputs		4	
Type		Source	
Rated voltage		24 VDC at 7 mA, nominal	5 VDC at 15 mA, nominal
Continuous permissible voltage		28.8 VDC	6 VDC
Surge voltage		35 VDC for 0.5 sec.	6 V
Logic 1 signal (min.)		L+ minus 10 VDC at 2.9 mA	L+ minus 2.0 VDC at 5.1 mA
Logic 0 signal (max.)		L+ minus 5 VDC at 1.4 mA	L+ minus 1.0 VDC at 2.2 mA
HSC clock input rates (max.)		Single phase: 200 kHz Quadrature phase: 160 kHz	
Isolation (field side to logic)		500 VAC for 1 minute	
Isolation groups		1	
Filter times	us settings	0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 10.0,12.8, 20.0	

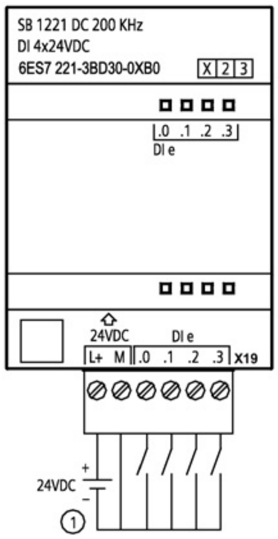
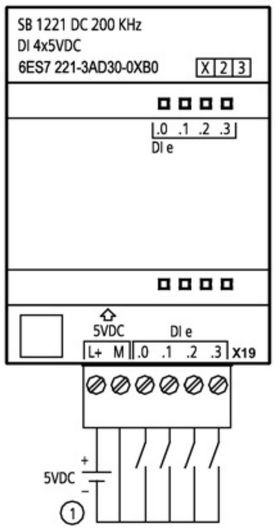
Technical data		SB 1221 DI 4 x 24 VDC, 200 kHz	SB 1221 DI 4 x 5 VDC, 200 kHz
	ms settings	0.05, 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 10.0, 12.8, 20.0	
Number of inputs on simultaneously		• 2 (no adjacent points) at 60 °C horizontal or 50 °C vertical • 4 at 55 °C horizontal or 45 °C vertical	4
Cable length (meters)		50 shielded twisted pair	

Note

When switching frequencies above 20 kHz, it is important that the digital inputs receive a square wave. Consider the following options to improve the signal quality to the inputs:

- Minimize the cable length
- Change a driver from a sink only driver to a sinking and sourcing driver
- Change to a higher quality cable
- Reduce the circuit/components from 24 V to 5 V
- Add an external load at the input

Table A- 188 Wiring diagrams for the 200 kHz digital input SBs

SB 1221 DI 4 x 24 VDC, 200 kHz (6ES7 221-3BD30-0XB0)	SB 1221 DI 4 x 5 VDC, 200 kHz (6ES7 221-3AD30-0XB0)
 SB 1221 DC 200 KHz DI 4x24VDC 6ES7 221-3BD30-0XB0 X 2 3 .0 .1 .2 .3 DI e 24VDC DI e L+ M .0 .1 .2 .3 X19 24VDC + – 1	 SB 1221 DC 200 KHz DI 4x5VDC 6ES7 221-3AD30-0XB0 X 2 3 .0 .1 .2 .3 DI e 5VDC DI e L+ M .0 .1 .2 .3 X19 5VDC + – 1

① Supports sourcing inputs only

Table A- 189 Connector pin locations for SB 1221 DI 4 x 24 VDC, 200 kHz (6ES7 221-3BD30-0XB0)

Pin	X19
1	L+ / 24 VDC
2	M / 24 VDC
3	DI e.0
4	DI e.1
5	DI e.2
6	DI e.3

Table A- 190 Connector pin locations for SB 1221 DI 4 x 5 VDC, 200 kHz (6ES7 221-3AD30-0XB0)

Pin	X19
1	L+ / 5 VDC
2	M / 5 VDC
3	DI e.0
4	DI e.1
5	DI e.2
6	DI e.3

A.11.2 SB 1222 200 kHz digital output specifications

Table A- 191 General specifications

Technical data	SB 1222 DQ 4 x 24 VDC, 200 kHz	SB 1222 DQ 4 x 5 VDC, 200 kHz
Article number	6ES7 222-1BD30-0XB0	6ES7 222-1AD30-0XB0
Dimensions W x H x D (mm)	38 x 62 x 21	
Weight	35 grams	
Power dissipation	0.5 W	
Current consumption (SM Bus)	35 mA	
Current consumption (24 VDC)	15 mA	

Table A- 192 Digital outputs

Technical data	SB 1222 DQ 4 x 24 VDC, 200 kHz	SB 1222 DQ 4 x 5 VDC, 200 kHz
Number of outputs	4	
Output type	Solid state - MOSFET sink and source[1]	
Voltage range	20.4 to 28.8 VDC	4.25 to 6.0 VDC
Logic 1 signal at max. current	L+ minus 1.5 V	L+ minus 0.7 V
Logic 0 signal at max. current	1.0 VDC, max.	0.2 VDC, max.
Current (max.)	0.1 A	
Lamp load	--	

Technical data	SB 1222 DQ 4 x 24 VDC, 200 kHz	SB 1222 DQ 4 x 5 VDC, 200 kHz
On state contact resistance	11 Ω max.	7 Ω max.
Off state resistance	6 Ω max.	0.2 Ω max.
Leakage current per point	--	
Pulse Train Output rate	200 kHz max., 2 Hz min.	
Surge current	0.11 A	
Overload protection	No	
Isolation (field side to logic)	500 VAC for 1 minute	
Isolation groups	1	
Currents per common	0.4 A	
Inductive clamp voltage	Non	
Switching delay	1.5 μs + 300 ns rise 1.5 μs + 300 ns fall	200 ns + 300 ns rise 200 ns + 300 ns fall
Behavior on RUN to STOP	Last value or substitute value (default value 0)	
Number of outputs on simultaneously	• 2 (no adjacent points) at 60 °C horizontal or 50 °C vertical • 4 at 55 °C horizontal or 45 °C vertical	4
Cable length (meters)	50 shielded twisted pair	

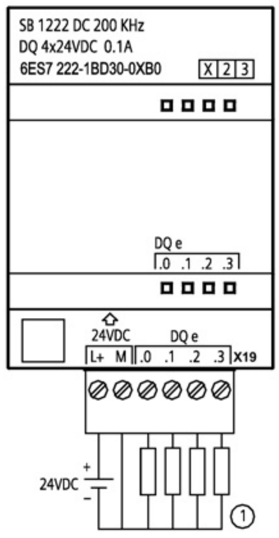
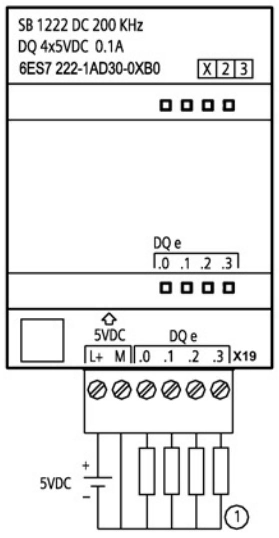
[1] Because both sinking and sourcing configurations are supported by the same circuitry, the active state of a sourcing load is opposite that of a sinking load. A source output exhibits positive logic (Q bit and LED are ON when the load has current flow), while a sink output exhibits negative logic (Q bit and LED are OFF when the load has current flow). If the module is plugged in with no user program, the default for this module is 0 V, which means that a sinking load will be turned ON.

Note

When switching frequencies above 20 kHz, it is important that the digital inputs receive a square wave. Consider the following options to improve the signal quality to the inputs:

- Minimize the cable length
- Change a driver from a sink only driver to a sinking and sourcing driver
- Change to a higher quality cable
- Reduce the circuit/components from 24 V to 5 V
- Add an external load at the input

Table A- 193 Wiring diagrams for the 200 kHz digital output SBs

SB 1222 DQ 4 x 24 VDC, 200 kHz (6ES7 222-1BD30-0XB0)	SB 1222 DQ 4 x 5 VDC, 200 kHz (6ES7 222-1AD30-0XB0)
 SB 1222 DC 200 KHz DQ 4x24VDC 0.1A 6ES7 222-1BD30-0XB0 X 2 3 DQ e .0 .1 .2 .3 24VDC DQ e L+ M .0 .1 .2 .3 X19 24VDC + - ①	 SB 1222 DC 200 KHz DQ 4x5VDC 0.1A 6ES7 222-1AD30-0XB0 X 2 3 DQ e .0 .1 .2 .3 5VDC DQ e L+ M .0 .1 .2 .3 X19 5VDC + - ①

① For sourcing outputs, connect "Load" to "-" (shown). For sinking outputs, connect "Load" to "+". Because both sinking and sourcing configurations are supported by the same circuitry, the active state of a sourcing load is opposite that of a sinking load. A source output exhibits positive logic (Q bit and LED are ON when the load has current flow), while a sink output exhibits negative logic (Q bit and LED are OFF when the load has current flow). If the module is plugged in with no user program, the default for this module is 0 V, which means that a sinking load will be turned ON.

Table A- 194 Connector pin locations for SB 1222 DQ 4 x 24 VDC, 200 kHz (6ES7 222-1BD30-0XB0)

Pin	X19
1	L+ / 24 VDC
2	M / 24 VDC
3	DQ e.0
4	DQ e.1
5	DQ e.2
6	DQ e.3

Table A- 195 Connector pin locations for SB 1222 DQ 4 x 5 VDC, 200 kHz (6ES7 222-1AD30-0XB0)

Pin	X19
1	L+ / 5 VDC
2	M / 5 VDC
3	DQ e.0
4	DQ e.1
5	DQ e.2
6	DQ e.3

A.11.3 SB 1223 200 kHz digital input / output specifications

Table A- 196 General specifications

Technical data	**SB 1223 DI 2 x 24 VDC / DQ 2 x 24 VDC, 200 kHz**	**SB 1223 DI 2 x 5 VDC / DQ 2 x 5 VDC, 200 kHz**
Article number	6ES7 223-3BD30-0XB0	6ES7 223-3AD30-0XB0
Dimensions W x H x D (mm)	38 x 62 x 21	
Weight	35 grams	
Power dissipation	1.0 W	0.5 W
Current consumption (SM Bus)	35 mA	
Current consumption (24 VDC)	7 mA / Input + 30 mA	15 mA / input + 15 mA

Table A- 197 Digital inputs

Technical data		**SB 1223 DI 2 x 24 VDC / DQ 2 x 24 VDC, 200 kHz**	**SB 1223 DI 2 x 5 VDC / DQ 2 x 5 VDC, 200 kHz**
Number of inputs		2	
Type		Source	
Rated voltage		24 VDC at 7 mA, nominal	5 VDC at 15 mA, nominal
Continuous permissible voltage		28.8 VDC	6 VDC
Surge voltage		35 VDC for 0.5 sec.	6 V
Logic 1 signal (min.)		L+ minus 10 VDC at 2.9 mA	L+ minus 2.0 VDC at 5.1 mA
Logic 0 signal (max.)		L+ minus 5 VDC at 1.4 mA	L+ minus 1.0 VDC at 2.2 mA
HSC clock input rates (max.)		Single phase: 200 kHz Quadrature phase: 160 kHz	
Isolation (field side to logic)		500 VAC for 1 minute	
Isolation groups		1 (no isolation to outputs)	
Filter times	us settings	0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 10.0, 12.8, 20.0	
	ms settings	0.05, 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 10.0, 12.8, 20.0	
Number of inputs on simultaneously		2	
Cable length (meters)		50 shielded twisted pair	

Table A- 198 Digital outputs

Technical data	**SB 1223 DI 2 x 24 VDC / DQ 2 x 24 VDC, 200 kHz**	**SB 1223 DI 2 x 5 VDC / DQ 2 x 5 VDC, 200 kHz**
Number of outputs	2	
Output type	Solid state - MOSFET sink and source[1]	
Voltage range	20.4 to 28.8 VDC	4.25 to 6.0 VDC
Rated value	24 VDC	5 VDC
Logic 1 signal at max. current	L+ minus 1.5 V	L+ minus 0.7 V
Logic 0 signal at max. current	1.0 VDC, max.	0.2 VDC, max.

Technical data	SB 1223 DI 2 x 24 VDC / DQ 2 x 24 VDC, 200 kHz	SB 1223 DI 2 x 5 VDC / DQ 2 x 5 VDC, 200 kHz
Current (max.)	0.1 A	
Lamp load	--	
On state contact resistance	11 Ω max.	7 Ω max.
Off state resistance	6 Ω max.	0.2 Ω max.
Leakage current per point	--	
Pulse Train Output rate	200 kHz max., 2 Hz min.	
Surge current	0.11 A	
Overload protection	No	
Isolation (field side to logic)	500 VAC for 1 minute	
Isolation groups	1 (no isolation to inputs)	
Currents per common	0.2 A	
Inductive clamp voltage	Non	
Switching delay	1.5 μs + 300 ns rise 1.5 μs + 300 ns fall	200 ns + 300 ns rise 200 ns + 300 ns fall
Behavior on RUN to STOP	Last value or substitute (default value 0)	
Number of outputs on simultaneously	2	
Cable length (meters)	50 shielded twisted pair	

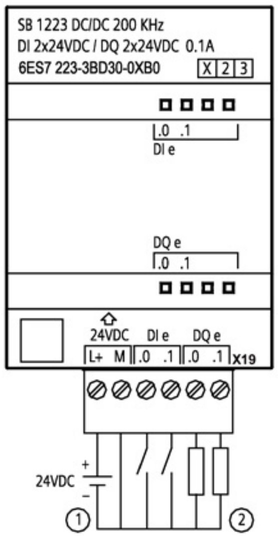
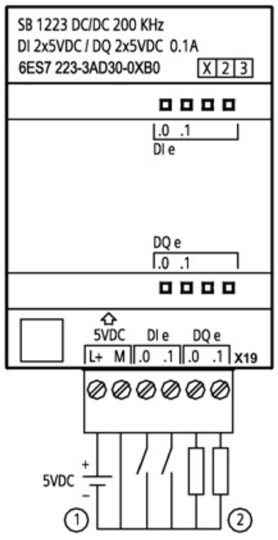
[1] Because both sinking and sourcing configurations are supported by the same circuitry, the active state of a sourcing load is opposite that of a sinking load. A source output exhibits positive logic (Q bit and LED are ON when the load has current flow), while a sink output exhibits negative logic (Q bit and LED are OFF when the load has current flow). If the module is plugged in with no user program, the default for this module is 0 V, which means that a sinking load will be turned ON.

Note

When switching frequencies above 20 kHz, it is important that the digital inputs receive a square wave. Consider the following options to improve the signal quality to the inputs:

- Minimize the cable length
- Change a driver from a sink only driver to a sinking and sourcing driver
- Change to a higher quality cable
- Reduce the circuit/components from 24 V to 5 V
- Add an external load at the input

Table A- 199 Wiring diagrams for the 200 kHz digital input/output SBs

SB 1223 DI 2 x 24 VDC/DQ 2 x 24 VDC, 200 kHz (6ES7 223-3BD30-0XB0)	SB 1223 DI 2 x 5 VDC / DQ 2 x 5 VDC, 200 kHz (6ES7 223-3AD30-0XB0)
 SB 1223 DC/DC 200 KHz DI 2x24VDC / DQ 2x24VDC 0.1A 6ES7 223-3BD30-0XB0 X 2 3 .0 .1 DI e DQ e .0 .1 24VDC DI e DQ e L+ M .0 .1 .0 .1 X19 24VDC + − ① ②	 SB 1223 DC/DC 200 KHz DI 2x5VDC / DQ 2x5VDC 0.1A 6ES7 223-3AD30-0XB0 X 2 3 .0 .1 DI e DQ e .0 .1 5VDC DI e DQ e L+ M .0 .1 .0 .1 X19 5VDC + − ① ②

① Supports sourcing inputs only

② For sourcing outputs, connect "Load" to "-" (shown). For sinking outputs, connect "Load" to "+". [1] Because both sinking and sourcing configurations are supported by the same circuitry, the active state of a sourcing load is opposite that of a sinking load. A source output exhibits positive logic (Q bit and LED are ON when the load has current flow), while a sink output exhibits negative logic (Q bit and LED are OFF when the load has current flow). If the module is plugged in with no user program, the default for this module is 0 V, which means that a sinking load will be turned ON.

Table A- 200 Connector pin locations for SB 1223 DI 2 x 24 VDC/DQ 2 x 24 VDC, 200 kHz (6ES7 223-3BD30-0XB0)

Pin	X19
1	L+ / 24 VDC
2	M / 24 VDC
3	DI e.0
4	DI e.1
5	DQ e.0
6	DQ e.1

Table A- 201 Connector pin locations for SB 1223 DI 2 x 5 VDC / DQ 2 x 5 VDC, 200 kHz (6ES7 223-3AD30-0XB0)

Pin	X19
1	L+ / 5 VDC
2	M / 5 VDC
3	DI e.0
4	DI e.1
5	DQ e.0
6	DQ e.1

A.11.4 SB 1223 2 X 24 VDC input / 2 X 24 VDC output specifications

Table A- 202 General specifications

Technical Data	SB 1223 DI 2 x 24 VDC, DQ 2 x 24 VDC
Article number	6ES7 223-0BD30-0XB0
Dimensions W x H x D (mm)	38 x 62 x 21
Weight	40 grams
Power dissipation	1.0 W
Current consumption (SM Bus)	50 mA
Current consumption (24 VDC)	4 mA / Input used

Table A- 203 Digital inputs

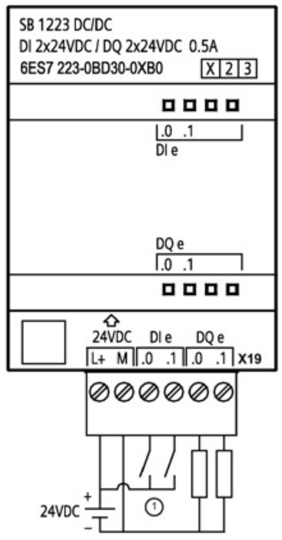
Technical Data		SB 1223 DI 2 x 24 VDC, DQ 2 x 24 VDC
Number of inputs		2
Type		IEC Type 1 sink
Rated voltage		24 VDC at 4 mA, nominal
Continuous permissible voltage		30 VDC, max.
Surge voltage		35 VDC for 0.5 sec.
Logic 1 signal (min.)		15 VDC at 2.5 mA
Logic 0 signal (max.)		5 VDC at 1 mA
HSC clock input rates (max.)		Single phase: 30 kHz (15 to 26 VDC) Quadrature phase: 20 kHz (15 to 26 VDC)
Isolation (field side to logic)		500 VAC for 1 minute
Isolation groups		1
Filter times	us settings	0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4,10.0, 12.8, 20.0
	ms settings	0.05, 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 10.0, 12.8, 20.0
Number of inputs on simultaneously		2
Cable length (meters)		500 shielded, 300 unshielded

Table A- 204 Digital outputs

Technical Data	SB 1223 DI 2 x 24 VDC, DQ 2 x 24 VDC
Number of outputs	2
Output type	Solid state - MOSFET (sourcing)
Voltage range	20.4 to 28.8 VDC
Logic 1 signal at max. current	20 VDC min.
Logic 0 signal with 10K Ω load	0.1 VDC max.
Current (max.)	0.5 A
Lamp load	5 W
On state contact resistance	0.6 Ω max.
Leakage current per point	10 μA max.
Pulse Train Output (PTO) rate	20 kHz max., 2 Hz min.[1]
Surge current	5 A for 100 ms max.
Overload protection	No
Isolation (field side to logic)	500 VAC for 1 minute
Isolation groups	1
Currents per common	1 A
Inductive clamp voltage	L+ minus 48 V, 1 W dissipation
Switching delay	2 μs max. off to on 10 μs max. on to off
Behavior on RUN to STOP	Last value or substitute value (default value 0)
Number of outputs on simultaneously	2
Cable length (meters)	500 m shielded, 150 m unshielded

[1] Depending on your pulse receiver and cable, an additional load resistor (at least 10% of rated current) may improve pulse signal quality and noise immunity.

Table A- 205 Wiring diagram for the digital input/output SB

SB 1223 DI 2 x 24 VDC, DQ 2 x 24 VDC (6ES7 223-0BD30-0XB0)	
 SB 1223 DC/DC DI 2x24VDC / DQ 2x24VDC 0.5A 6ES7 223-0BD30-0XB0 X 2 3 .0 .1 DI e DQ e .0 .1 24VDC DI e DQ e L+ M .0 .1 .0 .1 X19 24VDC + – ①	

① Supports sinking inputs only

Table A- 206 Connector pin locations for SB 1223 DI 2 x 24 VDC, DQ 2 x 24 VDC (6ES7 223-0BD30-0XB0)

Pin	X19
1	L+ / 24 VDC
2	M / 24 VDC
3	DI e.0
4	DI e.1
5	DQ e.0
6	DQ e.1

A.12 Analog signal boards (SBs)

A.12.1 SB 1231 1 analog input specifications

Note

To use this SB, your CPU firmware must be V2.0 or higher.

Table A- 207 General specifications

Technical data	SB 1231 AI 1 x 12 bit
Article number	6ES7 231-4HA30-0XB0
Dimensions W x H x D (mm)	38 x 62 x 21
Weight	35 grams
Power dissipation	0.4 W
Current consumption (SM Bus)	55 mA
Current consumption (24 VDC)	none

Table A- 208 Analog inputs

Technical data	SB 1231 AI 1x12 bit
Number of inputs	1
Type	Voltage or current (differential)
Range	±10 V, ±5 V, ±2.5 or 0 to 20 mA
Resolution	11 bits + sign bit
Full scale range (data word)	-27648 to 27648
Over/Under range (data word)	Voltage: 32511 to 27649 / -27649 to -32512 Current: 32511 to 27649 / 0 to -4864 (Refer to Analog input representation for voltage and Analog input representation for current (Page 1236).)
Overflow/Underflow (data word)	Voltage: 32767 to 32512 / -32513 to -32768 Current: 32767 to 32512 / -4865 to -32768 (Refer to Analog input representation for voltage and Analog input representation for current (Page 1236).)
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 1236).)
Noise rejection	400, 60, 50, or 10 Hz (refer to Analog input response times for sample rates (Page 1236).)
Accuracy (25 °C / -20 to 60 °C)	±0.3% / ±0.6% of full scale

Technical data	SB 1231 AI 1x12 bit
Input impedance Differential Common mode	 Voltage: 220 kΩ; Current: 250 Ω Voltage: 55 kΩ; Current: 55 kΩ
Behavior on RUN to STOP	Last value or substitute value (default value 0)
Measuring principle	Actual value conversion
Common mode rejection	40 dB, DC to 60 Hz
Operational signal range	Signal plus common mode voltage must be less than +35 V and greater than -35 V
Isolation (field side to logic)	None
Cable length (meters)	100 m, twisted and shielded

Table A- 209 Diagnostics

Technical data	SB 1231 AI 1 x 12 bit
Overflow/underflow	Yes
24 VDC low voltage	no

Table A- 210 Wiring diagram for the analog input SB

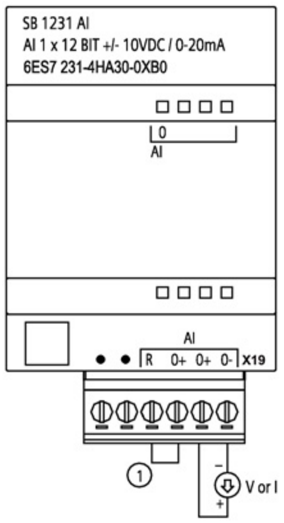
SB 1231 AI x 12 bit (6ES7 231-4HA30-0XB0)	
 SB 1231 AI AI 1 x 12 BIT +/- 10VDC / 0-20mA 6ES7 231-4HA30-0XB0 0 AI AI R 0+ 0+ 0- X19 ① V or I	① Connect "R" and "0+" for current applications. Note: Connectors must be gold. See Appendic C, Spare Parts for article number.

Table A- 211 Connector pin locations for SB 1231 AI x 12 bit (6ES7 231-4HA30-0XB0)

Pin	X19 (gold)
1	No connection
2	No connection
3	AI R
4	AI 0+
5	AI 0+
6	AI 0-

A.12.2 SB 1232 1 analog output specifications

Table A- 212 General specifications

Technical data	SB 1232 AQ 1 x 12 bit
Article number	6ES7 232-4HA30-0XB0
Dimensions W x H x D (mm)	38 x 62 x 21
Weight	40 grams
Power dissipation	1.5 W
Current consumption (SM Bus)	15 mA
Current consumption (24 VDC)	40 mA (no load)

Table A- 213 Analog outputs

Technical data	SB 1232 AQ 1 x 12 bit
Number of outputs	1
Type	Voltage or current
Range	±10 V or 0 to 20 mA
Resolution	Voltage: 12 bits Current: 11 bits
Full scale range (data word) Refer to the output ranges for voltage and current (Page 1237).	Voltage: -27648 to 27648 Current: 0 to 27648
Accuracy (25 °C / -20 to 60 °C)	±0.5% / ±1% 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
Behavior on RUN to STOP	Last value or substitute value (default value 0)
Isolation (field side to logic)	None
Cable length (meters)	100 m, twisted and shielded

Table A- 214 Diagnostics

Technical data	SB 1232 AQ 1 x 12 bit
Overflow/underflow	Yes
Short to ground (voltage mode only)	Yes
Wire break (current mode only)	Yes

Table A- 215 Wiring diagram for the SB 1232 AQ 1 x 12 bit

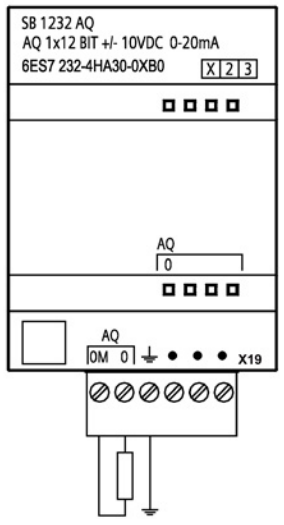
SB 1232 AQ 1 x 12 bit (6ES7 232-4HA30-0XB0)	
 SB 1232 AQ AQ 1x12 BIT +/- 10VDC 0-20mA 6ES7 232-4HA30-0XB0 X 2 3 AQ 0 AQ 0M 0 X19	
Note: Connectors must be gold. See Appendix C, Spare Parts for article number.	

Table A- 216 Connector pin locations for SB 1232 AQ 1 x 12 bit (6ES7 232-4HA30-0XB0)

Pin	X19 (gold)
1	AQ 0M
2	AQ 0
3	Functional Earth
4	No connection
5	No connection
6	No connection

A.12.3 Measurement ranges for analog inputs and outputs

A.12.3.1 Step response of the analog inputs

Table A- 217 Step response (ms), 0 V to 10 V measured at 95%

Smoothing selection (sample averaging)	**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	4.5 ms	18.7 ms	22.0 ms	102 ms
Weak (4 cycles): 4 samples	10.6 ms	59.3 ms	70.8 ms	346 ms
Medium (16 cycles): 16 samples	33.0 ms	208 ms	250 ms	1240 ms
Strong (32 cycles): 32 samples	63.0 ms	408 ms	490 ms	2440 ms
Sample time	**0.156 ms**	**1.042 ms**	**1.250 ms**	**6.250 ms**

A.12.3.2 Sample time and update times for the analog inputs

Table A- 218 Sample time and update time

Selection	**Sample time**	**SB update time**
400 Hz (2.5 ms)	0.156 ms	0.156 ms
60 Hz (16.6 ms)	1.042 ms	1.042 ms
50 Hz (20 ms)	1.250 ms	1.25 ms
10 Hz (100 ms)	6.250 ms	6.25 ms

A.12.3.3 Measurement ranges of the analog inputs for voltage and current (SB and SM)

Table A- 219 Analog input representation for voltage (SB and SM)

System		**Voltage Measuring Range**				
Decimal	**Hexadecimal**	**±10 V**	**±5 V**	**±2.5 V**	**±1.25 V**	
32767	7FFF[1]	11.851 V	5.926 V	2.963 V	1.481 V	Overflow
32512	7F00					
32511	7EFF	11.759 V	5.879 V	2.940 V	1.470 V	Overshoot range
27649	6C01					
27648	6C00	10 V	5 V	2.5 V	1.250 V	Rated range
20736	5100	7.5 V	3.75 V	1.875 V	0.938 V	
1	1	361.7 μV	180.8 μV	90.4 μV	45.2 μV	
0	0	0 V	0 V	0 V	0 V	
-1	FFFF					
-20736	AF00	-7.5 V	-3.75 V	-1.875 V	-0.938 V	
-27648	9400	-10 V	-5 V	-2.5 V	-1.250 V	

System		Voltage Measuring Range				
Decimal	Hexadecimal	±10 V	±5 V	±2.5 V	±1.25 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- 220 Analog input representation for current (SB and SM)

System		Current measuring range		
Decimal	Hexadecimal	0 mA to 20 mA	4 mA to 20 mA	
32767	7FFF	23.70 mA	22.96 mA	Overflow
32512	7F00			
32511	7EFF	23.52 mA	22.81 mA	Overshoot range
27649	6C01			
27648	6C00	20 mA	20 mA	Nominal range
20736	5100	15 mA	16 mA	
1	1	723.4 nA	4 mA + 578.7 nA	
0	0	0 mA	4 mA	
-1	FFFF			Undershoot range
-4864	ED00	-3.52 mA	1.185 mA	
-4865	ECFF			Underflow
-32768	8000			

A.12.3.4 Measurement ranges of the analog outputs for voltage and current (SB and SM)

Table A- 221 Analog output representation for voltage (SB and SM)

System		Voltage Output Range	
Decimal	Hexadecimal	± 10 V	
32767	7FFF	See note 1	Overflow
32512	7F00	See note 1	
32511	7EFF	11.76 V	Overshoot range
27649	6C01		
27648	6C00	10 V	Rated range
20736	5100	7.5 V	
1	1	361.7 µ V	
0	0	0 V	
-1	FFFF	-361.7 µ V	
-20736	AF00	-7.5 V	
-27648	9400	-10 V	

System		Voltage Output Range	
Decimal	**Hexadecimal**	**± 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- 222 Analog output representation for current (SB and SM)

System		Current output range		
Decimal	**Hexadecimal**	**0 mA to 20 mA**	**4 mA to 20 mA**	
32767	7FFF	See note 1	See note 1	Overflow
32512	7F00	See note 1	See note 1	
32511	7EFF	23.52 mA	22.81 mA	Overshoot range
27649	6C01			
27648	6C00	20 mA	20 mA	Rated range
20736	5100	15 mA	16 mA	
1	1	723.4 nA	4 mA + 578.7 nA	
0	0	0 mA	4mA	
-1	FFFF		4 mA to 578.7 nA	Undershoot range
-6912	E500		0 mA	
-6913	E4FF			Not possible. Output value limited to 0 mA.
-32512	8100			
-32513	80FF	See note 1	See note 1	Underflow
-32768	8000	See note 1	See note 1	

1 In an overflow or underflow condition, analog outputs will take on the substitute value of the STOP mode.

A.12.4 Thermocouple signal boards (SBs)

A.12.4.1 SB 1231 1 analog thermocouple input specifications

Note

To use this SB, your CPU firmware must be V2.0 or higher.

Table A- 223 General specifications

Technical data	SB 1231 AI 1 x 16 bit Thermocouple
Article number	6ES7 231-5QA30-0XB0
Dimensions W x H x D (mm)	38 x 62 x 21
Weight	35 grams
Power dissipation	0.5 W
Current consumption (SM Bus)	5 mA
Current consumption (24 VDC)	20 mA

Table A- 224 Analog inputs

Technical data		SB 1231 AI 1x16 bit Thermocouple
Number of inputs		1
Type		Floating TC and mV
Range • Nominal range (data word) • Overrange/underrange (data word) • Overflow/underflow (data word)		See Thermocouple filter selection table (Page 1240).
Resolution	Temperature	0.1° C / 0.1° F
	Voltage	15 bits plus sign
Maximum withstand voltage		±35 V
Noise rejection		85 dB for the selected filter setting (10 Hz, 50 Hz, 60 Hz, 400 Hz)
Common mode rejection		> 120 dB at 120 VAC
Impedance		≥ 10 M Ω
Accuracy		See Thermocouple selection table (Page 1240).
Repeatability		±0.05% FS
Measuring principle		Integrating
Module update time		See Thermocouple filter selection table (Page 1240).
Cold junction error		±1.5° C
Isolation (field side to logic)		500 VAC

Technical data	SB 1231 AI 1x16 bit Thermocouple
Cable length (meters)	100 m to sensor max.
Wire resistance	100 Ω max.

Table A- 225 Diagnostics

Technical data	SB 1231 AI 1 x 16 bit Thermocouple
Overflow/underflow[1]	Yes
Wire break[2]	Yes

[1] The overflow and underflow 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 SM 1231 Thermocouple (TC) analog signal module measures the value of voltage connected to the module inputs.

The SB 1231 Thermocouple analog signal board measures the value of voltage connected to the signal board 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.

A.12.4.2 Basic operation for 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 SM 1231 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.

Selection table for the SB 1231 thermocouple

The ranges and accuracy for the different thermocouple types supported by the SB 1231 Thermocouple signal board are shown in the table below.

Table A- 226 SB 1231 Thermocouple selection table

Thermocouple Type	Under range minimum[1]	Nominal range low limit	Nominal range high limit	Over range maximum[2]	Normal range [3] accuracy @ 25 °C	Normal range [3], accuracy -20 °C to 60 °C
J	-210.0 °C	-150.0 °C	1200.0 °C	1450.0 °C	±0.3 °C	±0.6 °C
K	-270.0 °C	-200.0 °C	1372.0 °C	1622.0 °C	±0.4 °C	±1.0 °C
T	-270.0 °C	-200.0 °C	400.0 °C	540.0 °C	±0.5 °C	±1.0 °C
E	-270.0 °C	-200.0 °C	1000.0 °C	1200.0 °C	±0.3 °C	±0.6 °C
R & S	-50.0 °C	100.0 °C	1768.0 °C	2019.0 °C	±1.0 °C	±2.5 °C
B	0.0 °C	200.0 °C	800.0 °C	--	±2.0 °C	±2.5 °C
	--	800.0 °C	1820.0 °C	1820.0 °C	±1.0 °C	±2.3 °C
N	-270.0 °C	0.0 °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	1050.0 °C	±0.6 °C	±1.2 °C
Voltage	-32511	-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 minimum value are reported as 32767.

[3] Internal cold junction error is ±1.5 °C for all ranges. This adds to the error in this table. The signal board requires at least 30 minutes of warm-up time to meet this specification.

Table A- 227 Filter selection table for the SB 1231 Thermocouple

Rejection frequency (Hz)	Integration time (ms)	Signal board update time (seconds)
10	100	0.306
50	20	0.066
60	16.67	0.056
400[1]	10	0.036

[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 to the module, it 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.

Table A- 228 Wiring diagram for SB 1231 AI 1 x 16 thermocouple

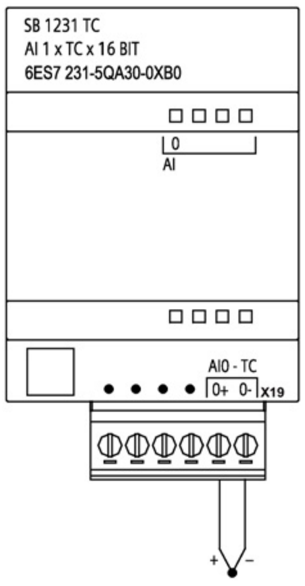
SB 1231 AI 1 x 16 bit thermocouple (6ES7 231-5QA30-0XB0)	
 SB 1231 TC AI 1 x TC x 16 BIT 6ES7 231-5QA30-0XB0 0 AI AI0 - TC 0+ 0- X19 + –	
Note: Connectors must be gold. See Appendix C, Spare Parts for article number.	

Table A- 229 Connector pin locations for SB 1231 AI 1 x 16 bit thermocouple (6ES7 231-5QA30-0XB0)

Pin	X19 (gold)
1	No connection
2	No connection
3	No connection
4	No connection
5	AI 0- /TC
6	AI 0+ /TC

A.12.5 RTD signal boards (SBs)

A.12.5.1 SB 1231 1 analog RTD input specifications

Note

To use this SB, your CPU firmware must be V2.0 or higher.

Table A- 230 General specifications

Technical data	SB 1231 AI 1 x 16 bit RTD
Article number	6ES7 231-5PA30-0XB0
Dimensions W x H x D (mm)	38 x 62 x 2
Weight	35 grams
Power dissipation	0.7 W
Current consumption (SM Bus)	5 mA
Current consumption (24 VDC)	25 mA

Table A- 231 Analog inputs

Technical data		SB 1231 AI 1 x 16 bit RTD
Number of inputs		1
Type		Module referenced RTD and Ohms
Range • Nominal range (data word) • Overrange/underrange (data word) • Overflow/underflow (data word)		See Selection tables (Page 1246).
Resolution	Temperature	0.1 °C/ 0.1 °F
	Voltage	15 bits plus sign
Maximum withstand voltage		±35 V
Noise rejection		85 dB (10 Hz, 50 Hz, 60 Hz, 400 Hz)
Common mode rejection		> 120 dB
Impedance		≥ 10 MΩ
Accuracy		See Selection tables (Page 1246).
Repeatability		±0.05% FS
Maximum sensor dissipation		0.5 m W
Measuring principle		Integrating
Module update time		See Selection table (Page 1246).
Isolation (field side to logic)		500 VAC
Cable length (meters)		100 m to sensor max.
Wire resistance		20 Ω, 2.7 for 10 Ω RTD max.

Table A- 232 Diagnostics

Technical data	SB 1231 AI 1 x 16 bit RTD
Overflow/underflow[1, 2]	Yes
Wire break [3]	Yes

1 The overflow and underflow 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.

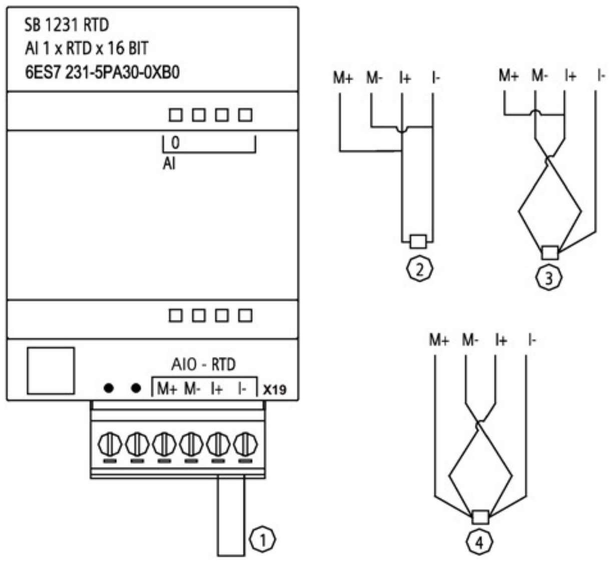
3 When wire break alarm is disabled and an open wire condition exists in the sensor wiring, the module may report random values.

The SM 1231 RTD analog signal board measures the value of resistance connected to the signal board 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 climatic range values will be reported in degrees multiplied by one hundred (for example, 25.34 degrees will be reported as decimal 2534).

The SB 1231 RTD signal board supports measurements with 2-wire, 3-wire and 4-wire connections to the sensor resistor.

Table A- 233 Wiring diagram for SB 1231 AI 1 x 16 bit RTD

SB 1231 AI 1 x 16 bit RTD (6ES7 231-5PA30-0XB0)


① Loop-back unused RTD input

② 2-wire RTD

③ 3-wire RTD

④ 4-wire RTD

Note: Connectors must be gold. See Appendix C, Spare Parts for article number.

Table A- 234 Connector pin locations for SB 1231 AI 1 x 16 bit RTD (6ES7 231-5PA30-0XB0)

Pin	X19 (gold)
1	No connection
2	No connection
3	AI 0 M+ /RTD
4	AI 0 M- /RTD
5	AI 0 I+ /RTD
6	AI 0 I- /RTD

A.12.5.2 Selection tables for the SB 1231 RTD

Table A- 235 Ranges and accuracy for the different sensors supported by the RTD modules

Temperature coefficient	RTD type	Under range minimum[1]	Nominal range low limit	Nominal range high limit	Over range maximum[2]	Normal range accuracy @ 25 °C	Normal range accuracy -20 °C to 60 °C
Pt 0.003850 ITS90 DIN EN 60751	Pt 100 climatic	-145.00 °C	-120.00 °C	-145.00 °C	-155.00 °C	±0.20 °C	±0.40 °C
	Pt 10	-243.0 °C	-200.0 °C	850.0 °C	1000.0 °C	±1.0 °C	±2.0 °C
	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						
	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
Cu 0.004270	Cu 10	-240.0 °C	-200.0 °C	260.0 °C	312.0 °C	±1.0 °	±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 are reported as -32768.

[2] RTD values above the over-range maximum value are reported as +32768.

Table A- 236 Resistance

Range	Under range minimum	Nominal range low limit	Nominal range high limit	Over range maximum[1]	Normal range accuracy @ 25 °C	Normal range accuracy -20 °C to 60 °C
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%

[1] Resistance values above the over-range maximum 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.

Best accuracy will be achieved for the 10 Ω RTD ranges if 4 wire connections are used.

The resistance of the connection wires in 2 wire mode will cause an error in the sensor reading and therefore accuracy is not guaranteed.

Table A- 237 Noise reduction and update times for the RTD modules

Rejection frequency selection	Integration time	4-/2-wire, 1-channel module Update time (seconds)	3-wire, 1-channel module Update time (seconds)
400 Hz (2.5 ms)	10 ms [1]	0.036	0.071
60 Hz (16.6 ms)	16.67 ms	0.056	0.111
50 Hz (20 ms)	20 ms	0.066	1.086
10 Hz (100 ms)	100 ms	0.306	0.611

[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.

A.13 BB 1297 Battery board

BB 1297 Battery Board

The S7-1200 BB 1297 Battery Board is designed for long-term backup of the Real-time clock. It is pluggable in the signal board slot of the S7-1200 CPU (firmware 3.0 and later versions). You must add the BB 1297 to the device configuration and download the hardware configuration to the CPU for the BB to be functional.

The battery (type CR1025) is not included with the BB 1297 and must be purchased by the user.

Note

The BB 1297 is mechanically designed to fit the CPUs with the firmware 3.0 and later versions.

Do not use the BB 1297 with earlier version CPUs as the BB 1297 connector will not plug into the CPU.

⚠ WARNING

Installing an unspecified battery in the BB 1297, or otherwise connecting an unspecified battery to the circuit can result in fire or component damage and unpredictable operation of machinery.

Fire or unpredictable operation of machinery can result in death, severe personal injury, or property damage.

Use only the specified CR1025 battery for backup of the Real-time clock.

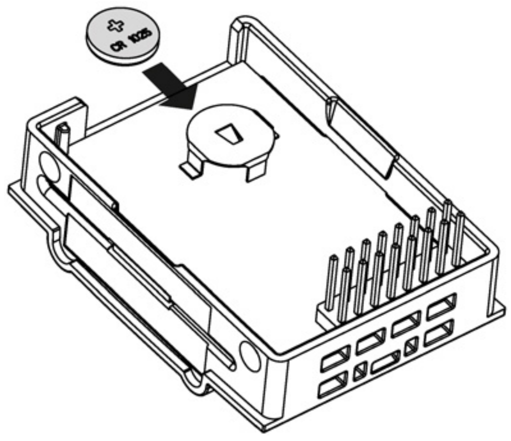
Table A- 238 General specifications

Technical data	BB 1297 Battery Board
Article number	6ES7 297-0AX30-0XA0
Dimensions W x H x D (mm)	38 x 62 x 21
Weight	28 grams
Power dissipation	0.5 W
Current consumption (SM Bus)	11 mA
Current consumption (24 VDC)	none

Battery (not included)	BB 1297 Battery Board
Hold up time	Approximately 1 year
Battery type	CR1025 Refer to Installing or replacing a battery in the BB 1297 battery board (Page 60)
Nominal voltage	3 V
Nominal capacity	At least 30 mAH

Diagnostics	BB 1297 Battery Board
Critical battery level	< 2.5 V
Battery diagnostic	Low voltage indicator: • Low battery voltage causes the CPU MAINT LED to illuminate with the amber light continuously ON. • Diagnostic Buffer Event: 16#06:2700 "Submodule maintenance demanded: At least one battery exhausted (BATTF)"
Battery status	Battery status bit provided 0 = Battery OK 1 = Battery low
Battery status update	Battery status is updated at power up and then once per day while CPU is in RUN mode.

Table A- 239 Insertion diagram for the BB 1297 battery board

BB 1297 battery board (6ES7 297-0AX30-0XA0)	
	

A.14 Communication interfaces

A.14.1 PROFIBUS

A.14.1.1 CM 1242-5 PROFIBUS DP SLAVE

Table A- 240 Technical specifications of the CM 1242-5

Technical specifications	
Article number	6GK7 242-5DX30-0XE0
Interfaces	
Connection to PROFIBUS	9-pin D-sub female connector
Maximum current consumption on the PROFIBUS interface when connecting network components (for example optical network components)	15 mA at 5 V (only for bus termination) *)
Permitted ambient conditions	
Ambient temperature • during storage • during transportation • during operation with a vertical installation (DIN rail horizontal) • during operation with a horizontal installation (DIN rail vertical)	 • -40 °C to 70 °C • -40 °C to 70 °C • 0 °C to 55 °C • 0 °C to 45 °C
Relative humidity at 25 °C during operation, without condensation, maximum	95 %
Degree of protection	IP20
Power supply, current consumption and power loss	
Type of power supply	DC
Power supply from the backplane bus	5 V
Current consumption (typical)	150 mA
Effective power loss (typical)	0.75 W
Electrical isolation • PROFIBUS interface to ground • PROFIBUS interface to internal circuit	710 VDC for 1 minute
Dimensions and weights	
• Width • Height • Depth	• 30 mm • 100 mm • 75 mm

Technical specifications	
Weight • Net weight • Weight including packaging	 • 115 g • 152 g

*)The current load of an external consumer connected between VP (pin 6) and DGND (pin 5) must not exceed a maximum of 15 mA (short-circuit proof) for bus termination.

A.14.1.2 Pinout of the D-sub socked of the CM 1242-5

PROFIBUS interface

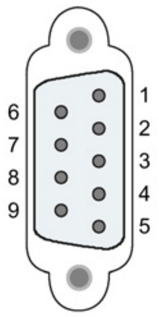


Table A- 241 Pinout of the D-sub socket

Pin	Description	Pin	Description
1	- not used -	6	P5V2: +5V power supply
2	- not used -	7	- not used -
3	RxD/TxD-P: Data line B	8	RxD/TxD-N: Data line A
4	RTS	9	- not used -
5	M5V2: Data reference potential (ground DGND)	Housing	Ground connector

A.14.1.3 CM 1243-5 PROFIBUS DP Master

Table A- 242 Technical specifications of the CM 1243-5

Technical specifications	
Article number	6GK7 243-5DX30-0XE0
Interfaces	
Connection to PROFIBUS	9-pin D-sub female connector
Maximum current consumption on the PROFIBUS interface when connecting network components (for example optical network components)	15 mA at 5 V (only for bus termination) *)
Permitted ambient conditions	
Ambient temperature • during storage • during transportation • during operation with a vertical installation (DIN rail horizontal) • during operation with a horizontal installation (DIN rail vertical)	 • -40 °C to 70 °C • -40 °C to 70 °C • 0 °C to 55 °C • 0 °C to 45 °C
Relative humidity at 25 °C during operation, without condensation, maximum	95 %
Degree of protection	IP20
Power supply, current consumption and power loss	
Type of power supply	DC
Power supply / external • minimum • maximum	24 V • 19.2 V • 28.8 V
Current consumption (typical) • from 24 V DC • from the S7-1200 backplane bus	 • 100 mA • 0 mA
Effective power loss (typical) • from 24 V DC • from the S7-1200 backplane bus	 • 2.4 W • 0 W
Power supply 24 VDC / external • Min. cable cross section • Max. cable cross section • Tightening torque of the screw terminals	 • min.: 0.14 mm^2 (AWG 25) • max.: 1.5 mm^2 (AWG 15) • 0.45 Nm (4 lb-in)
Electrical isolation • PROFIBUS interface to ground • PROFIBUS interface to internal circuit	710 VDC for 1 minute
Dimensions and weights	

Technical specifications	
• Width • Height • Depth	• 30 mm • 100 mm • 75 mm
Weight • Net weight • Weight including packaging	 • 134 g • 171 g

*)The current load of an external consumer connected between VP (pin 6) and DGND (pin 5) must not exceed a maximum of 15 mA (short-circuit proof) for bus termination.

Note

The CM 1243- (PROFIBUS master module) must eceive power from the 24 VDC sensor supply of the CPU.

Note

The CM 1243-5 (PROFIBUS master module) must receive power from the 24 VDC sensor supply of the CPU.

A.14.1.4 Pinout of the D-sub socket of the CM 1243-5

PROFIBUS interface

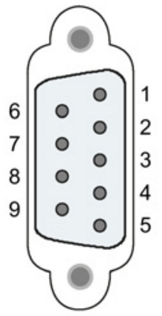


Table A- 243 Pinout of the D-sub socket

Pin	Description	Pin	Description
1	- not used -	6	VP: Power supply +5 V only for bus terminating resistors; not for supplying external devices
2	- not used -	7	- not used -

Pin	Description	Pin	Description
3	RxD/TxD-P: Data line B	8	RxD/TxD-N: Data line A
4	CNTR-P: RTS	9	- not used -
5	DGND: Ground for data signals and VP	Housing	Ground connector

PROFIBUS cable

Note

Contacting the shield of the PROFIBUS cable

The shield of the PROFIBUS cable must be contacted.

To do this, strip the insulation from the end of the PROFIBUS cable and connect the shield to functional earth.

A.14.2 CP 1242-7

Note

The CP 1242-7 is not approved for Maritime applications

The CP 1242-7 does not have Maritime approval.

Note

To use these modules, your CPU firmware must be V2.0 or higher.

A.14.2.1 CP 1242-7 GPRS

Table A- 244 Technical specifications of the CP 1242-7 GPRS V2

Technical specifications	
Article number	6GK7 242-7KX3-0XE0
Wireless interface	
Antenna connector	SMA socket
Nominal impedance	50 ohms
Wireless connection	
Maximum transmit power	• GSM 850, class 4: +33 dBm ±2dBm • GSM 900, class 4: +33 dBm ±2dBm • GSM 1800, class 1: +30 dBm ±2dBm • GSM 1900, class 1: +30 dBm ±2dBm

Technical specifications	
GPRS	Multislot class 10 device class B coding scheme 1...4 (GMSK)
SMS	Mode outgoing: MO service: point-to-point
Permitted ambient conditions	
Ambient temperature • during storage • during transportation • during operation with a vertical installation (DIN rail horizontal) • during operation with a horizontal installation (DIN rail vertical)	 • -40 °C to 70 °C • -40 °C to 70 °C • 0 °C to 55 °C • 0 °C to 45 °C
Relative humidity at 25 °C during operation, without condensation, maximum	95 %
Degree of protection	IP20
Power supply, current consumption and power loss	
Type of power supply	DC
Power supply / external • minimum • maximum	24 V • 19.2 V • 28.8 V
Current consumption (typical) • from 24 V DC • from the S7-1200 backplane bus	 • 100 mA • 0 mA
Effective power loss (typical) • from 24 V DC • from the S7-1200 backplane bus	 • 2.4 W • 0 W
24 V DC power supply • Min. cable cross section • Max. cable cross section • Tightening torque of the screw terminals	 • min.: 0.14 mm^2 (AWG 25) • max.: 1.5 mm^2 (AWG 15) • 0.45 Nm (4 lb-in)
Electrical isolation Power supply unit to internal circuit	710 VDC for 1 minute
Dimensions and weights	
• Width • Height • Depth	• 30 mm • 100 mm • 75 mm
Weight • Net weight • Weight including packaging	 • 133 g • 170 g

A.14.2.2 GSM/GPRS antenna ANT794-4MR

Technical specifications of the ANT794-4MR GSM/GPRS antenna

ANT794-4MR	
Article number	6NH9860-1AA00
Mobile wireless networks	GSM/GPRS
Frequency ranges	• 824 to 960 MHz (GSM 850, 900) • 1 710 to 1 880 MHz (GSM 1 800) • 1 900 to 2 200 MHz (GSM / UMTS)
Characteristics	omnidirectional
Antenna gain	0 dB
Impedance	50 ohms
Standing wave ratio (SWR)	< 2,0
Max. power	20 W
Polarity	linear vertical
Connector	SMA
Length of antenna cable	5 m
External material	Hard PVC, UV-resistant
Degree of protection	IP20
Permitted ambient conditions • Operating temperature • Transport/storage temperature • Relative humidity	 • -40 °C through +70 °C • -40 °C through +70 °C • 100 %
External material	Hard PVC, UV-resistant
Construction	Antenna with 5 m fixed cable and SMA male connector
Dimensions (D x H) in mm	25 x 193
Weight • Antenna incl. cable • Fittings	 • 310 g • 54 g
Installation	With supplied bracket

A.14.2.3 Flat antenna ANT794-3M

Technical specifications of the flat antenna ANT794-3M

ANT794-3M		
Article number	6NH9870-1AA00	
Mobile wireless networks	**GSM 900**	**GSM 1800/1900**
Frequency ranges	890 - 960 MHz	1710 - 1990 MHz
Standing wave ratio (VSWR)	≤ 2:1	≤ 1,5:1
Return loss (Tx)	≈ 10 dB	≈ 14 dB
Antenna gain	0 dB	
Impedance	50 ohms	
Max. power	10 W	
Antenna cable	HF cable RG 174 (fixed) with SMA male connector	
Cable length	1.2 m	
Degree of protection	IP64	
Permitted temperature range	-40°C to +75°C	
Flammability	UL 94 V2	
External material	ABS Polylac PA-765, light gray (RAL 7035)	
Dimensions (W x L x H) in mm	70.5 x 146.5 x 20.5	
Weight	130 g	

A.14.3 CM 1243-2 AS-i master

A.14.3.1 Technical data for the AS-i master CM 1243-2

Table A- 245 Technical data for the AS-i master CM 1243-2

Technical data	
Article number	3RK7243-2AA30-0XB0
Firmware version	V1.0
Date	01.12.2011
Interfaces	
Maximum current consumption From the S7-1200 backplane bus From the AS-i cable	 Max. 250 mA, supply voltage S7-1200 communication bus 5 V DC Max. 100 mA
Maximum current carrying capacity between the ASI+/ASI- terminals	8 A
Pin assignment	See section Electrical connections of the AS-i master (Page 1259)
Conductor cross-section	0.2 mm² (AWG 24) ... 3.3 mm² (AWG 12)

Technical data	
ASI connector tightening torque	0.56 Nm
Permissible ambient conditions	
Ambient temperature During storage During transport During the operating phase, with vertical installation (horizontal standard mounting rail) During the operating phase, with horizontal installation (vertical standard mounting rail)	 -40 °C ... 70 °C -40 °C ... 70 °C 0 °C ... 55 °C 0 °C ... 45 °C
Relative humidity at 25 °C during operating phase, no condensation, maximum	95 %
Degree of protection	IP20
Power supply, current consumption, power loss	
Type of power supply	DC
Current consumption (typically) From the S7-1200 backplane bus	 200 mA
Total power loss (typical): • From the S7-1200 backplane bus • From AS-i cable	 1 W 2.4 W
Dimensions and weights	
Width Height Depth	30 mm 100 mm 75 mm
Weight Net weight Weight including packaging	 122 g 159 g

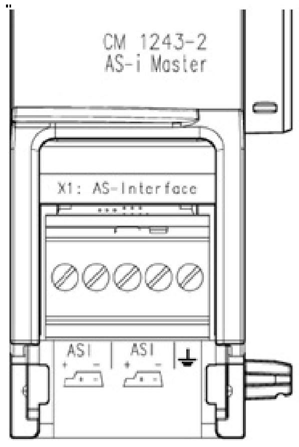
A.14.3.2 Electrical connections of the AS-i master

Power supply of the AS-i master CM 1243-2

The AS-i master CM 1243-2 is supplied over the communications bus of the S7-1200. This means that a diagnostics message can still be sent to the S7-1200 following failure of the AS-i supply voltage. The connection to the communications bus is on the right-hand side of the AS-i master CM 1243-2.

AS-Interface terminals

The removable terminal for connecting the AS-i cable is located behind the lower cover on the front of the AS-i master CM 1243-2.



If the AS-i shaped cable is used, you can recognize the correct polarity of the cable by means of the symbol

Information on how to remove and re-install the terminal block can be found in the Installation chapter (Page 65).

Note

Maximum current carrying capacity of the terminal contacts

The current carrying capacity of the connection contacts is max. 8 A. If this value is exceeded on the AS-i cable, the AS-i master CM 1243-2 must not be "looped in" to the AS-i cable, but must instead be connected via a spur line (only one connection pair assigned on the AS-i master CM 1243-2).
Please also ensure that the cables used are suitable for operating temperatures of at least 75°C if current is being conducted via the AS-i master and currents of greater than 4 amperes are present.
You will find additional information on connecting the AS-i cable in the section "Installation, connection and commissioning of the modules" in the manual "AS-i Master CM 1243-2 and AS-i data decoupling unit DCM 1271 for SIMATIC S7-1200".

Terminal assignment

Label	Meaning
ASI+	AS-i connection – positive polarity
ASI–	AS-i connection – negative polarity
⏚	Functional ground

A.14.4 RS232, RS422, and RS485

A.14.4.1 CB 1241 RS485 specifications

Note

To use this CB, your CPU firmware must be V2.0 or higher.

Table A- 246 General specifications

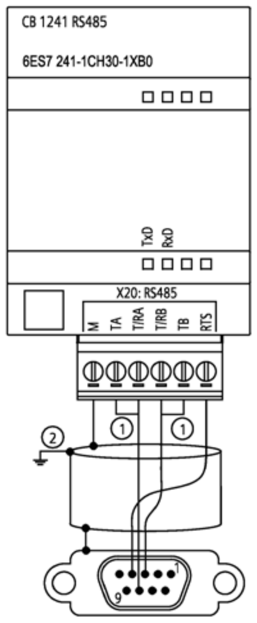
Technical data	CB 1241 RS485
Article number	6ES7 241-1CH30-1XB0
Dimensions W x H x D (mm)	38 x 62 x 21
Weight	40 grams

Table A- 247 Transmitter and receiver

Technical data	CB 1241 RS485
Type	RS485 (2-wire half-duplex)
Common mode voltage range	-7 V to +12 V, 1 second, 3 VRMS continuous
Transmitter differential output voltage	2 V min. at R_L = 100 Ω 1.5 V min. at R_L = 54 Ω
Termination and bias	10K to +5 V on B, RS485 Pin 3 10K to GND on A, RS485 Pin 4
Optional termination	Short Pin TB to Pin T/RB, effective termination impedance is 127 Ω, connects to RS485 Pin 3 Short Pin TA to Pin T/RA, effective termination impedance is 127 Ω, connects to RS485 Pin 4
Receiver input impedance	5.4K Ω min. including termination
Receiver threshold/sensitivity	+/- 0.2 V min., 60 mV typical hysteresis
Isolation RS485 signal to chassis ground RS485 signal to CPU logic common	500 VAC, 1 minute
Cable length, shielded	1000 m max.
Baud rate	300 baud, 600 baud, 1.2 kbits, 2.4 kbits, 4.8 kbits, 9.6 kbits (default), 19.2 kbits, 38.4 kbits, 57.6 kbits, 76.8 kbits, 115.2 kbits,
Parity	No parity (default), even, odd, Mark (parity bit always set to 1), Space (parity bit always set to 0)
Number of stop bits	1 (default), 2
Flow control	Not supported
Wait time	0 to 65535 ms

Table A- 248 Power supply

Technical data	CB 1241 RS485
Power loss (dissipation)	1.5 W
Current consumption (SM Bus), max.	50 mA
Current consumption (24 VDC) max.	80 mA

CB 1241 RS485 (6ES7 241-1CH30-1XB0)	
 CB 1241 RS485 6ES7 241-1CH30-1XB0 TxD RxD X20: RS485 M TA T/RA T/RB TB RTS ① ① ② 1 9	
① Connect "TA" and TB" as shown to terminate the network. (Terminate only the end devices on the RS485 network.)	
② Use shielded twisted pair cable and connect the cable shield to ground.	

You terminate only the two ends of the RS485 network. The devices in between the two end devices are not terminated or biased. See the topic "Biasing and terminating an RS485 network connector" (Page 862)

Table A- 249 Connector pin locations for CB 1241 RS485 (6ES7 241-1CH30-1XB0)

Pin	9-Pin connector	X20
1	RS485 / Logic GND	--
2	RS485 / Not Used	--
3	RS485 / TxD+	3 - T/RB
4	RS485 / RTS	1 - RTS
5	RS485 / Logic GND	--
6	RS485 / 5 V Power	--
7	RS485 / Not used	--
8	RS485 / TxD-	4 - T/RA
9	RS485 / Not Used	--
Shell		7 - M

A.14.4.2 CM 1241 RS232 specifications

Table A- 250 General specifications

Technical data	**CM 1241 RS232**
Article number	6ES7 241-1AH32-0XB0
Dimensions (mm)	30 x 100 x 75
Weight	150 grams

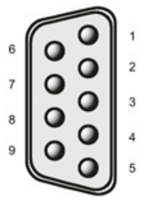
Table A- 251 Transmitter and receiver

Technical data	**CM 1241 RS232**
Type	RS232 (full-duplex)
Transmitter output voltage	+/- 5 V min. at R_L = 3K Ω
Transmit output voltage	+/- 15 VDC max.
Receiver input impedance	3 K Ω min.
Receiver threshold/sensitivity	0.8 V min. low, 2.4 max. high 0.5 V typical hysteresis
Receiver input voltage	+/- 30 VDC max.
Isolation RS 232 signal to chassis ground RS 232 signal to CPU logic common	500 VAC, 1 minute
Cable length, shielded	10 m max.
Baud rate	300 baud, 600 baud, 1.2 kbits, 2.4 kbits, 4.8 kbits, 9.6 kbits (default), 19.2 kbits, 38.4 kbits, 57.6 kbits, 76.8 kbits, 115.2 kbits,
Parity	No parity (default), even, odd, Mark (parity bit always set to 1), Space (parity bit always set to 0)
Number of stop bits	1 (default), 2
Flow control	Hardware, software
Wait time	0 to 65535 ms

Table A- 252 Power supply

Technical data	**CM 1241 RS232**
Power loss (dissipation)	1 W
From +5 VDC	200 mA

Table A- 253 RS232 connector (male)

Pin	Description	Connector (male)	Pin	Description
1 DCD	Data carrier detect: Input		6 DSR	Data set ready: Input
2 RxD	Received data from DCE: Input		7 RTS	Request to send: Output
3 TxD	Transmitted data to DCE: Output		8 CTS	Clear to send: Input
4 DTR	Data terminal ready: Output		9 RI	Ring indicator (not used)
5 GND	Logic ground		SHELL	Chassis ground

A.14.4.3 CM 1241 RS422/485 specifications

CM 1241 RS422/485 Specifications

Table A- 254 General specifications

Technical data	CM 1241 RS422/485
Article number	6ES7 241-1CH32-0XB0
Dimensions W x H x H (mm)	30 x 100 x 75
Weight	155 grams

Table A- 255 Transmitter and receiver

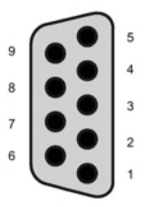
Technical data	CM 1241 RS422/485
Type	RS422 or RS485, 9-pin sub D female connector
Common mode voltage range	-7 V to +12 V, 1 second, 3 VRMS continuous
Transmitter differential output voltage	2 V min. at R_L = 100 Ω 1.5 V min. at R_L = 54 Ω
Termination and bias	10K Ω to +5 V on B, PROFIBUS Pin 3 10K Ω to GND on A, PROFIBUS Pin 8 Internal bias options provided, or no internal bias. In all cases, external termination is required, see Biasing and terminating an RS485 network connector (Page 862) and Configuring the RS422 and RS485 in the S7-1200 Programmable Controller System Manual (Page 905)
Receiver input impedance	5.4K Ω min. including termination
Receiver threshold/sensitivity	+/- 0.2 V min., 60 mV typical hysteresis
Isolation RS485 signal to chassis ground RS485 signal to CPU logic common	500 VAC, 1 minute
Cable length, shielded	1000 m max. (baud rate dependent)
Baud rate	300 baud, 600 baud, 1.2 kbits, 2.4 kbits, 4.8 kbits, 9.6 kbits (default), 19.2 kbits, 38.4 kbits, 57.6 kbits, 76.8 kbits, 115.2 kbits,
Parity	No parity (default), even, odd, Mark (parity bit always set to 1), Space (parity bit always set to 0)

Technical data	CM 1241 RS422/485
Number of stop bits	1 (default), 2
Flow control	XON/XOFF supported for the RS422 mode
Wait time	0 to 65535 ms

Table A- 256 Power supply

Technical data	CM 1241 RS422/485
Power loss (dissipation)	1.1 W
From +5 VDC	220 mA

Table A- 257 RS485 or RS422 connector (female)

Pin	Description	Connector (female)	Pin	Description
1	Logic or communication ground		6 PWR	+5 V with 100 ohm series resistor: Output
2 TxD+ [1]	Connected for RS422 Not used for RS485: Output		7	Not connected
3 TxD+	Signal B (RxD/TxD+): Input/Output		8 TXD-	Signal A (RxD/TxD-): Input/Output
4 RTS [2]	Request to send (TTL level) Output		9 TXD- [1]	Connected for RS422 Not used for RS485: Output
5 GND	Logic or communication ground		SHELL	Chassis ground

[1] Pins 2 and 9 are only used as transmit signals for RS422.

[2] The RTS is a TTL level signal and can be used to control another half duplex device based on this signal. It is active when you transmit and is inactive all other times.

A.15 TeleService (TS Adapter and TS Adapter modular)

The following manuals contain the technical specification for the TS Adapter IE Basic and the TS Adapter modular:

- Industrial Software Engineering Tools
  Modular TS Adapter
- Industrial Software Engineering Tools
  TS Adapter IE Basic

For more information about this product and for the product documentation, refer to the product catalog web site for the TS Adapter (https://eb.automation.siemens.com/mall/en/de/Catalog/Search?searchTerm=TS%20Adapter%20IE%20basic&tab=).

A.16 SIMATIC memory cards

Article Number	Capacity
6ES7 954-8LP01-0AA0	2 GB
6ES7 954-8LL02-0AA0	256 MB
6ES7 954-8LF02-0AA0	24 MB
6ES7 954-8LE02-0AA0	12 MB
6ES7 954-8LC02-0AA0	4 MB

A.17 Input simulators

Table A- 258 General specifications

Technical data	8 Position Simulator	14 Position Simulator	CPU 1217C Simulator
Article number	6ES7 274-1XF30-0XA0	6ES7 274-1XH30-0XA0	6ES7 274-1XK30-0XA0
Dimensions W x H x D (mm)	43 x 35 x 23	67 x 35 x 23	93 x 40 x 23
Weight	20 grams	30 grams	43 grams
Points	8	14	14
Used with CPU	CPU 1211C, CPU 1212C	CPU 1214C, CPU 1215C	CPU 1217C

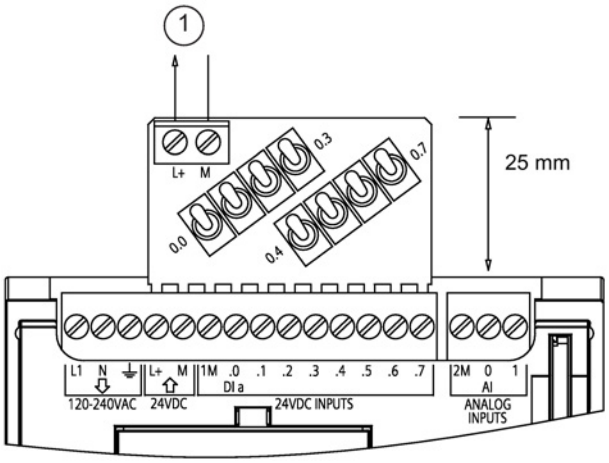
⚠ WARNING

Safe use of input simulators

These input simulators are not approved for use in Class I DIV 2 or Class I Zone 2 hazardous locations. The switches present a potential spark hazard/explosion hazard if used in a Class I DIV 2 or Class I Zone 2 location. Unapproved use could result in death or serious injury to personnel, and/or damage to equipment.

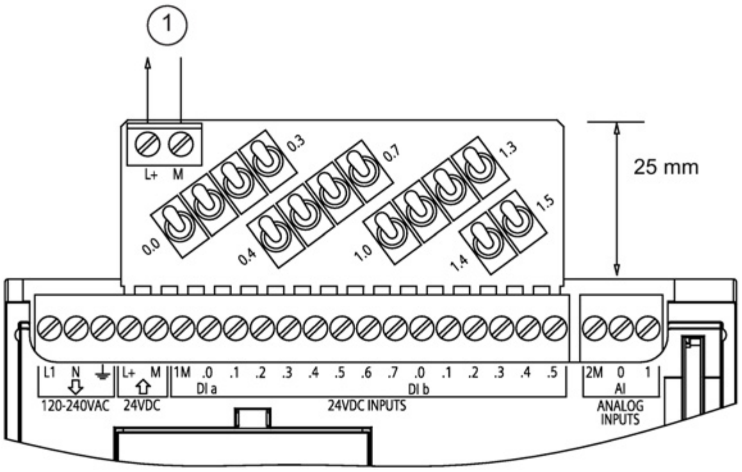
Use these input simulators only in non-hazardous locations. Do not use in Class I DIV 2 or Class I Zone 2 hazardous locations.

8 Position Simulator (6ES7 274-1XF30-0XA0)



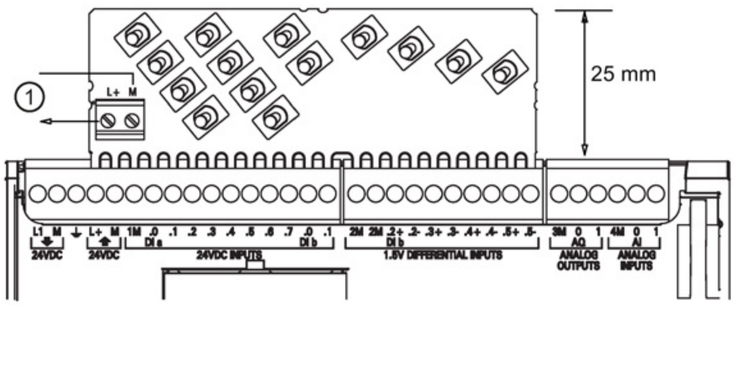
① 24 VDC sensor power out

14 Position Simulator (6ES7 274-1XF30-0XA0)



① 24 VDC sensor power out

CPU 1217C Simulator (6ES7 274-1XK30-0XA0)



① 24 VDC sensor power out

A.18 S7-1200 Potentiometer module

The S7-1200 Potentiometer module is an accessory for S7-1200 CPU. Each potentiometer creates an output voltage proportional to the position of the potentiometer to drive each of the two CPU analog inputs 0 VDC to 10 VDC. To install the potentiometer:

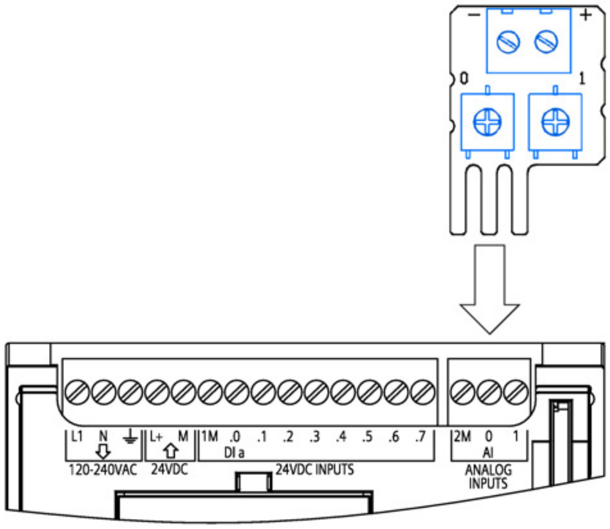
1. Insert the circuit board 'fingers' into any S7-1200 CPU analog input terminal block, and connect an external DC power supply to the 2-position connector on the potentiometer module.
2. Use a small screwdriver to make the adjustments: turn the potentiometer clockwise (to the right) to increase the voltage output, and counterclockwise (to the left) to decrease the voltage output.

Note

Follow ESD guidelines when handling the S7-1200 Potentiometer module.

Technical data	S7-200 Potentiometer module
Article number	6ES7 274-1XA30-0XA0
Used with CPU	All S7-1200 CPUs
Number of potentiometers	2
Dimensions W x H x D (mm)	20 x 33 x 14
Weight	26 grams
User-supplied voltage input at 2-position connector[1] (Class 2, Limited Power, or sensor power from PLC)	16.4 VDC to 28.8 VDC
Cable length (meters)/type	<30 m, shielded twisted pair
Input current consumption	10 mA max.
Potentiometer voltage output to S7-1200 CPU analog inputs[1]	0 VDC to 10.5 VDC min.
Isolation	Not isolated
Ambient temperature range	-20 °C to 60 °C

[1] Potentiometer module output voltage stability depends on the quality of the user-supplied voltage input at the 2-position connector - consider it as an analog input voltage.

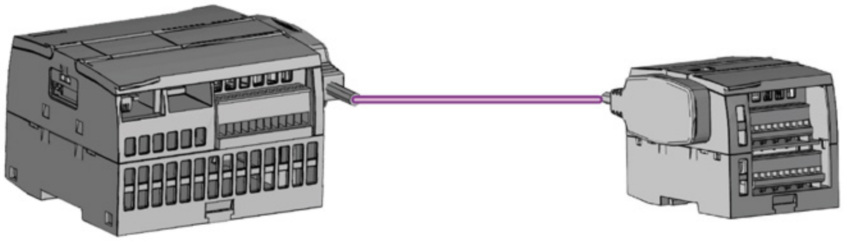


A.19 I/O expansion cable

Table A- 259 Expansion cables

Technical Data	
Article number	6ES7 290-6AA30-0XA0
Cable length	2 m
Weight	200 g

Refer to the installation section (Page 66) for information about installing and removing the S7-1200 expansion cable.



A.20 Companion products

A.20.1 PM 1207 power module

The PM 1207 is a power supply module for the SIMATIC S7-1200. It provides the following features:

- Input 120/230 VAC, output 24 VDC/2.5A
- Article number 6ESP 332-1SH71-4AA0

For more information about this product and for the product documentation, refer to the product catalog web site for the PM 1207 (https://eb.automation.siemens.com/mall/en/de/Catalog/Product/6AG1332-1SH71-4AA0).

A.20.2 CSM 1277 compact switch module

The CSM1277 is an Industrial Ethernet compact switch module. It can be used to multiply the Ethernet interface of the S7-1200 to allow simultaneous communication with operator panels, programming devices, or other controllers. It provides the following features:

- 4 x RJ45 sockets for connecting to Industrial Ethernet
- 3 pole plug in terminal strip for connection of the external 24 VDC supply on top
- LEDs for diagnostics and status display of Industrial Ethernet ports
- Article number 6GK7 277-1AA00-0AA0

For more information about this product and for the product documentation, refer to the product catalog web site for the CSM 1277 (https://eb.automation.siemens.com/mall/en/de/Catalog/Search?searchTerm=csm%201277&tab=).

A.20.3 CM CANopen module

The CM CANopen module is a plug-in module between the SIMATIC S7-1200 PLC and any device running CANopen. The CM CANopen can be configured to be both master or slave. There are two CM CANopen modules: the CANopen module (article number 021620-B), and the CANopen (Ruggedized) module (article number 021730-B).

The CANopen module provides the following features:

- Able to connect 3 modules per CPU
- Connects up to 16 CANopen slave nodes
- 256 byte input and 256 byte output per module
- 3 LEDs provide diagnostic information on module, network, and I/O status
- Supports storage of CANopen network configuration in the PLC
- The module is integratable in the hardware catalogue of the TIA Portal configuration suite
- CANopen configuration via included CANopen Configuration Studio (included) or via any other externanal CANopen configuration tool
- Complies to the CANopen communication profiles CiA 301 rev. 4.2 and the CiA 302 rev. 4.1
- Supports transparent CAN 2.0A for custom protocol handling
- Pre-made function blocks available for each PLC programming in TIA portal
- CM CANopen modules include; DSUB with screw terminals for subnetwork. CM CANopen configuration studio CD, and USB configuration cable

For more information about this product and for the product documentation, refer to the product catalog web site for the CM CANopen.

A.20.4 RF120C communications module

The RF10C allows Siemens RFID and code reading systems to be connected directly and easily to an S7-1200. The reader is connected to the RF120C via a point-to-point connecton. Up to three communications modules can be conneded to an S7-1200 to the left of the CPU. The RF120C comminications module is configured via the TIA Portal. The article number for the RF120C communications module is 6GT2002-0LA00.

For more information about this product and for the product documentation, refer to the product catalog web site for the RF120C.

Calculating a power budget B

The CPU has an internal power supply that provides power for the CPU itself, for any expansion modules, and for other 24 VDC user power requirements.

There are four types of expansion modules:

- Signal modules (SM) are installed on the right-side of the CPU. Each CPU allows a maximum number of signal modules possible without regard to the power budget.
  - CPU 1214C, CPU 1215C and CPU 1217C allows 8 signal modules
  - CPU 1212C allows 2 signal modules
  - CPU 1211C allows no signal modules
- Communication modules (CM) are installed on the left-side of the CPU. A maximum of 3 communication modules is allowed for any CPU without regard to the power budget.
- Signal boards (SB), communications boards (CB), and battery boards (BB) are installed on top of the CPU. A maximum of 1 signal board, communication board, or battery board is allowed for any CPU.

Use the following information as a guide for determining how much power (or current) the CPU can provide for your configuration.

Each CPU supplies both 5 VDC and 24 VDC power:

- The CPU provides 5 VDC power for the expansion modules when an expansion module is connected. If the 5 VDC power requirements for expansion modules exceed the power budget of the CPU, you must remove expansion modules until the requirement is within the power budget.
- Each CPU has a 24 VDC sensor supply that can supply 24 VDC for local input points or for relay coils on the expansion modules. If the power requirement for 24 VDC exceeds the power budget of the CPU, you can add an external 24 VDC power supply to provide 24 VDC to the expansion modules. You must manually connect the 24 VDC supply to the input points or relay coils.

WARNING

Connecting an external 24 VDC power supply in parallel with the DC sensor supply 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, severe personal injury and/or property damage.

The DC sensor supply on 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 VDC power input ports in the PLC system are interconnected, with a logic common circuit connecting multiple M terminals. The CPU 24 VDC power supply input, the SM relay coil power input, and a non-isolated analog power supply input are examples of circuits that are interconnected when designated as not isolated in the data sheets. All non-isolated M terminals must connect to the same external reference potential.

WARNING

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 connected equipment.

Such damage or unpredictable operation could result in death, severe personal injury and/or property damage.

Always be sure that all non-isolated M terminals in a PLC system are connected to the same reference potential.

Information about the power budgets of the CPUs and the power requirements of the signal modules is provided in the technical specifications (Page 1099).

Note

Exceeding the power budget of the CPU may result in not being able to connect the maximum number of modules allowed for your CPU.

Example power budget

The following example shows a sample calculation of the power requirements for a configuration that includes one CPU 1214C AC/DC/Relay, one SB 1223 2 x 24 VDC Input/ 2 x 24 VDC Output, one CM 1241, three SM 1223 8 DC In/8 Relay Out, and one SM 1221 8 DC In. This example has a total of 48 inputs and 36 outputs.

Note

The CPU has already allocated the power required to drive the internal relay coils. You do not need to include the internal relay coil power requirements in a power budget calculation.

The CPU in this example provides sufficient 5 VDC current for the SMs, but does not provide enough 24 VDC current from the sensor supply for all of the inputs and expansion relay coils. The I/O requires 456 mA and the CPU provides only 400 mA. This installation requires an additional source of at least 56 mA at 24 VDC power to operate all the included 24 VDC inputs and outputs.

Table B- 1 Sample power budget

CPU power budget	5 VDC	24 VDC
CPU 1214C AC/DC/Relay	1600 mA	400 mA
	Minus	
System requirements	**5 VDC**	**24 VDC**
CPU 1214C, 14 inputs	-	14 * 4 mA = 56 mA
1 SB 1223 2 x 24 VDC Input/ 2 x 24 VDC Output	50 mA	2 * 4 mA = 8 mA
1 CM 1241 RS422/485, 5 V power	220 mA	
3 SM 1223, 5 V power	3 * 145 mA = 435 mA	-
1 SM 1221, 5 V power	1 * 105 mA = 105 mA	-
3 SM 1223, 8 inputs each	-	3 * 8 * 4 mA = 96 mA
3 SM 1223, 8 relay coils each	-	3 * 8 * 11 mA = 264 mA
1 SM 1221, 8 inputs each	-	8 * 4 mA = 32 mA
Total requirements	810 mA	456 mA
	Equals	
Current balance	**5 VDC**	**24 VDC**
Current balance total	790 mA	(56 mA)

Form for calculating your power budget

Use the following table to determine how much power (or current) the S7-1200 CPU can provide for your configuration. Refer to the technical specifications (Page 1099) for the power budgets of your CPU model and the power requirements of your signal modules.

Table B- 2 Calculations for a power budget

CPU power budget	5 VDC	24 VDC
	Minus	
System requirements	**5 VDC**	**24 VDC**
Total requirements		
	Equals	
Current balance	**5 VDC**	**24 VDC**
Current balance total		

Ordering Information C

C.1 CPU modules

Table C- 1 S7-1200 CPUs

CPU models		Article number
CPU 1211C	CPU 1211C DC/DC/DC	6ES7 211-1AE40-0XB0
	CPU 1211C AC/DC/Relay	6ES7 211-1BE40-0XB0
	CPU 1211C DC/DC/Relay	6ES7 211-1HE40-0XB0
CPU 1212C	CPU 1212C DC/DC/DC	6ES7 212-1AE40-0XB0
	CPU 1212C AC/DC/Relay	6ES7 212-1BE40-0XB0
	CPU 1212C DC/DC/Relay	6ES7 212-1HE40-0XB0
CPU 1214C	CPU 1214C DC/DC/DC	6ES7 214-1AG40-0XB0
	CPU 1214C AC/DC/Relay	6ES7 214-1BG40-0XB0
	CPU 1214C DC/DC/Relay	6ES7 214-1HG40-0XB0
CPU 1215C	CPU 1215C DC/DC/DC	6ES7 215-1AG40-0XB0
	CPU 1215C AC/DC/Relay	6ES7 215-1BG40-0XB0
	CPU 1215C DC/DC/Relay	6ES7 215-1HG40-0XB0
CPU 1217C	CPU 1217C DC/DC/DC	6ES7 217-1AG40-0XB0

C.2 Signal modules (SMs), signal boards (SBs), and battery boards (BB)

Table C- 2 Signal modules (SMs)

Signal modules		Article number
Digital input	SM 1221 8 x 24 VDC Input (Sink/Source)	6ES7 221-1BF32-0XB0
	SM 1221 16 x 24 VDC Input (Sink/Source)	6ES7 221-1BH32-0XB0
Digital output	SM 1222 8 x 24 VDC Output (Source)	6ES7 222-1BF32-0XB0
	SM 1222 16 x 24 VDC Output (Source)	6ES7 222-1BH32-0XB0
	SM 1222 8 x Relay Output	6ES7 222-1HF32-0XB0
	SM 1222 8 x Relay Output (Changeover)	6ES7 222-1XF32-0XB0
	SM 1222 16 x Relay Output	6ES7 222-1HH32-0XB0
Digital input / output	SM 1223 8 x 24 VDC Input (Sink/Source) / 8 x 24 VDC Output (Source)	6ES7 223-1BH32-0XB0
	SM 1223 16 x 24 VDC Input (Sink/Source) / 16 x 24 VDC Output (Source)	6ES7 223-1BL32-0XB0
	SM 1223 8 x 24 VDC Input (Sink/Source) / 8 x Relay Output	6ES7 223-1PH32--0XB0
	SM 1223 16 x 24 VDC Input (Sink/Source) / 16 x Relay Output	6ES7 223-1PL32-0XB0
	SM 1223 8 x 120/230 VAC Input (Sink/Source) / 8 x Relay Outputs	6ES7 223-1QH32-0XB0
Analog input	SM 1231 4 x Analog Input	6ES7 231-4HD32-0XB0
	SM 1231 8 x Analog Input	6ES7 231-4HF32-0XB0
	SM 1231 4 x Analog Input x 16 bit (high feature)	6ES7 231-5ND32-0XB0
Analog output	SM 1232 2 x Analog Output	6ES7 232-4HB32-0XB0
	SM 1232 4 x Analog Output	6ES7 232-4HD32-0XB0
Analog input / output	SM 1234 4 x Analog Input / 2 x Analog Output	6ES7 234-4HE32-0XB0
RTD and thermocouple	SM 1231 TC 4 x 16 bit	6ES7 231-5QD32-0XB0
	SM 1231 TC 8 x 16 bit	6ES7 231-5QF32-0XB0
	SM 1231 RTD 4 x 16 bit	6ES7 231-5PD32-0XB0
	SM 1231 RTD 8 x 16 bit	6ES7 231-5PF32-0XB0
Technology modules	SM 1278 4xIO-Link Master	6ES7 278-4BD32-0XB0

Table C- 3 Signal boards (SB) and battery boards (BBs)

Signal and battery boards		Article number
Digital input	SB 1221 200 kHz 4 x 24 VDC Input (Source)	6ES7 221-3BD30-0XB0
	SB 1221 200 kHz 4 x 5 VDC Input (Source)	6ES7 221-3AD30-0XB0
Digital output	SB 1222 200 kHz 4 x 24 VDC Output (Sink/Source)	6ES7 222-1BD30-0XB0
	SB 1222 200 kHz 4 x 5 VDC Output (Sink/Source)	6ES7 222-1AD30-0XB0
Digital input / output	SB 1223 2 x 24 VDC Input (Sink) / 2 x 24 VDC Output (Source)	6ES7 223-0BD30-0XB0
	SB 1223 200 kHz 2 x 24 VDC Input (Source) / 2 x 24 VDC Output (Sink/Source)	6ES7 223-3BD30-0XB0
	SB 1223 200 kHz 2 x 5 VDC Input (Source) / 2 x 5 VDC Output (Sink/Source)	6ES7 223-3AD30-0XB0
Analog	SB 1232 1 Analog Output	6ES7 232-4HA30-0XB0
	SB 1231 1 Analog Input	6ES7 231-4HA30-0XB0
	SB 1231 1 Analog Input Thermocouple	6ES7 231-5QA30-0XB0
	SB 1231 1 Analog Input RTD	6ES7 231-5PA30-0XB0
Battery	BB 1297 Battery Board (battery type CR1025 not included)	6ES7 297-0AX30-0XA0

C.3 Communication

Table C- 4 Communication module (CM)

Communication module (CM)			Article number
RS232, RS422, and RS485	CM 1241 RS232	RS232	6ES7 241-1AH32-0XB0
	CM 1241 RS422/485	RS422/485	6ES7 241-1CH32-0XB0
PROFIBUS	CM 1243-5	PROFIBUS Master	6GK7 243-5DX30-0XE0
	CM 1242-5	PROFIBUS Slave	6GK7 242-5DX30-0XE0
AS-i Master	CM 1243-2	AS-i Master	3RK7 243-2AA30-0XB0
RF120C CM	RF120C	RFID reader	6GT2 002-0LA00

Table C- 5 Communication board (CB)

Communication board (CB)			Article number
RS485	CB 1241 RS485	RS485	6ES7 241-1CH30-1XB0

Table C- 6 Communication Processor (CP)

CP	Interface	Article number
CP 1242-7 GPRS V2	GPRS	6GK7 242-7KX31-0XE0
CP 1243-7 LTE-EU	LTE	6GK7 243-7KX30-0XE0
CP 1243-1 DNP3	IE-interface	6GK7 243-1JX30-0XE0
CP 1243-1 IEC	IE-interface	6GK7 243-1PX30-0XE0
CP 1243-1	IE-interface	6GK7 243-1BX30-0XE0
CP 1243-1 PCC	IE-interface	6GK7 243-1HX30-0XE0
CP 1243-8 ST7	IE- and serial interface	6GK7 243-8RX30-0XE0

Table C- 7 TeleService

TS Adapter	Article number
TS Adapter IE Basic	6ES7 972-0EB00-0XA0
TS Adapter IE Advanced	6ES7 972-0EA00-0XA0
TS Module GSM	6GK7 972-0MG00-0XA0
TS Module RS232	6ES7 792-0MS00-0XA0
TS Module Modem	6ES7 972-0MM00-0XA0
TS Module ISDN	6ES7 972-0MD00-0XA0

Table C- 8 Accessories

Accessory			Article number
Antenna	ANT794-4MR	GSM/GPRS antenna	6NH9 860-1AA00
	ANT794-3M	Flat antenna	6NH9 870-1AA00

Table C- 9 Connectors

Type of Connector		Article number
RS485	35-degree cable output, screw-terminal connection	6ES7 972-0BA42-0XA0
	35-degree cable output, FastConnect connection	6ES7 972-0BA60-0XA0

C.4 Fail-Safe CPUs and signal modules

Table C- 10 Fail-Safe CPUs

Fail-Safe CPU models		Article number
CPU 1214FC	CPU 1214FC DC/DC/DC	6ES7 214-1AF40-0XB0
	CPU 1214FC DC/DC/Relay	6ES7 214-1HF40-0XB0
CPU 1215FC	CPU 1215FC DC/DC/DC	6ES7 215-1AF40-0XB0
	CPU 1215FC DC/DC/Relay	6ES7 215-1HF40-0XB0

Table C- 11 Fail-Safe signal modules

Functional Safety signal modules		Article number
Digital input	SM 1226 F-DI 16 x 24 VDC	6ES7 226-6BA32-0XB0
Digital output	SM 1226 F-DQ 4 x 24 VDC	6ES7 226-6DA32-0XB0
	SM 1226 F-DQ 2 x Relay	6ES7 226-6RA32-0XB0

C.5 Other modules

Table C- 12 Companion products

Item		Article number
Power supply	PM 1207 power supply	6EP1 332-1SH71-4AA0
Ethernet switch	CSM 1277 Ethernet switch - 4 ports	6GK7 277-1AA10-0AA0
CM CANopen	CANopen for SIMATIC S7-1200	021620-B
	CANopen (Ruggedized) for SIMATIC S7-1200	021730-B
RF120C	RF120C communications module	6GT2002-0LA00

C.6 Memory cards

Table C- 13 Memory cards

SIMATIC memory cards	Article number
SIMATIC MC 2 GB	6ES7 954-8LP01-0AA0
SIMATIC MC 256 MB	6ES7 954-8LL02-0AA0
SIMATIC MC 24 MB	6ES7 954-8LF02-0AA0
SIMATIC MC 12 MB	6ES7 954-8LE02-0AA0
SIMATIC MC 4 MB	6ES7 954-8LC02-0AA0

C.7 Basic HMI devices

Table C- 14 HMI devices

HMI Basic Panels	Article number
KTP400 Basic (Mono, PN)	6AV2 123-2DB03-0AX0
KTP700 Basic	6AV2 123-2GB03-0AX0
KTP700 Basic DP	6AV2 123-2GA03-0AX0
KTP900 Basic	6AV2 123-2JB03-0AX0
KTP1200 Basic	6AV2 123-2MB03-0AX0
KTP1200 Basic DP	6AV2 123-2MA03-0AX0

C.8 Spare parts and other hardware

Table C- 15 Expansion cables, simulators, and end retainers

Item		Article number
I/O expansion cable	I/O Expansion cable, 2 m	6ES7 290-6AA30-0XA0
I/O simulator	Simulator (1211C/1212C - 8 position)	6ES7 274-1XF30-0XA0
	Simulator (1214C/1215C - 14 position)	6ES7 274-1XH30-0XA0
	Simulator, CPU 1217C	6ES7 274-1XK30-0XA0
Potentiometer module	S7-1200 Potentiometer module	6ES7 274-1XA30-0XA0
Ethernet strain relief	Single port RJ45 strain relief	6ES7 290-3AA30-0XA0
	Dual port RJ45 strain relief	6ES7 290-3AB30-0XA0
Spare door kit	CPU 1211C/1212C	6ES7 291-1AA30-0XA0
	CPU 1214C	6ES7 291-1AB30-0XA0
	CPU 1215C	6ES7 291-1AC30-0XA0
	CPU 1217C	6ES7 291-1AD30-0XA0

Item		Article number
	Signal module, 45 mm	6ES7 291-1BA30-0XA0
	Signal module, 70 mm	6ES7 291-1BB30-0XA0
	Communication module (for use with 6ES7 2xx-xxx32-0XB0 and 6ES7 2xx-xxx30-0XB0 modules)	6ES7 291-1CC30-0XA0
End Retainer	End Retainer Thermoplastic, 10 MM	8WA1808
	End Retainer Steel, 10.3 MM	8WA1805

Table C- 16 S7-1200 CPU V4.0 and later - Terminal block spare kits

If you have S7-1200 CPU V4.0 and later (article number)	Use this terminal block spare kit (4/pk)	
	Terminal block article number	Terminal block description
CPU 1211C DC/DC/DC (6ES7 211-1AE40-0XB0)	6ES7 292-1BC30-0XA0	3 pin, gold-plated
	6ES7 292-1AH30-0XA0	8 pin, tin-plated
	6ES7 292-1AP30-0XA0	14 pin, tin-plated
CPU 1211C DC/DC/Relay (6ES7 211-1HE40-0XB0)	6ES7 292-1AH30-0XA0	3 pin, gold-plated
	6ES7 292-1AH40-0XA0	8 pin, tin-plated, keyed
	6ES7 292-1AP30-0XA0	14 pin, tin-plated
CPU 1211C AC/DC/Relay (6ES7 211-1BE40-0XB0)	6ES7 292-1BC30-0XA0	3 pin, gold-plated
	6ES7 292-1AH40-0XA0	8 pin, tin-plated, keyed
	6ES7 292-1AP40-0XA0	14 pin, tin-plated, keyed
CPU 1212C DC/DC/DC (6ES7 212-1AE40-0XB0)	6ES7 292-1BC30-0XA0	3 pin, gold-plated
	6ES7 292-1AH30-0XA0	8 pin, tin-plated
	6ES7 292-1AP30-0XA0	14 pin, tin-plated
CPU 1212C DC/DC/Relay (6ES7 212-1HE40-0XB0)	6ES7 292-1AH30-0XA0	3 pin, gold-plated
	6ES7292-1AH40-0XA0	8 pin, tin-plated, keyed
	6ES7 292-1AP30-0XA0	14 pin, tin-plated
CPU 1212C AC/DC/Relay (6ES7 212-1BE40-0XB0)	6ES7 292-1BC30-0XA0	3 pin, gold-plated
	6ES7 292-1AH40-0XA0	8 pin, tin-plated, keyed
	6ES7 292-1AP40-0XA0	14 pin, tin-plated, keyed
CPU 1214C DC/DC/DC (6ES7 214-1AG40-0XB0)	6ES7 292-1BC30-0XA0	3 pin, gold-plated
	6ES7 292-1AM30-0XA0	12 pin, tin-plated
	6ES7 292-1AV30-0XA0	20 pin, tin-plated
CPU 1214C DC/DC/Relay (6ES7 214-1HG40-0XB0)	6ES7 292-1BC30-0XA0	3 pin, gold-plated
	6ES7 292-1AM40-0XA0	12 pin, tin-plated, keyed
	6ES7 292-1AV30-0XA0	20 pin, tin-plated
CPU 1214C AC/DC/Relay (6ES7 214-1BG40-0XB0)	6ES7 292-1BC30-0XA0	3 pin, gold-plated
	6ES7 292-1AM40-0XA0	12 pin, tin-plated, keyed
	6ES7 292-1AV40-0XA0	20 pin, tin-plated, keyed
CPU 1215C DC/DC/DC (6ES7 215-1AG40-0XB0)	6ES7 292-1BF30-0XB0	6 pin, gold-plated
CPU 1215C DC/DC/Relay (6ES7 215-1HG40-0XB0)	6ES7 292-1BF30-0XB0	6 pin, gold-plated
	6ES7 292-1AM40-0XA0	12 pin, tin-plated, keyed
	6ES7 292-1AV30-0XA0	20 pin, tin-plated

If you have S7-1200 CPU V4.0 and later (article number)	Use this terminal block spare kit (4/pk) Terminal block article number	Terminal block description
CPU 1215C AC/DC/Relay (6ES7 215-1BG40-0XB0)	6ES7 292-1BF30-0XB0	6 pin, gold-plated
	6ES7 292-1AM40-0XA0	12 pin, tin-plated, keyed
	6ES7 292-1AV40-0XA0	20 pin, tin-plated, keyed
CPU 1217C DC/DC/DC (6ES7 217-1AG40-0XB0)	6ES7 292-1BF30-0XB0	6 pin, gold-plated
	6ES7 292-1AK30-0XA0	10 pin, pin-plated
	6ES7 292-1AR30-0XA0	16 pin, pin-plated
	6ES7 292-1AT30-0XA0	18 pin, tin-plated

Table C- 17 S7-1200 SMs V4.0 and later - Terminal block spare kits

If you have S7-1200 SM V4.0 and later (article number)	Use this terminal block spare kit (4/pk) Terminal block article number	Terminal block description
SM1221 DI 8xDC (6ES7 221-1BF32-0XB0)	6ES7 292-1AG30-0XA0	7 pin, tin-plated
SM1222 DQ 8xDC (6ES7 222-1BF32-0XB0)	6ES7 292-1AG30-0XA0	7 pin, tin-plated
SM1222 DQ 8xRelay (6ES7 222-1HF32-0XB0)	6ES7 292-1AG40-0XA1	7 pin, tin-plated, keyed-left
SM1231 AI 4x13 bit (6ES7 231-4HD32-0XB0)	6ES7 292-1BG30-0XA0	7 pin, gold-plated
SM1232 AQ 2x14 bit (6ES7 232-4HB32-0XB0)	6ES7 292-1BG30-0XA0	7 pin, gold-plated
SM1231 AI4 x TC (6ES7 231-5QD32-0XB0)	6ES7 292-1BG30-0XA0	7 pin, gold-plated
SM1231 AI4 x 16 bit (6ES7 231-5ND32-0XB0)	6ES7 292-1BG30-0XA0	7 pin, gold-plated
SM1221 DI 16xDC (6ES7 221-1BH32-0XB0)	6ES7 292-1AG30-0XA0	7 pin, tin-plated
SM1222 DQ 16xDC (6ES7 222-1BH32-0XB0)	6ES7 292-1AG30-0XA0	7 pin, tin-plated
SM1222 DQ 16xRelay (6ES7 222-1HH32-0XB0)	6ES7 292-1AG40-0XA0	7 pin, tin-plated, keyed-right
SM1223 DI 8xDC/DQ 8xDC (6ES7 223-1BH32-0XB0)	6ES7 292-1AG30-0XA0	7 pin, tin-plated
SM1223 8xDC/8xRelay (6ES7 223-1PH32-0XB0)	6ES7 292-1AG30-0XA0	7 pin, tin-plated
	6ES7 292-1AG40-0XA0	7 pin, tin-plated, keyed-right
SM1223 8xAC/8xRelay (6ES7 223-1QH32-0XB0)	6ES7 292-1AG40-0XA0	7 pin, tin-plated, keyed-right
SM1234 AI 4 / AQ 2 (6ES7 234-4HE32-0XB0)	6ES7 292-1BG30-0XA0	7 pin, gold-plated
SM1231 AI 8x13 BIT (6ES7 231-4HF32-0XB0)	6ES7 292-1BG30-0XA0	7 pin, gold-plated
SM1232 AQ4x14 bit (6ES7 232-4HD32-0XB0)	6ES7 292-1BG30-0XA0	7 pin, gold-plated
SM1231 AI4 x RTD (6ES7 231-5PD32-0XB0)	6ES7 292-1BG30-0XA0	7 pin, gold-plated
SM1231 AI8 x TC (6ES7 231-5QF32-0XB0)	6ES7 292-1BG30-0XA0	7 pin, gold-plated
SM 1278 IO LINK (6ES7 278-4BD32 0XB0)	6ES7 292-1AG30-0XA0	7 pin, tin-plated
SM1222 DQ 8xRelay (Changeover) (6ES7 222-1XF32-0XB0)	6ES7 292-1AL30-0XA0	11 pin, tin-plated
SM1223 DI 16xDC/DQ 16xDC (6ES7 223-1BL32-0XB0)	6ES7 292-1AL30-0XA0	11 pin, tin-plated
	6ES7 292-1AL40-0XA0	11 pin, tin-plated, keyed
SM1231 AI8 x RTD (6ES7 231-5PF32-0XB0)	6ES7 231-5PF32-0XB0	11 pin, gold-plated

Table C- 18 Fail-Safe CPUs - Terminal block spare kit

If you have Fail-Safe CPU (article number)	Use this terminal block spare kit (4/pk)	
	Terminal block article number	Terminal block description
CPU 1214FC DC/DC/DC (6ES7 214-1AF40-0XB0)	6ES7 292-1BC30-0XA0	3 pin, gold-plated
	6ES7 292-1AM30-0XA0	12 pin, tin-plated
	6ES7 292-1AV30-0XA0	20 pin, tin-plated
CPU 1214FC DC/DC/Relay (6ES7 214-1HF40-0XB0)	6ES7 292-1BC30-0XA0	3 pin, gold-plated
	6ES7 292-1AM40-0XA0	12 pin, tin-plated, keyed
	6ES7 292-1AV30-0XA0	20 pin, tin-plated
CPU 1215FC DC/DC/DC (6ES7 215 1AF40 0XB0)	6ES7 292-1BF30-0XB0	6 pin, gold-plate
	6ES7 292-1AM30-0XA0	12 pin, tin-plated
	6ES7 292-1AV30-0XA0	20 pin, tin-plated
CPU 1215FC DC/DC/Relay (6ES7 215 1HF40 0XB0)	6ES7 292-1BF30-0XB01	6 pin, gold-plated
	6ES7 292-1AM40-0XA0	2 pin, tin-plated, keyed
	6ES7 292-1AV30-0XA0	20 pin, tin-plated

Table C- 19 Fail-Safe signal modules - Terminal block spare kit

If you have Fail-Safe signal module (article number)	Use this Terminal block spare kit (4/pk)	
	Terminal block article number	Terminal block description
SM 1226 F-DI (6ES7 226-6BA32-0XB0)	6ES7 292-1AL30-0XA0	11 pin, tin-plated
SM 1226 F-DQ (6ES7 226-6DA32-0XB0)	6ES7 292-1AL30-0XA0	11 pin, tin-plated
SM 1226 F-Relay (6ES7 226-6RA32-0XB0)	6ES7 292-1AL40-0XA0	11 pin, tin-plated, keyed

C.9 Programming software

Table C- 20 Programming software

SIMATIC software		Article number
Programming software	STEP 7 Basic V13	6ES7 822-0AA01-0YA0
	STEP 7 Professional V13	6ES7 822-1AA01-0YA5
Visualization soft-ware	WinCC Basic V12 SP1	6AV2100-0AA01-0AA0
	WinCC Comfort V12 SP1	6AV2101-0AA01-0AA5
	WinCC Advanced V12 SP1	6AV2102-0AA01-0AA5
	WinCC Professional 512 PowerTags V12 SP1	6AV2103-0DA01-0AA5
	WinCC Professional 4096 PowerTags V12 SP1	6AV2103-0HA01-0AA5
	WinCC Professional max. PowerTags V12 SP1	6AV2103-0XA01-0AA5

Device exchange and spare parts compatibility D

You can replace your V3.0 CPU with a V4.1 CPU (Page 161) and use your existing STEP 7 project that you designed for the V3.0 CPU. You cannot upgrade a V3.0 CPU to a V4.1 CPU by firmware update; you must replace the hardware. When you replace a V3.0 CPU with a V4.1 CPU, you might also want to check for and apply firmware updates (Page 141) to your connected signal and communication modules.

Note

No device exchange possible from V4.1 to V3.0

You can exchange a V3.0 CPU for a V4.1 CPU, but you cannot exchange a V4.1 CPU for a V3.0 CPU after you download the configuration. If you want to view or otherwise use your existing STEP 7 V3.0 project, make an archive of your STEP 7 V3.0 project prior to the device exchange.

Note that if you have not downloaded the exchanged device configuration, you can undo it. After downloading, however, you cannot undo the exchange from V3.0 to V4.1.

You need to be aware of some configuration and operational changes between the two CPU versions:

Organization blocks

With V4.1, you can configure OB execution to be interruptible or non-interruptible (Page 98). For projects from former V3.0 CPUs, STEP 7 sets all OBs by default to be non-interruptible.

STEP 7 sets all OB priorities (Page 98) to the values they were in the V3.0 CPU STEP 7 project.

You can subsequently change the interruptability or priority settings if you choose.

The Diagnostic error interrupt OB (Page 92) start information references the submodule as a whole if no diagnostics event is pending.

CPU password protection

STEP 7 sets the password protection level (Page 197) for the V4.1 CPU to be the equivalent password protection level that was set for the V3.0 CPU, and assigns the V3.0 password to the "Full access (no protection)" password for the V4.1 CPU:

V3.0 protection level	V4.1 access level
No protection	Full access (no protection)
Write protection	Read access
Write/read protection	HMI access

Note that the V4.1 access level "No access (complete protection)" did not exist for V3.0.

Web server

If you use user-defined Web pages in your V3.0 project, store them in your project installation folder under the subfolder "UserFiles\Webserver" prior to upgrading your project. If you store your user-defined pages at this location, saving the STEP 7 project will also save the user-defined Web pages.

If you exchange a V3.0 CPU for a V4.1 CPU, your Web server project setting (Page 787) for activating the Web server and HTTPS setting will be the same as it was in V3.0. You can then configure users, privileges, passwords (Page 789), and languages (Page 787) as needed to use the Web server. If you do not configure users with additional privileges, then you are limited as to what you can view from the standard Web pages (Page 794). The S7-1200 V4.1 CPU does not support the former pre-configured "admin" user and password.

The S7-1200 V3.0 Web server Data log page provided a "Download and Clear" operation. The V4.1 Web server File browser page (Page 810), from which you access data logs, no longer provides this feature. Instead, the Web server provides the ability to download, rename, and delete data log files.

Transfer card incompatibility

You cannot use a V3.0 transfer card (Page 133) to transfer a V3.0 program to a V4.1 CPU. You must open the V3.0 project in STEP 7, change the device to a V4.1 CPU (Page 161), and download the STEP 7 project to your V4.1 CPU. After you have changed your project to a V4.1 project, you can then make a V4.1 transfer card for subsequent program transfers.

GET/PUT communication

By default, GET/PUT communication was enabled in V3.0. When you replace your V3.0 CPU with a V4.1 CPU (Page 161), you see a message in the compatibility information section stating that GET/PUT is enabled.

Motion control support

S7-1200 V4.1 CPUs do not support the V1.0 and V2.0 motion libraries. If you perform a device exchange for a STEP 7 project with V1.0 or V2.0 motion libraries, the device exchange substitutes compatible V3.0 motion control instructions (Page 571) for the V1.0 or V2.0 motion library instructions at compile.

If you perform a device exchange from a V3.0 CPU to a V4.1 CPU for a STEP 7 project that contains two different motion control instruction versions (V3.0 and V5.0), the device exchange substitutes compatible V5.0 motion control instructions (Page 571) at compile.

During a device exchange from a V3.0 CPU to a V4.1 CPU, the motion control Technological Object (TO) version does not automatically change from V3.0 to V5.0. If you want to upgrade to the later versions, you must go to the Instructions tree and select the required S7-1200 Motion Control version for your project as shown in the table below:

CPU version	Allowed motion control versions
V4.1 (motion control V5.0)	V5.0 or V4.0 or V3.0
V4.0 (motion control V4.0)	V4.0 or V3.0
V3.0 (motion control V3.0)	V3.0

The TO structure is different between motion control versions V3.0 and V5.0. All associated blocks change as well. Block interfaces, watch tables, and traces update to the new motion control V5.0 structure. You can find the differences between the V3.0 CPU and V4.1 CPU motion control axis parameters in the following two tables:

V3.0 CPU (Motion control V3.0)	V4.1 CPU (Motion control V5.0)
Config.General.LengthUnit	Units.LengthUnit
Config.Mechanics.PulsesPerDriveRevolution	Actor.DriveParameter.PulsesPerDriveRevolution
Config.Mechanics.LeadScrew	Mechanics.LeadScrew
Config.Mechanics.InverseDirection	Actor.InverseDirection
Config.DynamicLimits.MinVelocity	DynamicLimits.MinVelocity
Config.DynamicLimits.MaxVelocity	DynamicLimits.MaxVelocity
Config.DynamicDefaults.Acceleration	DynamicDefaults.Acceleration
Config.DynamicDefaults.Deceleration	DynamicDefaults.Deceleration
Config.DynamicDefaults.EmergencyDeceleration	DynamicDefaults.EmergencyDeceleration
Config.DynamicDefaults.Jerk	DynamicDefaults.Jerk
Config.PositionLimits_SW.Active	PositionLimitsSW.Active
Config.PositionLimits_SW.MinPosition	PositionLimitsSW.MinPosition
Config.PositionLimits_SW.MaxPosition	PositionLimitsSW.MaxPosition
Config.PositionLimits_HW.Active	PositionLimitsHW.Active
Config.PositionLimits_HW.MinSwitchedLevel	PositionLimitsHW.MinSwitchLevel
Config.PositionLimits_HW.MaxSwitchedLevel	PositionLimitsHW.MaxSwitchLevel
Config.Homing.AutoReversal	Homing.AutoReversal
Config.Homing.Direction	Homing.ApproachDirection
Config.Homing.SideActiveHoming	Sensor[1].ActiveHoming.SideInput
Config.Homing.SidePassiveHoming	Sensor[1].PassiveHoming.SideInput
Config.Homing.Offset	Sensor[1].ActiveHoming.HomePositionOffset
Config.Homing.FastVelocity	Homing.ApproachVelocity
Config.Homing.SlowVelocity	Homing.ReferencingVelocity
MotionStatus.Position	Position
MotionStatus.Velocity	Velocity
MotionStatus.Distance	StatusPositioning.Distance
MotionStatus.TargetPosition	StatusPositioning.TargetPosition
StatusBits.SpeedCommand	StatusBits.VelocityCommand
StatusBits.Homing	StatusBits.HomingCommand

The only "commandtable" parameter that is renamed is the array with the commands:

V3.0	V4.1
Config.Command[]	Command[]

Note: The array "Command[]" is a UDT of the type "TO_CmdTab_Config_Command" in V3.0 and "TO_Struct_Command" in V4.1.

Instruction changes

The following instructions have changes in parameters or behavior:

- RDREC and WRREC (Page 351)
- CONV (Page 269)

HMI panel communication

If you had one or more HMI panels (Page 30) connected to your S7-1200 V3.0 CPU, the communication to the S7-1200 V4.1 CPU depends on the type of communication you use and the firmware version of the HMI panel. Recompile and download your project to the CPU and the HMI and/or update your HMI firmware.

Requirement to recompile program blocks

After exchanging a V3.0 CPU for a V4.1 CPU, you must recompile all program blocks before you can download them to the V4.1 CPU. Additionally, if any of the blocks have know-how protection (Page 200) or copy protection bound to a PLC serial number (Page 201), you must remove the protection before you compile and download the blocks. (You do not, however, need to deactivate copy protection bound to a memory card.) After a successful compile, you can reconfigure the know-how protection and/or PLC serial number copy protection. Note that if your project includes any blocks with know-how protection that an OEM (Original Equipment Manufacturer) provided, you must contact the OEM to provide V4.1 versions of those blocks.

In general, Siemens recommends that you recompile the hardware configuration and software in STEP 7 and download to all devices in your project after the device exchange. Correct any errors that compiling the project finds, and recompile until you have no errors. Then, you can download the project to the V4.1 CPU.

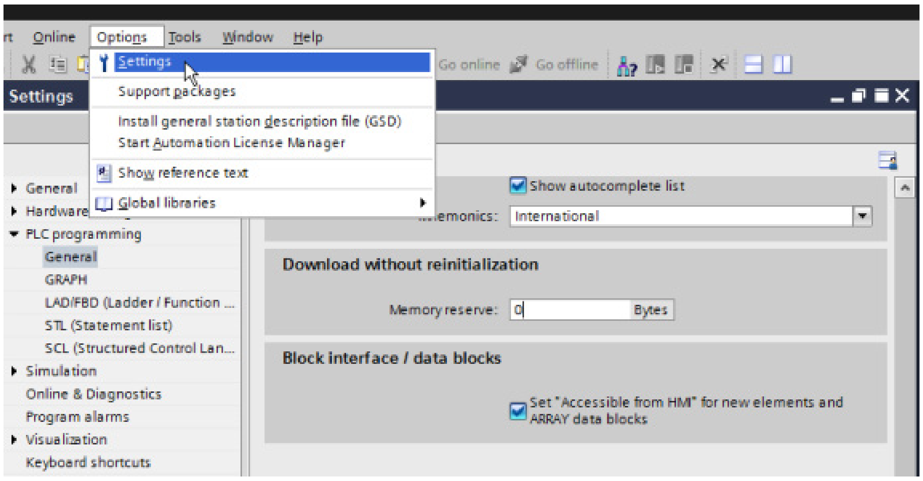
S7-1200 V3.0 projects might not fit in S7-1200 V4.1 CPUs

S7-1200 V4.0 added a reserve area of 100 bytes to each DB to support download without reinitialization.

You can remove the 100-byte reserve area from DBs prior to attempting to download a V3.0 project to a V4.1CPU.

To remove the 100-byte reserve area, follow these steps before you perform the device exchange:

1. From the TIA Portal main menu, select the Options > Settings menu command.
2. From the navigation tree, open the PLC programming > General node.
3. In the "Download without reinitialization" area, set the memory reserve to 0 bytes.



Show autocomplete list ✓

...emonics: International

Download without reinitialization

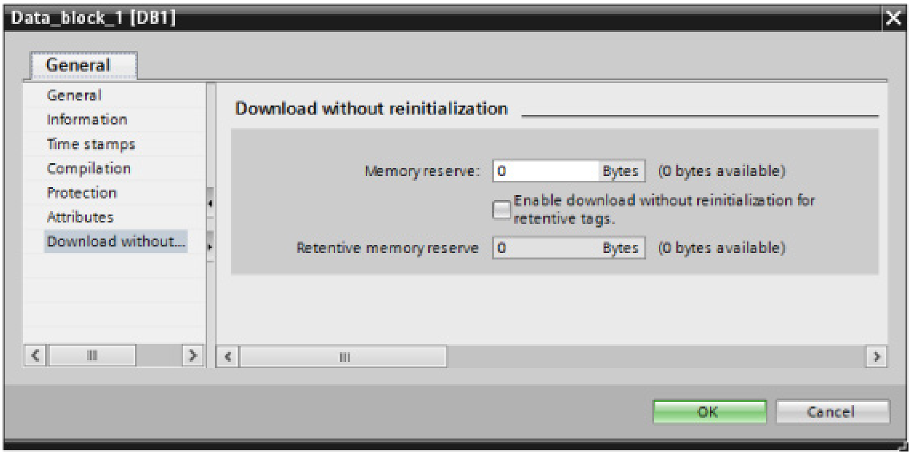
Memory reserve: 0 Bytes

Block interface / data blocks

✓ Set "Accessible from HMI" for new elements and ARRAY data blocks

If you have already performed the device exchange, you must remove the 100-byte reserve from each block individually:

1. From the project tree, right-click a data block from the Program blocks folder and select Properties from the shortcut menu.
2. In the Data block properties dialog, select the "Download without reinitialization" node.
3. Set the memory reserve to 0 bytes.
4. Repeat for each data block in your project.



Download without reinitialization

Memory reserve: 0 Bytes (0 bytes available)

☐ Enable download without reinitialization for retentive tags.

Retentive memory reserve 0 Bytes (0 bytes available)

D.1 S7-1200 V3.0 and V4.0 terminal block spare kits

Table D- 1 S7-1200 CPU V3.0 and earlier - Terminal Block spare kits

If you have	Use this terminal block spare kit (4/pk)	
S7-1200 CPU V3.0 and earlier (article number)	**Terminal block article number**	**Terminal block description**
CPU 1211C DC/DC/DC (6ES7 211-1AE31-0XB0)	6ES7 292-1BC30-0XA0	3 pin, gold-plated
CPU 1211C DC/DC/Relay (6ES7 211-1HE31-0XB0)	6ES7 292-1AH30-0XA0	8 pin, gold-plated
CPU 1211C AC/DC/Relay (6ES7 211-1BE31-0XB0)	6ES7 292-1AP30-0XA0	14 pin, tin-plated, keyed
CPU 1212C DC/DC/DC (6ES7 212-1AE31-0XB0)		
CPU 1212C DC/DC/Relay (6ES7 212-1HE31-0XB0)		
CPU 1212C AC/DC/Relay (6ES7 212-1BE31-0XB0)		
CPU 1214C DC/DC/DC (6ES7 214-1AG31-0XB0)	6ES7 292-1BC3-0XA0	3 pin, gold-plated
CPU 1214C DC/DC/Relay (6ES7 214-1HG31-0XB0)	6ES7 292-1AM30-0XA0	12 pin, tin-plated
CPU 1214C AC/DC/Relay (6ES7 214-1BG31-0XB0)	6ES7 292-1AV30-0XA0	20 pin, tin-plated
CPU 1215C DC/DC/DC (6ES7 215-1AG31-0XB0)	6ES7 292-1BF30-0XB0	6 pin, gold-plated
CPU 1215C DC/DC/Relay (6ES7 215-1HG31-0XB0)	6ES7 292-1AM30-0XA0	12 pin, tin-plated
CPU 1215C AC/DC/Relay (6ES7 215-1BG31-0XB0)	6ES7 292-1AV30-0XA0	20 pin, tin-plated, keyed

Table D- 2 S7-1200 SMs V3.0 and earlier - Terminal block spare kits

If you have	Use this terminal block spare kit (4/pk)	
S7-1200 SM V3.0 and earlier (article number)	**Terminal block article number**	**Terminal block description**
SM1221 DI 8xDC (6ES7 221-1BF32-0XB0)	6ES7 292-1AG30-0XA0	7 pin, tin-plated
SM1222 DQ 8xDC (6ES7 222-1BF32-0XB0)		
SM1222 DQ 8xRelay (6ES7 222-1HF32-0XB0)	6Es7 292-1AG40-0XA1	7 pin, tin-plated, keyed-left
SM1231 AI 4x13 bit (6ES7 231-4HD32-0XB0)	6ES7 292-1BG30-0XA0	7 pin, gold-plated
SM1232 AQ 2x14 bit (6ES7 232-4HB32-0XB0)		
SM1231 AI4 x TC (6ES7 231-5QD32-0XB0)		
SM1231 AI4 x 16 bit (6ES7 231-5ND32-0XB0)		
SM1221 DI 16xDC (6ES7 221-1BH32-0XB0)	6ES7 292-1AG30-0XA0	7 pin, tin-plated
SM1222 DQ 16xDC (6ES7 222-1BH32-0XB0)	6ES7 292-1AG30-0XA0	7 pin, tin-plated
SM1222 DQ 16xRelay (6ES7 222-1HH32-0XB0)	6ES7 292-1AG40-0XA0	7 pin, tin-plated, keyed-right
SM1223 DI 8xDC/DQ 8xDC (6ES7 223-1BH32-0XB0)	6ES7 292-1AG30-0XA0	7 pin, tin-plated
SM1223 8xDC/8xRelay (6ES7 223-1PH32-0XB0)	6ES7 292-1AG30-0XA0	7 pin, tin-plated
	6ES7 292-1AG40-0XA0	7 pin, tin-plated, keyed-right
SM1223 8xAC/8xRelay (6ES7 223-1QH32-0XB0)	6Es7 292-1AG40-0XA0	7 in, tin-plated, keyed right

If you have	Use this terminal block spare kit (4/pk)	
S7-1200 SM V3.0 and earlier (article number)	Terminal block article number	Terminal block description
SM1234 AI 4 / AQ 2 (6ES7 234-4HE32-0XB0)	6ES7 292-1BG30-0XA0	7 pin, gold-plated
SM1231 AI 8x13 BIT (6ES7 231-4HF32-0XB0)		
SM1232 AQ4x14 bit (6ES7 232-4HD32-0XB0)		
SM1231 AI4 x RTD (6ES7 231-5PD32-0XB0)		
SM1231 AI8 x TC (6ES7 231-5QF32-0XB0)		
SM 1278 IO Link(6ES7 278-4BD32-0XB0)	6ES7 292-1AG30-0XA0	7 pin, tin-plated

Table D- 3 S7-1200 SMs V3.2 and later - Terminal Block spare kits

If you have	Use this terminal block spare kit (4/pk)	
S7-1200 SM V3.2 and later (article number)	Terminal block article number	Terminal block description
SM1221 DI 8xDC (6ES7 221-1BF30-0XB0)	6ES7 292-1AG30-0XA0	7 pin, tin-plated
SM1222 DQ 8xDC (6ES7 222-1BF30-0XB0)		
SM1222 DQ 8xRelay (6ES7 222-1HF30-0XB0)		
SM1231 AI 4x13 bit (6ES7 231-4HD30-0XB0)	6ES7 292-1BG30-0XA0	7 pin, gold-plated
SM1232 AQ 2x14 bit (6ES7 232-4HB30-0XB0)		
SM1231 AI4 x TC (6ES7 231-5QD30-0XB0)		
SM1231 AI4 x 16 bit (6ES7 231-5ND30-0XB0)		
SM1221 DI 16xDC (6ES7 221-1BH30-0XB0)	6ES7 292-1AG30-0XA0	7 pin, tin-plated
SM1222 DQ 16xDC (6ES7 222-1BH30-0XB0)		
SM1222 DQ 16xRelay (6ES7 222-1HH30-0XB0)		
SM1223 DI 8xDC/DQ 8xDC (6ES7 223-1BH30-0XB0)		
SM1223 8xDC/8xRelay (6ES7 223-1PH30-0XB0)		
SM1223 8xAC/8xRelay (6ES7 223-1QH30-0XB0)		
SM1234 AI 4 / AQ 2 (6ES7 234-4HE30-0XB0)	6ES7 292-1BG30-0XA0	7 pin, gold-plated
SM1231 AI 8x13 BIT (6ES7 231-4HF30-0XB0)		
SM1232 AQ4x14 bit (6ES7 232-4HD30-0XB0)		
SM1231 AI4 x RTD (6ES7 231-5PD30-0XB0)		
SM1231 AI8 x TC (6ES7 231-5QF30-0XB0)		
SM1222 DQ 8xRelay (Changeover) (6ES7 222-1XF30-0XB0)	6ES7 292-1AL30-0XA0	11 pin, tin-plated
SM1223 DI 16xDC/DQ 16xDC (6ES7 223-1BL30-0XB0)		
SM 1223 16 xDC/16X Relay (6ES7 223-1PL30-1XB0)		
SM1231 AI8 x RTD (6ES7 231-5PF30-0XB0)	6ES7 292-1BL30-0XA0	11 pin, gold-plated

Index