

SIEMENS

SIMATIC

Industrial Software SIMATIC Safety - Configuring and Programming

Programming and Operating Manual

Important notes

<u>Product Overview</u>	1
<u>Configuration</u>	2
<u>Safety Administration Editor</u>	3
<u>Access protection</u>	4
<u>Programming</u>	5
<u>F-I/O access</u>	6
<u>Implementation of user acknowledgment</u>	7
<u>Data exchange between standard user program and safety program</u>	8
<u>Configuring and Programming Communication</u>	9
<u>Compiling and commissioning a safety program</u>	10
<u>System Acceptance Test</u>	11
<u>Operation and Maintenance</u>	12
<u>STEP 7 Safety Advanced V11 Instructions</u>	13
<u>Monitoring and response times</u>	A
<u>Checklist</u>	B

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
with a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.
CAUTION
without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.
NOTICE
indicates that an unintended result or situation can occur if the relevant information is not taken into account.

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.

Important notes

Purpose of this documentation

The information in this documentation enables you to configure and program SIMATIC Safety fail-safe systems. In addition, you will obtain information on acceptance testing of an SIMATIC Safety F-system.

Basic knowledge requirements

General basic knowledge of automation engineering is needed to understand this documentation. Basic knowledge of the following is also necessary:

- Fail-safe automation systems
- S7-300/S7-400 automation systems
- Distributed I/O systems on PROFIBUS DP/PROFINET IO
- Totally Integrated Automation Portal, including:
 - Hardware configuration with the *hardware and network editor*
 - Programming in the LAD and FBD programming languages using the *program editor*.
 - Communication between CPUs

Scope of this documentation

This documentation is valid for the *STEP 7 Safety Advanced V11* optional package.

The *STEP 7 Safety Advanced V11* optional package is used for configuring and programming of the SIMATIC Safety fail-safe system. In this context, integration of the fail-safe I/O listed below in SIMATIC Safety is also addressed:

- ET 200S fail-safe modules
- ET 200eco fail-safe I/O modules
- ET 200pro fail-safe modules
- ET 200iSP fail-safe modules
- S7-300 fail-safe signal modules
- Fail-safe DP standard slaves
- Fail-safe standard I/O devices

Approvals

The SIMATIC Safety F-system is certified for use in safety mode up to:

- Safety Integrity Level SIL3 in accordance with IEC 61508:2010
- Category 4 in accordance with EN 954-1
- Performance Level (PL) e according to ISO 13849-1: 2006 or in accordance with EN ISO 13849-1: 2008

Incorporation in the information landscape

Depending on your application, you will need the following supplementary documentation when working with *STEP 7 Safety Advanced V11*.

This documentation includes references to the supplementary documentation where appropriate.

Documentation	Brief description of relevant content
For the SIMATIC Safety system	<p>Depending on which F-CPU you are using, you will need the following documentation:</p> <ul style="list-style-type: none"> • The Operating Instructions "S7-300, CPU 31xC and CPU 31x: Installation" (http://support.automation.siemens.com/WW/view/en/13008499) describes the installation and wiring of S7-300 systems. • The Device Manual "CPU 31xC and CPU 31x, Technical Data" (http://support.automation.siemens.com/WW/view/en/12996906) describes the CPUs 315-2 DP and PN/DP, the CPU 317-2 DP and PN/DP, and the CPU 319-3 PN/DP. • The Installation Manual "S7-400 Automation System, Installation" (http://support.automation.siemens.com/WW/view/en/1117849) describes the installation and wiring of S7-400 systems. • The Reference Manual "S7-400 Automation System, CPU Data" (http://support.automation.siemens.com/WW/view/en/23904550) describes the CPUs 414-3 PN/DP, the CPU 416-2, and the CPU 416-3 PN/DP. • The Manual "ET 200S Interface Module IM 151-7 CPU" (http://support.automation.siemens.com/WW/view/en/12714722) describes the IM 151-7 CPU. • The Manual "ET 200S, Interface Module IM 151-8 PN/DP CPU" (http://support.automation.siemens.com/WW/view/en/47409312) describes the IM 151-8 PN/DP CPU. • The Manual "ET 200S, Interface Module IM 154-8 CPU" (http://support.automation.siemens.com/WW/view/de/24363739/0/en) describes the IM 154-8 CPU. • The Manual "Windows Automation Center RTX WinAC RTX (F) 2010" (http://support.automation.siemens.com/WW/view/en/43715176) describes the WinAC RTX 2010 and the WinAC RTX F 2010. • Each F-CPU that can be used has its own product information. The product information only describes the deviations from the respective standard CPUs.

Documentation	Brief description of relevant content
Manual "ET 200eco Distributed I/O Station Fail-safe I/O Block (http://support.automation.siemens.com/WW/view/en/19033850)"	describes the hardware of the ET 200eco fail-safe I/O module (including installation, wiring, and technical specifications)
Operating Instructions "Distributed I/O System ET 200S, Fail-Safe Modules (http://support.automation.siemens.com/WW/view/en/27235629)"	describes the hardware of the ET 200S fail-safe modules (including installation, wiring, and technical specifications)
Manual "S7-300 Automation System, ET 200M Distributed I/O System, Fail-safe Signal Modules (http://support.automation.siemens.com/WW/view/en/19026151)"	describes the hardware of the S7-300 fail-safe signal modules (including installation, wiring, and technical specifications)
Operating Instructions "ET 200pro Distributed I/O Station, Fail-safe I/O Module (http://support.automation.siemens.com/WW/view/en/22098524)"	describes the hardware of the ET 200pro fail-safe modules (including installation, wiring, and technical specifications)
Operating Instructions "ET 200iSP distributed I/O device - Fail-safe modules (http://support.automation.siemens.com/WW/view/en/47357221)"	describes the hardware of the ET 200iSP fail-safe modules (including installation, wiring, and technical specifications)
Online help on <i>STEP 7 Professional</i>	<ul style="list-style-type: none"> describes how to operate the standard tools of <i>STEP 7 Professional</i>. contains information regarding configuration and parameter assignment of hardware contains a description of the FBD and LAD programming languages

The complete *SIMATIC S7* documentation is available on DVD. You will find more information on the Internet (<http://www.automation.siemens.com/mcms/industrial-automation-systems-simatic/en/manual-overview/manual-collection/Pages/Default.aspx>).

Guide

This documentation describes how to work with the *STEP 7 Safety Advanced V11* optional package. It includes instructions and reference sections (description of the instructions for the safety program).

The following topics are addressed:

- Configuration of SIMATIC Safety
- Access protection for SIMATIC Safety
- Programming of the safety program (safety-related user program)
- Safety-related communication
- Instructions for the safety program
- Support for the system acceptance test
- Operation and maintenance of SIMATIC Safety
- Monitoring and response times

Conventions

In this documentation, the terms "safety engineering" and "fail-safe engineering" are used synonymously. The same applies to the terms "fail-safe" and "F-".

When "*STEP 7 Safety Advanced V11*" appears in italics, it refers to the optional package for the "SIMATIC Safety" F-system.

The term "safety program" refers to the fail-safe portion of the user program and is used instead of "fail-safe user program," "F-program," etc. For purposes of contrast, the non-safety-related part of the user program is referred to as the "standard user program".

All fail-safe modules and instructions are highlighted in yellow to distinguish them from the modules and instructions of the standard user programs on the software interface (e.g., in the project tree). Similarly, the fail-safe parameters of F-CPU's and F-I/O are highlighted in yellow in the hardware configuration.

Each warning is marked with a unique number at the end of the text. This enables you to easily reference other documents to obtain an overview of the safety requirements for the system.

Additional support

If you have further questions about the use of products presented in this manual, contact your local Siemens representative.

You will find information on whom to contact on the Web (<http://www.siemens.com/automation/partner>).

A guide to the technical documentation for the various SIMATIC products and systems is available on the Web (<http://www.siemens.com/simatic-tech-doku-portal>).

You will find the online catalog and online ordering system on the Web (www.siemens.com/industrymall).

Training center

We offer courses to help you get started with the S7 automation system. Contact your regional training center or the central training center in Nuremberg (90327), Federal Republic of Germany.

You will find more information on the Internet (<http://www.sitrain.com>).

H/F Competence Center

The H/F Competence Center in Nuremberg offers special workshops on *SIMATIC S7* fail-safe automation systems with high degree of availability. The H/F Competence Center can also provide assistance with onsite configuration, commissioning, and troubleshooting.

For questions about workshops, etc., contact: hf-cc.aud@siemens.com

Technical Support

To contact Technical Support for all Industry Automation products, use the Support Request Web form (<http://www.siemens.com/automation/support-request>).

You can find additional information about our Technical Support on the Web (<http://www.siemens.com/automation/service>).

Service & Support on the Web

In addition to our documentation, we also offer a comprehensive technical knowledge base on the Internet (<http://www.siemens.com/automation/service&support>).

There, you will find the following information:

- Newsletters providing the latest information on your products
- A search engine in Service & Support for locating the documents you need
- A forum where users and experts from all over the world exchange ideas
- Your local contact.
- Information about on-site services, repairs, and about spare parts. Lots more can be found on our "Services" page.

Important note for maintaining the operational safety of your system

Note

The operators of systems with safety-related characteristics must adhere to operational safety requirements. The supplier is also obliged to comply with special product monitoring measures. Siemens publishes a special newsletter to keep plant operators informed about product developments and properties which may form important issues in terms of operational safety. You should subscribe to the corresponding newsletter in order to obtain the latest information and to allow you to modify your plant accordingly. Please go to the Internet

(<https://www.automation.siemens.com/WW/newsletter/guiThemes2Select.aspx?HTTPS=REDIR&subjectID=2>) and register for the following newsletters:

- SIMATIC S7-300/S7-300F
- SIMATIC S7-400/S7-400H/S7-400F/FH
- Distributed I/O
- SIMATIC Industrial Software

To receive these newsletters, select the check box "Update".

Table of contents

	Important notes	3
1	Product Overview	17
1.1	Overview	17
1.2	Hardware and Software Components.....	18
1.3	Installation/uninstallation of the STEP 7 Safety Advanced V11 optional package	21
1.4	Migration of projects from S7 Distributed Safety V5.4 SP5 to STEP 7 Safety Advanced V11	22
1.5	First steps.....	24
2	Configuration	25
2.1	Overview of Configuration.....	25
2.2	Particularities for configuring the F-System	28
2.3	Configuring the F-CPU.....	29
2.4	Configuring the F-I/O	33
2.5	Configuring fail-safe DP standard slaves and fail-safe standard I/O devices.....	38
3	Safety Administration Editor	39
3.1	"General" tab.....	41
3.2	"F-blocks" tab	44
3.3	"Settings" tab.....	45
4	Access protection	49
4.1	Overview of Access Protection	49
4.2	Setting up, changing and revoking access permission for the safety program	51
4.3	Setting up access permission for the F-CPU.....	54
5	Programming	55
5.1	Overview of Programming	55
5.1.1	Structure of the safety program	56
5.1.2	Fail-Safe Blocks	58
5.1.3	Restrictions in the programming languages FBD/LAD	59
5.2	Defining F-Runtime Groups	65
5.2.1	Rules for F-Runtime Groups of the Safety Program.....	65
5.2.2	Procedure for Defining an F-Runtime Group	66
5.2.3	Safety-Related Communication between F-Runtime Groups of a Safety Program	69
5.2.4	Deleting an F-runtime group	72
5.2.5	Modifying an F-runtime group.....	72

5.3	Creating F-blocks in FBD / LAD	73
5.3.1	Creating F-blocks	73
5.3.2	Use libraries	75
5.4	Programming startup protection.....	76
6	F-I/O access	79
6.1	F-I/O access	79
6.2	Process Data or Fail-Safe Values	80
6.3	F-I/O DB	82
6.4	Accessing F-I/O DB Variables.....	88
6.5	Passivation and reintegration of F-I/O	89
6.5.1	After startup of F-system.....	90
6.5.2	after communication errors	92
6.5.3	After F-I/O / channel faults	94
6.5.4	Group passivation	97
7	Implementation of user acknowledgment.....	99
7.1	Implementing User Acknowledgment in the Safety Program of the F-CPU of a DP Master or IO Controller.....	99
7.2	Implementing user acknowledgment in the safety program of the F-CPU of a I-slave or I- device.....	102
8	Data exchange between standard user program and safety program.....	105
8.1	Data Transfer from the Safety Program to the Standard User Program	105
8.1.1	F-Shared DB	107
8.2	Data Transfer from Standard User Program to Safety Program.....	108
9	Configuring and Programming Communication	111
9.1	Overview of communication	111
9.2	Safety-related IO controller-IO controller communication.....	114
9.2.1	Configure safety-related IO controller-IO controller communication.....	114
9.2.2	Safety-related IO controller-IO controller communication via SENDDP and RCVDP	117
9.2.3	Program safety-related IO controller-IO controller communication.....	118
9.2.4	Safety-related IO controller-IO controller communication - Limits for data transfer.....	121
9.3	Safety-related master-master communication	122
9.3.1	Configure safety-related master-master communication	122
9.3.2	Safety-related master-master communication via SENDDP and RCVDP.....	125
9.3.3	Program safety-related master-master communication	126
9.3.4	Safety-related master-master communication:Limits for data transfer	129
9.4	Safety-related communication between I/O-controller and I-device	130
9.4.1	Configuring safety-related communication between I/O-controller and I-device	130
9.4.2	Safety-related IO controller-IO device communication via SENDDP and RCVDP	133
9.4.3	Programming safety-related IO controller I-device communication	134
9.4.4	Safety-related IO-Controller-IO-Device communication - Limits for data transfer	135

9.5	Safety-related master-I-slave communication	136
9.5.1	Configuring safety-related master-I-slave communication	136
9.5.2	Safety-related master-I-slave or I-slave-I-slave communication via SENDDP and RCVDP	139
9.5.3	Program the safety-related master-I-slave or I-slave-I-slave communication	140
9.5.4	Limits for data transfer of safety-related master-I-slave or I-slave-I-slave communication	142
9.6	Safety-related I-slave-I-slave communication.....	143
9.6.1	Configure safety-related I-slave-I-slave communication.....	143
9.6.2	Safety-related I-slave-I-slave communication via SENDDP and RCVDP	147
9.6.3	Programming safety-related I-slave-I-slave communication.....	147
9.6.4	Limits for data transfer of safety-related I-slave-I-slave communication	147
9.7	Safety-Related I-Slave-Slave Communication.....	148
9.7.1	Configuring Safety-Related I-Slave-Slave Communication	148
9.7.2	Safety-Related I-Slave-Slave Communication - F-I/O Access	153
9.7.3	Limits for data transfer of safety-related I-slave-I-slave communication	153
9.8	Safety-related IO Controller-I-slave communication.....	154
9.9	Safety-related communication via S7 connections.....	155
9.9.1	Configuring safety-related communication via S7 connections	155
9.9.2	Communication via SENDS7, RCVS7, and F-Communication DB	157
9.9.3	Programming safety-related communication via S7 connections.....	158
9.9.4	Safety-related communication via S7 connections - Limits of data transfer.....	161
9.10	Safety-related communication with S7 F-systems.....	162
9.10.1	Introduction	162
9.10.2	Communication with S7 Distributed Safety via PN/PN coupler (IO Controller-IO Controller communication).....	162
9.10.3	Communication with S7 Distributed Safety via DP/DP coupler (master-master communication).....	163
9.10.4	Communication with S7 Distributed Safety via S7 connections	164
9.10.5	Communication with S7 F/FH Systems via S7 connections.....	166
10	Compiling and commissioning a safety program.....	169
10.1	Compiling the safety program	169
10.2	Downloading the Safety Program	171
10.3	Work Memory Requirement for Safety Program.....	177
10.4	Function Test of Safety Program and Protection through Program Identification	178
10.5	Comparing Safety Programs.....	183
10.6	Printing project data	187
10.7	Testing safety program	189
10.7.1	Overview of Testing the Safety Program	189
10.7.2	Deactivating Safety Mode	190
10.7.3	Testing the Safety Program	193
10.7.4	Modifying the safety program in RUN mode.....	197
10.7.5	Deleting the Safety Program.....	199

11	System Acceptance Test	201
11.1	Overview of System Acceptance Test	201
11.2	Correctness of the safety program including hardware configuration	202
11.3	Completeness of the safety printout	203
11.4	Compliance of the used instructions with the TÜV certificate	204
11.5	Correctness of the hardware configuration	205
11.6	Correctness of the communication configuration.....	208
11.7	Other characteristics	209
11.8	Acceptance Test of Changes	210
12	Operation and Maintenance.....	213
12.1	Notes on Safety Mode of the Safety Program	213
12.2	Replacing Software and Hardware Components.....	214
12.3	Guide to Diagnostics	216
13	STEP 7 Safety Advanced V11 Instructions.....	219
13.1	Overview of instructions.....	219
13.2	Instructions - LAD.....	220
13.2.1	General.....	220
13.2.1.1	New network (STEP 7 Safety Advanced V11).....	220
13.2.1.2	Empty box (STEP 7 Safety Advanced V11).....	221
13.2.1.3	Open branch (STEP 7 Safety Advanced V11).....	222
13.2.1.4	Close branch (STEP 7 Safety Advanced V11)	223
13.2.2	Bit logic operations.....	224
13.2.2.1	-- --: Normally open contact (STEP 7 Safety Advanced V11)	224
13.2.2.2	-- / --: Normally closed contact (STEP 7 Safety Advanced V11)	225
13.2.2.3	-- NOT --: Invert RLO (STEP 7 Safety Advanced V11).....	226
13.2.2.4	--()---: Assignment (STEP 7 Safety Advanced V11)	227
13.2.2.5	--(R)---: Reset output (STEP 7 Safety Advanced V11).....	228
13.2.2.6	--(S)---: Set output (STEP 7 Safety Advanced V11)	229
13.2.2.7	SR: Set/reset flip-flop (STEP 7 Safety Advanced V11)	230
13.2.2.8	RS: Reset/set flip-flop (STEP 7 Safety Advanced V11).....	232
13.2.2.9	-- P --: Scan operand for positive signal edge (STEP 7 Safety Advanced V11).....	234
13.2.2.10	-- N --: Scan operand for negative signal edge (STEP 7 Safety Advanced V11)	236
13.2.2.11	P_TRIG: Scan RLO for positive signal edge (STEP 7 Safety Advanced V11).....	237
13.2.2.12	N_TRIG: Scan RLO for negative signal edge (STEP 7 Safety Advanced V11)	239
13.2.3	Safety functions.....	240
13.2.3.1	ESTOP1: Emergency Stop up to Stop Category 1 (STEP 7 Safety Advanced V11)	240
13.2.3.2	TWO_HAND: Two-hand monitoring (STEP 7 Safety Advanced V11)	246
13.2.3.3	TWO_H_EN: Two-hand monitoring with enable (STEP 7 Safety Advanced V11).....	249
13.2.3.4	MUTING: Muting (STEP 7 Safety Advanced V11).....	254
13.2.3.5	MUT_P: Parallel muting (STEP 7 Safety Advanced V11).....	265
13.2.3.6	EV1oo2DI: 1oo2 evaluation with discrepancy analysis (STEP 7 Safety Advanced V11).....	276
13.2.3.7	FDBACK: Feedback monitoring (STEP 7 Safety Advanced V11)	283
13.2.3.8	SFDOOR: Safety door monitoring (STEP 7 Safety Advanced V11).....	289
13.2.3.9	ACK_GL: Global acknowledgment of all F-I/O in an F-runtime group (STEP 7 Safety Advanced V11).....	295

13.2.4	Timer operations	297
13.2.4.1	TP: Generate pulse (STEP 7 Safety Advanced V11)	297
13.2.4.2	TON: Generate on-delay (STEP 7 Safety Advanced V11)	300
13.2.4.3	TOF: Generate off-delay (STEP 7 Safety Advanced V11)	303
13.2.5	Counter operations	306
13.2.5.1	CTU: Count up (STEP 7 Safety Advanced V11)	306
13.2.5.2	CTD: Count down (STEP 7 Safety Advanced V11)	307
13.2.5.3	CTUD: Count up and down (STEP 7 Safety Advanced V11)	309
13.2.6	Comparator operations	311
13.2.6.1	CMP ==: Equal (STEP 7 Safety Advanced V11)	311
13.2.6.2	CMP <>: Not equal (STEP 7 Safety Advanced V11)	312
13.2.6.3	CMP >=: Greater than or equal (STEP 7 Safety Advanced V11)	313
13.2.6.4	CMP <=: Less than or equal (STEP 7 Safety Advanced V11)	314
13.2.6.5	CMP >: Greater than (STEP 7 Safety Advanced V11)	315
13.2.6.6	CMP <: Less than (STEP 7 Safety Advanced V11)	316
13.2.7	Math functions	317
13.2.7.1	ADD: Add (STEP 7 Safety Advanced V11)	317
13.2.7.2	SUB: Subtract (STEP 7 Safety Advanced V11)	319
13.2.7.3	MUL: Multiply (STEP 7 Safety Advanced V11)	321
13.2.7.4	DIV: Divide (STEP 7 Safety Advanced V11)	323
13.2.7.5	NEG: Create twos complement (STEP 7 Safety Advanced V11)	325
13.2.8	Move operations	327
13.2.8.1	MOVE: Move value (STEP 7 Safety Advanced V11)	327
13.2.8.2	WR_FDB: Write value indirectly to an F-DB (STEP 7 Safety Advanced V11)	328
13.2.8.3	RD_FDB: Read value indirectly from an F-DB (STEP 7 Safety Advanced V11)	330
13.2.9	Conversion operations	331
13.2.9.1	CONVERT: Convert value (STEP 7 Safety Advanced V11)	331
13.2.9.2	BO_W: Convert 16 data elements of data type BOOL to a data element of data type WORD (STEP 7 Safety Advanced V11)	332
13.2.9.3	W_BO: Convert a data element of data type WORD to 16 data elements of data type BOOL (STEP 7 Safety Advanced V11)	334
13.2.9.4	SCALE: Scale values (STEP 7 Safety Advanced V11)	336
13.2.10	Program control operations	338
13.2.10.1	---(JMP): Jump if RLO = 1 (STEP 7 Safety Advanced V11)	338
13.2.10.2	---(JMPN): Jump if RLO = 0 (STEP 7 Safety Advanced V11)	339
13.2.10.3	LABEL: Jump label (STEP 7 Safety Advanced V11)	340
13.2.10.4	--(RET): Return (STEP 7 Safety Advanced V11)	341
13.2.10.5	---(OPN): Open global data block (STEP 7 Safety Advanced V11)	342
13.2.11	Word logic operations	344
13.2.11.1	AND: AND logic operation (STEP 7 Safety Advanced V11)	344
13.2.11.2	OR: OR logic operation (STEP 7 Safety Advanced V11)	345
13.2.11.3	XOR: EXCLUSIVE OR logic operation (STEP 7 Safety Advanced V11)	346
13.2.12	Shift and rotate	347
13.2.12.1	SHR: Shift right (STEP 7 Safety Advanced V11)	347
13.2.12.2	SHL: Shift left (STEP 7 Safety Advanced V11)	349
13.2.13	Operating	351
13.2.13.1	ACK_OP: Fail-safe acknowledgment (STEP 7 Safety Advanced V11)	351
13.2.14	Additional instructions	354
13.2.14.1	--- --- OV: Get status bit OV (STEP 7 Safety Advanced V11)	354
13.2.14.2	--- / --- OV: Get negated status bit OV (STEP 7 Safety Advanced V11)	355

13.2.15	Communication	356
13.2.15.1	PROFIBUS/PROFINET	356
13.2.15.2	S7 communication	363
13.3	Instructions - FBD	369
13.3.1	General.....	369
13.3.1.1	New network (STEP 7 Safety Advanced V11).....	369
13.3.1.2	Empty box (STEP 7 Safety Advanced V11).....	370
13.3.1.3	Open branch (STEP 7 Safety Advanced V11).....	371
13.3.1.4	Insert binary input (STEP 7 Safety Advanced V11).....	372
13.3.1.5	Invert RLO (STEP 7 Safety Advanced V11).....	373
13.3.2	Bit logic operations.....	374
13.3.2.1	AND logic operation (STEP 7 Safety Advanced V11).....	374
13.3.2.2	OR logic operation (STEP 7 Safety Advanced V11).....	376
13.3.2.3	X: EXCLUSIVE OR logic operation (STEP 7 Safety Advanced V11).....	377
13.3.2.4	=: Assignment (STEP 7 Safety Advanced V11).....	379
13.3.2.5	R: Reset output (STEP 7 Safety Advanced V11).....	380
13.3.2.6	S: Set output (STEP 7 Safety Advanced V11).....	381
13.3.2.7	SR: Set/reset flip-flop (STEP 7 Safety Advanced V11).....	383
13.3.2.8	RS: Reset/set flip-flop (STEP 7 Safety Advanced V11).....	385
13.3.2.9	P: Scan operand for positive signal edge (STEP 7 Safety Advanced V11).....	387
13.3.2.10	N: Scan operand for negative signal edge (STEP 7 Safety Advanced V11).....	389
13.3.2.11	P_TRIG: Scan RLO for positive signal edge (STEP 7 Safety Advanced V11).....	391
13.3.2.12	N_TRIG: Scan RLO for negative signal edge (STEP 7 Safety Advanced V11).....	393
13.3.3	Safety functions.....	395
13.3.3.1	ESTOP1: Emergency Stop up to Stop Category 1 (STEP 7 Safety Advanced V11).....	395
13.3.3.2	TWO_HAND: Two-hand monitoring (STEP 7 Safety Advanced V11).....	401
13.3.3.3	TWO_H_EN: Two-hand monitoring with enable (STEP 7 Safety Advanced V11).....	405
13.3.3.4	MUTING: Muting (STEP 7 Safety Advanced V11).....	410
13.3.3.5	MUT_P: Parallel muting (STEP 7 Safety Advanced V11).....	421
13.3.3.6	EV1oo2DI: 1oo2 evaluation with discrepancy analysis (STEP 7 Safety Advanced V11).....	432
13.3.3.7	FDBACK: Feedback monitoring (STEP 7 Safety Advanced V11).....	439
13.3.3.8	SFDOOR: Safety door monitoring (STEP 7 Safety Advanced V11).....	445
13.3.3.9	ACK_GL: Global acknowledgment of all F-I/O in an F-runtime group (STEP 7 Safety Advanced V11).....	451
13.3.4	Timer operations	453
13.3.4.1	TP: Generate pulse (STEP 7 Safety Advanced V11).....	453
13.3.4.2	TON: Generate on-delay (STEP 7 Safety Advanced V11).....	456
13.3.4.3	TOF: Generate off-delay (STEP 7 Safety Advanced V11).....	459
13.3.5	Counter operations.....	462
13.3.5.1	CTU: Count up (STEP 7 Safety Advanced V11).....	462
13.3.5.2	CTD: Count down (STEP 7 Safety Advanced V11).....	464
13.3.5.3	CTUD: Count up and down (STEP 7 Safety Advanced V11).....	466
13.3.6	Comparator operations	468
13.3.6.1	CMP ==: Equal (STEP 7 Safety Advanced V11).....	468
13.3.6.2	CMP <>: Not equal (STEP 7 Safety Advanced V11).....	469
13.3.6.3	CMP >=: Greater than or equal (STEP 7 Safety Advanced V11).....	470
13.3.6.4	CMP <=: Less than or equal (STEP 7 Safety Advanced V11).....	471
13.3.6.5	CMP >: Greater than (STEP 7 Safety Advanced V11).....	472
13.3.6.6	CMP <: Less than (STEP 7 Safety Advanced V11).....	473

13.3.7	Math functions.....	474
13.3.7.1	ADD: Add (STEP 7 Safety Advanced V11)	474
13.3.7.2	SUB: Subtract (STEP 7 Safety Advanced V11).....	476
13.3.7.3	MUL: Multiply (STEP 7 Safety Advanced V11).....	478
13.3.7.4	DIV: Divide (STEP 7 Safety Advanced V11)	480
13.3.7.5	NEG: Create twos complement (STEP 7 Safety Advanced V11).....	482
13.3.8	Move operations	484
13.3.8.1	MOVE: Move value (STEP 7 Safety Advanced V11)	484
13.3.8.2	WR_FDB: Write value indirectly to an F-DB (STEP 7 Safety Advanced V11)	485
13.3.8.3	RD_FDB: Read value indirectly from an F-DB (STEP 7 Safety Advanced V11).....	487
13.3.9	Conversion operations.....	488
13.3.9.1	CONVERT: Convert value (STEP 7 Safety Advanced V11)	488
13.3.9.2	BO_W: Convert 16 data elements of data type BOOL to a data element of data type WORD (STEP 7 Safety Advanced V11)	488
13.3.9.3	W_BO: Convert a data element of data type WORD to 16 data elements of data type BOOL (STEP 7 Safety Advanced V11)	490
13.3.9.4	SCALE: Scale values (STEP 7 Safety Advanced V11)	491
13.3.10	Program control operations.....	493
13.3.10.1	JMP: Jump if RLO = 1 (STEP 7 Safety Advanced V11).....	493
13.3.10.2	JMPN: Jump if RLO = 0 (STEP 7 Safety Advanced V11)	494
13.3.10.3	LABEL: Jump label (STEP 7 Safety Advanced V11).....	495
13.3.10.4	RET: Return (STEP 7 Safety Advanced V11)	496
13.3.10.5	OPN: Open global data block (STEP 7 Safety Advanced V11)	497
13.3.11	Word logic operations	499
13.3.11.1	AND: AND logic operation (STEP 7 Safety Advanced V11)	499
13.3.11.2	OR: OR logic operation (STEP 7 Safety Advanced V11).....	500
13.3.11.3	XOR: EXCLUSIVE OR logic operation (STEP 7 Safety Advanced V11)	501
13.3.12	Shift and rotate.....	502
13.3.12.1	SHR: Shift right (STEP 7 Safety Advanced V11)	502
13.3.12.2	SHL: Shift left (STEP 7 Safety Advanced V11)	504
13.3.13	Operating	506
13.3.13.1	ACK_OP: Fail-safe acknowledgment (STEP 7 Safety Advanced V11).....	506
13.3.14	Additional instructions	509
13.3.14.1	OV: Get status bit OV (STEP 7 Safety Advanced V11).....	509
13.3.15	Communication	510
13.3.15.1	PROFIBUS/PROFINET	510
13.3.15.2	S7 communication	517
A	Monitoring and response times.....	523
A.1	Configuring the monitoring times	524
A.1.1	Minimum Monitoring Time for F-Cycle Time.....	525
A.1.2	Minimum monitoring time for safety-related communication between F-CPU and F-I/O	526
A.1.3	Minimum monitoring time of safety-related CPU-CPU communication	527
A.1.4	Monitoring Time for Safety-Related Communication between F-Runtime Groups	528
A.2	Response Times of Safety Functions	528
B	Checklist.....	529
	Glossary	533
	Index.....	543

Product Overview

1.1 Overview

SIMATIC Safety fail-safe system

The SIMATIC Safety fail-safe system is available to implement safety concepts in the area of machine and personnel protection (for example, for emergency STOP devices for machining and processing equipment) and in the process industry (for example, for implementation of protection functions for safety devices of instrumentation and controls and of burners).

Achievable safety requirements

SIMATIC Safety F-systems can satisfy the following safety requirements:

- Safety class (Safety Integrity Level) SIL1 to SIL3 in accordance with IEC 61508:2010
- Category Cat.2 to Cat.4 in accordance with EN 954-1
- Performance Level (PL) a to e in accordance with ISO 13849-1: 2006 or in accordance with EN ISO 13849-1: 2008

Principles of safety functions in SIMATIC Safety

Functional safety is implemented principally through safety functions in the software. Safety functions are executed by the SIMATIC Safety system in order to bring the system to a safe state or maintain it in a safe state in case of a dangerous event. Safety functions are contained mainly in the following components:

- In the safety-related user program (safety program) in the F-CPU
- In the fail-safe inputs and outputs (F-I/O)

The F-I/O ensure the safe processing of field information (sensors: e.g., emergency OFF pushbutton, light barriers, actuators for motor control, for example). They have all of the required hardware and software components for safe processing, in accordance with the required Safety Integrity Level. The user only has to program the user safety function. The safety function for the process can be provided through a user safety function or a fault reaction function. In the event of an error, if the F-system can no longer execute its actual user safety function, it executes the fault reaction function; for example, the associated outputs are shut down, and the F-CPU switches to STOP mode, if necessary.

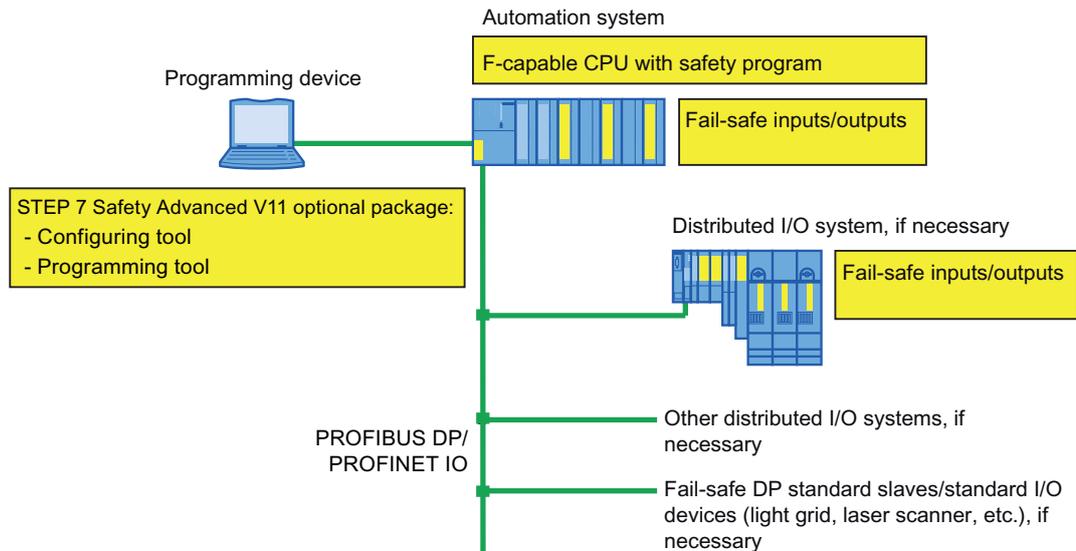
Example of user safety function and fault reaction function

In the event of overpressure, the F-system will open a valve (user safety function). In case of a dangerous malfunction of the F-CPU, all outputs will be shut down. For a non-faulty F-system, only the valve would be opened.

1.2 Hardware and Software Components

Hardware and software components of SIMATIC Safety

The following figure provides an overview of the hardware and software components required to configure and operate a SIMATIC Safety F-system.



Hardware components for PROFIBUS DP

You can use the following fail-safe components in SIMATIC Safety F-systems on PROFIBUS DP:

- PROFIBUS DP with DP interface, such as CPU 315F-2 DP
- Fail-safe inputs and outputs (F-I/O), such as:
 - S7-300 fail-safe signal modules in S7-300 (centralized configuration)
 - S7-300 fail-safe signal modules in ET 200M (distributed configuration)
 - ET 200S fail-safe modules
 - ET 200eco fail-safe I/O modules
 - ET 200pro fail-safe modules
 - ET 200iSP fail-safe modules
 - Fail-safe DP standard slaves (light grid, laser scanner, etc.)

You can expand the configuration using standard I/O.

The following CPs can be used in a SIMATIC Safety F-system on PROFIBUS DP:

- CP 443-5 Basic
- CP 443-5 Extended

Hardware components for PROFINET IO

You can use the following fail-safe components in SIMATIC Safety F-systems on PROFINET IO:

- F-CPU with PN interface, such as CPU 416F-3 PN/DP
- Fail-safe inputs and outputs (F-I/O), such as:
 - S7-300 fail-safe signal modules in S7-300 (centralized configuration)
 - S7-300 fail-safe signal modules in ET 200M (distributed configuration)
 - ET 200S fail-safe modules
 - ET 200pro fail-safe modules
 - ET 200iSP fail-safe modules
 - Fail-safe standard I/O devices (light grid, laser scanner, etc.)

You can expand the configuration using standard I/O.

The following CPs can be used in a SIMATIC Safety F-system on PROFINET IO:

- CP 443-1
- CP 443-1 Advanced-IT

Required software components

You require the following software components:

- *SIMATIC STEP 7 Professional V11 SP1*
- *STEP 7 Safety Advanced V11* optional package

STEP 7 Safety Advanced V11 optional package

This documentation describes the *STEP 7 Safety Advanced V11* optional package. *STEP 7 Safety Advanced V11* is the configuration and programming software for the SIMATIC Safety F-system. With *STEP 7 Safety Advanced V11*, you receive the following:

- Support for configuring the F-I/O in the *hardware and network editor* of the TIA portal
- Support for creating the safety program using LAD and FBD and integrating error detection functions into the safety program
- Instructions for programming your safety program in LAD and FBD, which you are familiar with from the standard user programs
- Instructions for programming your safety program in LAD and FBD with special safety functions

Moreover, *STEP 7 Safety Advanced V11* offers functions for comparing safety programs and for supporting you in the system acceptance test.

Safety program

You can create a safety program using the *program editor*. You can program fail-safe FBs and FCs in the FBD or LAD programming languages using the instructions from the optional package and create fail-safe DBs.

Safety checks are automatically performed and additional fail-safe blocks for error detection and error response are inserted when the safety program is compiled. This ensures that failures and faults are detected and appropriate reactions are triggered to maintain the F-system in the safe state or bring it to a safe state.

In addition to the safety program, a standard user program can be run on the F-CPU. A standard program can coexist with a safety program in an F-CPU because the safety-related data of the safety program are protected from being affected unintentionally by data of the standard user program.

Data can be exchanged between the safety program and the standard user program in the F-CPU by means of bit memory or data of a standard DB or by accessing the process image inputs and outputs.

See also

Data Transfer from the Safety Program to the Standard User Program (Page 105)

1.3 Installation/uninstallation of the STEP 7 Safety Advanced V11 optional package

Software requirements for *STEP 7 Safety Advanced V11*

At a minimum, the following software package must be installed on the programming device or PC:

- *SIMATIC STEP 7 Professional V11 SP1*

Reviewing the Readme file

The readme file contains important up-to-date information about the software (for example, Windows versions supported). You can display the readme file in the setup program or open it at a later time in the *SIMATIC STEP 7 Professional* information system.

Installing *STEP 7 Safety Advanced V11*

1. Start the programming device or PC on which the "*SIMATIC STEP 7 Professional*" software package has been installed, and make sure that *SIMATIC STEP 7 Professional* is closed.
2. Insert the optional package product DVD.
3. Initiate the *SETUP.EXE* program on the DVD.
4. Follow the instructions of the Setup program, bearing in mind the information in the readme file.

Displaying integrated Help

The help on *STEP 7 Safety Advanced V11* is completely integrated into the information system of *SIMATIC STEP 7 Professional*. You have the following two options to open the integrated help:

- In the "Help" menu, select the "Show help" command or press <F1> to show the appropriate help for the context.
- Click on the link within a tool tip cascade to go directly to a secondary place within the help.

Uninstalling *STEP 7 Safety Advanced V11*

To uninstall *STEP 7 Safety Advanced V11* proceed as described in the Help on *STEP 7 Professional* in the "Uninstallation" section.

Post-uninstall procedures

After uninstalling the *STEP 7 Safety Advanced V11* optional package, you can no longer open projects with F-CPU's whose F-capability is activated.

You may open and continue working with projects with F-CPU's whose F-capability was previously deactivated (see also Configuring the F-CPU (Page 29)).

1.4 Migration of projects from S7 Distributed Safety V5.4 SP5 to STEP 7 Safety Advanced V11

Introduction

In *STEP 7 Safety Advanced V11* you may continue projects with safety programs that you have created with *S7 Distributed Safety V5.4 SP5*. **To that end, the projects must have been compiled in *S7 Distributed Safety V5.4 SP5* and then migrated.**

Procedure as in *STEP 7 Professional*

You should proceed as you would for standard projects for the migration of projects from *S7 Distributed Safety V5.4 SP5* to *STEP 7 Safety Advanced V11*. Once the migration is complete, you should verify using the collective F-signature whether the project was migrated unchanged.

This migration approach is described in the "Migration" section of the *STEP 7 Professional* Help. Please find below the particularities for the *STEP 7 Safety Advanced V11*.

Older hardware versions

Older versions of F-hardware may not be supported by *STEP 7 Safety Advanced V11*.

If you have used and configured versions of F-CPU and F-I/O in *S7 Distributed Safety* projects that are not approved for *STEP 7 Safety Advanced V11*, you must upgrade this hardware to the new version in *S7 Distributed Safety V5.4 SP5*. The migration to *STEP 7 Safety Advanced V11* is then possible. A list of approved hardware is available on the Internet. (<http://support.automation.siemens.com/WW/view/en/49368678/133300>)

Particularities for safety-related CPU-CPU communication via S7 connections

For particularities of migrated projects with safety-related CPU-CPU communication via S7 connections, please see Safety-related communication via S7 connections (Page 155). Note also Communication with S7 Distributed Safety via S7 connections (Page 164).

Particularities for ESTOP1 or FDBACK instructions

Information on particularities when using the ESTOP1 and FDBACK instructions can be found in the "Instruction versions" section under ESTOP1: Emergency Stop up to Stop Category 1 (*STEP 7 Safety Advanced V11*) (Page 240) and FDBACK: Feedback monitoring (*STEP 7 Safety Advanced V11*) (Page 283).

Post-migration procedures

Once migration is complete, you obtain a complete project with a safety program which has maintained the program structure of *S7 Distributed Safety* and the collective F-signature. Therefore, the migrated project should not be considered a new one and can be loaded as it is to the F-CPU as long as it has not been modified after migration.

Note

Safety printout

You cannot create a safety printout in *STEP 7 Safety Advanced V11* for a migrated project. The printout of the project created with *S7 Distributed Safety V5.4 SP5* and the corresponding acceptance documents are still valid, because the collective F-signature has been retained.

Compiling the migrated safety programs

As a result of the compilation of the migrated project with *STEP 7 Safety Advanced V11*, the existing program structure (with F-CALL) is transformed into the new program structure of *STEP 7 Safety Advanced V11* (with main safety block). Moreover, F-blocks from the *S7 Distributed Safety (V1)* F-library are converted to instructions that are provided by *STEP 7 Safety Advanced V11*. This changes the collective F-signature and the safety program must be re-accepted, if necessary.

Note

Note that compiling the migrated safety program extends the runtime of the F-runtime group(s) and increases the memory requirements of the safety program (see also Excel file for calculating response time (<http://support.automation.siemens.com/WW/view/en/49368678/133100>)).

1.5 First steps

Getting Started in SIMATIC Safety

The Getting Started documentation is available to help you begin using SIMATIC Safety.

The Getting Started documentation is an instruction manual that provides a step-by-step description of how to create a project with SIMATIC Safety. It gives you the opportunity to quickly become familiar with the scope of performance of SIMATIC Safety.

Contents

The Getting Started documentation describes the creation of a single, continuous project that is extended with each chapter. Based on the configuration, you program a fail-safe shutdown, make changes to the programming, and accept the system.

In addition to the step-by-step instructions, the Getting Started documentation also gives you background information for every new topic, which explains the functions used in more detail and how they interrelate.

Target audience

The Getting Started documentation addresses beginners but is also suitable for users that are switching from S7 Distributed Safety.

Download

The Getting Started documentation is available as a PDF (<http://support.automation.siemens.com/WW/view/en/49972838>) in the Service&Support Portal free of charge.

Configuration

2.1 Overview of Configuration

Introduction

You configure a SIMATIC Safety F-system in basically the same way as a standard S7-300, S7-400, or ET 200S/ET 200pro automation system in *STEP 7 Professional*.

This section presents only the essential differences compared to standard configuration you encounter when configuring an SIMATIC Safety F-system.

Which F-components can you configure?

You configure the following hardware components for a SIMATIC Safety F-system:

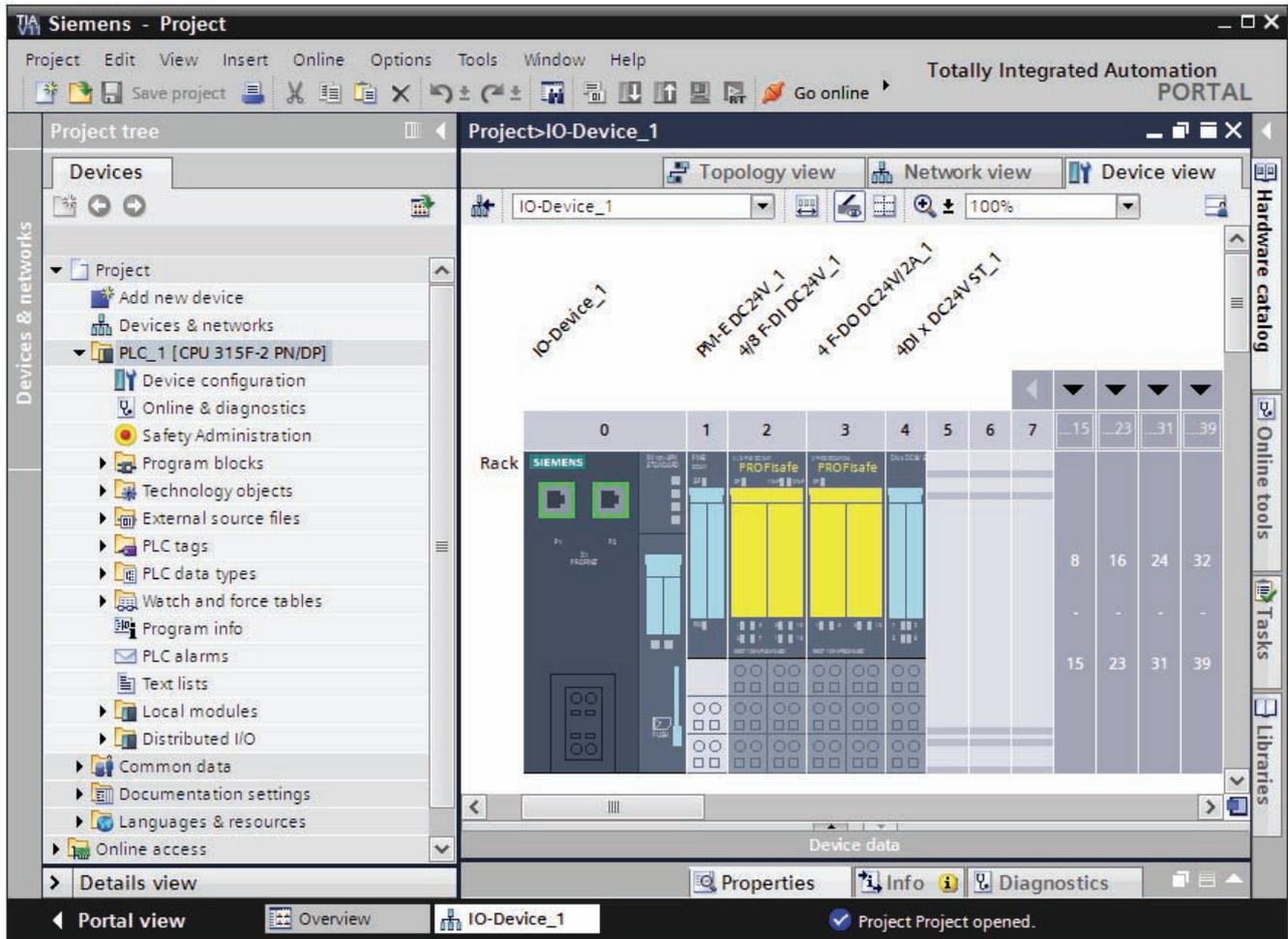
- F-CPU, such as CPU 315F-2 PN/DP
- F-I/O, such as:
 - ET 200S fail-safe modules
 - S7-300 fail-safe signal modules (for centralized configuration next to the F-CPU or distributed configuration in ET 200M)
 - ET 200pro fail-safe modules
 - ET 200iSP fail-safe modules
 - ET 200eco fail-safe I/O modules
 - Fail-safe DP standard slaves
 - Fail-safe standard I/O devices

The following CPs can be used in a SIMATIC Safety F-system:

- CP 443-5 Basic
- CP 443-5 Extended
- CP 443-1
- CP 443-1 Advanced-IT

Example: Configured F-system in STEP 7 Professional

The following figure presents a configured F-system. You choose the fail-safe components in the "Hardware catalog" task card as you would do with standard components and place them in the work area of the network or device view. F-components are shown in yellow.



Additional information

For detailed information on F-I/O, refer to the *manuals for the relevant F-I/O*.

Which safety-related communication options can you configure?

You must use the *hardware and network editor* to configure the following safety-related communication options (see Configuring and Programming Communication (Page 111)):

- Safety-related master-master communication
- Safety-related master-master communication for S7 Distributed Safety
- Safety-related master-I-slave communication
- Safety-related I-slave-I-slave communication
- Safety-related I-slave-slave communication
- Safety-related IO Controller-IO Controller communication
- Safety-related IO Controller-IO Controller communication for S7 Distributed Safety
- Safety-related IO Controller-I-Device communication
- Safety-related IO Controller-I-slave communication
- Safety-related communication via S7 connections
- Safety-related communication via S7 connections for S7 Distributed Safety or S7 F Systems

2.2 Particularities for configuring the F-System

Configuring same as for standard components

You configure a SIMATIC Safety F-system in the same way as a standard S7 system. That is, you configure the hardware and assign the hardware parameters in the *hardware and network editor*, as a centralized configuration (F-CPU and, if necessary, F-I/O, such as CPU 315F-2 PN/DP and S7-300 F-SMs) and/or as a decentralized (distributed) configuration (F-CPU, F-SMs in ET 200M, F-modules in ET 200S, ET 200 pro, ET 200iSP, and ET 200eco, fail-safe DP standard slaves, fail-safe standard I/O devices).

Special F-parameters

For the F-functionality there are special F-parameters that you can review and set in the "Properties" of the fail-safe components (F-CPU and F-I/O). F-parameters are marked in yellow.

F-parameters are explained in "Configuring the F-CPU (Page 29)" and "Configuring the F-I/O (Page 33)".

Compiling the hardware configuration

You must compile the hardware configuration of the SIMATIC Safety F-system (context menu "Compile > Hardware configuration"). A configured F-CPU with activated F-capability is the only prerequisite for programming the safety program.

Note

Inconsistencies are possible when configuring the hardware and can also be saved. A consistency check of the hardware configuration and possible connection data is done only on compilation. Therefore, perform "Edit > Compile" regularly.

Changing safety-related parameters

Note

If you change a safety-relevant parameter (marked in yellow) for an F-I/O or an F-CPU, you must compile the safety program (Page 169).

2.3 Configuring the F-CPU

Introduction

You configure the F-CPU basically the same way as a standard automation system.

F-CPU's are always configurable *STEP 7 Professional*, regardless of whether the *STEP 7 Safety Advanced V11* optional package is installed or not. Without an installed optional package, however, the F-CPU can only be used as a standard CPU.

When the *STEP 7 Safety Advanced V11* optional package is installed, you can activate or deactivate the F-capability for the F-CPU.

The F-capability is activated by default.

Activating/deactivating F-capability

If you want to modify the F-capability setting, proceed as follows:

1. Select the F-CPU in the device or network view, and select the "Properties" tab in the inspector window.
2. Open the "F-parameters".
3. Use the appropriate button to activate/deactivate the F-capability.
4. If you want to deactivate F-capability, confirm the "Turning off the F-activation" dialog with "Yes".

Deactivating the F-capability with existing safety program

If you want to deactivate the F-capability for an F-CPU, because you intend to use the F-CPU as a standard CPU, although a safety program is present, you must note following:

- You require the password for the F-CPU, if assigned.
- The Safety Administration Editor (Page 39) is deleted from the project tree.
- All F-blocks outside the "System blocks" folder are marked as no longer supported  in the project tree. They can no longer be opened.

All F-blocks within the "System blocks" folder are deleted.

A compiling error is reported in the next compilation.

- From now on you cannot use F-I/O in safety mode with this F-CPU. (Exception: if you access the F-I/O via safety-related I-slave-slave communication.)

Configuring the F-parameters of the F-CPU

In the "Properties" tab of the F-CPU, you can change or apply the default settings for the following parameters:

- Base for PROFIsafe addresses
- F-monitoring time of the F-CPU

Note

A change of the F-monitoring time of the F-CPU results in modifications to the safety program when it is recompiled, and consequently, a new acceptance test may be required.

"Base for PROFIsafe addresses" parameter

This information is required for internal administration of the PROFIsafe addresses of the F-system. The PROFIsafe destination address is used to uniquely identify the destination.

Setting this parameter defines a range for the automatic assignment of PROFIsafe destination addresses. This is useful if several DP master systems and PROFINET IO systems are operated on one network. Subsequent address changes are possible.

You can specify the "Base for PROFIsafe addresses" in increments of 100. The next free PROFIsafe destination address is selected for automatic assignment of PROFIsafe destination addresses. The maximum possible PROFIsafe destination address is 65534, for F-I/O with DIP switch, the maximum possible PROFIsafe destination address is 1022.

Example: You set "200" as the basic address. The PROFIsafe destination address is then automatically assigned in ascending order, starting from 200.

Note

The "Base for PROFIsafe addresses" parameter has no influence on the following F-I/O:

- SM 326; DI 8 x NAMUR (Order No. 6ES7326-1RF00-0AB0)
 - SM 326; DO 10 x DC 24V/2A (Order No. 6ES7326-2BF01-0AB0)
 - SM 336; AI 6 x 13 Bit (Order No. 6ES7336-1HE00-0AB0)
-

Parameter "F-monitoring time"

For monitoring of communication between F-CPU and F-I/O, you configure the PROFIsafe monitoring time for the F-CPU and its assigned F-I/O.

You can adjust the F-monitoring time for the F-CPU via the following parameters:

- "Default F-monitoring time for central F-I/O"
- "Default F-monitoring time for F-I/O of this interface"

The **F-monitoring time for central F-I/O** acts on the F-I/O that are arranged centrally, i.e., near the F-CPU. You set this parameter in the properties of the F-CPU (select F-CPU, then select "Properties > Fail-safe > F-parameters").

The **F-monitoring time for the F-I/O of this interface** acts on the F-I/O that are assigned to this interface of the F-CPU (PROFIBUS or PROFINET). You change this parameter in the properties of the relevant interface (select the interface in the "Device view" tab, then "Properties" > Interface > F-parameters").

Through the different setting options available, you can adapt the F-monitoring time variably to the conditions of your F-system, for instance, by taking into account different bus cycles.

You can also change the F-monitoring time individually for every F-I/O in the F-I/O Properties (see Configuring the F-I/O (Page 33)).

 WARNING
<p>It can only be ensured (from a fail-safe standpoint) that a signal level to be transferred will be captured on the sender side and transferred to the receiver if the signal level is pending for at least as long as the assigned monitoring time. (S018)</p>

Additional information can be found in Monitoring and response times (Page 523).

Allow automatic generation of the safety program

The safety program of an F-CPU consists of one or two F-runtime groups that contain the F-blocks (see also Defining F-Runtime Groups (Page 65)). When the F-CPU (with activated F-capability) is inserted into the work area of the device view or network view, an F-runtime group is generated automatically.

You can define in *STEP 7 Safety Advanced V11* that no F-runtime group is generated when inserting the F-CPU (with activated F-capability).

Proceed as follows:

1. Select the "Options > Settings" menu command.
2. Select the "STEP 7 Safety" area.
3. Activate/deactivate the automatic generation of an F-runtime group by selecting/deselecting the "Generate default fail-safe program" option.

This change has no influence on any existing safety programs, but only defines whether an F-runtime group is automatically generated for each one of the subsequently inserted F-CPU.

Configuring the protection level of the F-CPU

 **WARNING**

In safety mode, no access may be granted by the CPU password when changes are made in the standard user program, because changes can then also be made to the safety program. To rule out this possibility, you must configure **protection level "Write protection for fail-safe blocks"** and configure a password for the F-CPU. If only **one person** is authorized to change the standard user program **and** the safety program, then the protection level "Write protection" or "Write/Read" should be configured so that other persons have only limited access or no access at all to the entire user program (standard and safety programs). (See also Access protection (Page 49)) (*S001*)

Use the following procedure to configure the "Write protection for fail-safe blocks" protection level:

1. Select the F-CPU in the device or network view, and select the "Properties" tab in the inspector window.
2. Open the "Protection" area.
3. Set the corresponding protection level.
4. Enter a password for the write-read access in the fields designated for that purpose to activate write protection.

For information on the password for F-CPU, refer to Access protection (Page 49). Pay special attention to the warnings in Setting up access permission for the F-CPU (Page 54).

2.4 Configuring the F-I/O

Introduction

You configure the ET 200S, ET 200eco, ET 200pro, ET 200iSP F-modules and the S7-300 F-SMs as usual in *STEP 7 Professional*.

After you have inserted the F-I/O in the work area of *Device or network view*, you access the configuration dialogs by selecting the relevant F-I/O and the "Properties" tab.

Channel-granular passivation after channel faults

You can configure how the F-I/O will respond to channel faults, such as a short circuit, overload, discrepancy error, or wire break, provided the F-I/O supports this parameter (e.g., for ET 200S or ET 200pro F-modules). You configure this behavior in the properties for the relevant F-I/O ("Behavior after channel faults" parameter). This parameter is used to specify whether the entire F-I/O or just the faulty channel(s) are passivated in the event of channel faults.

Note

Note that channel-level passivation increases the runtime of the F-runtime group(s) compared to passivation of the entire F-I/O (see also Excel file for response time calculation (<http://support.automation.siemens.com/WW/view/en/49368678/133100>)).

Changing the name and number of the F-I/O DB

Name and number of the F-I/O DB (Page 82) are assigned automatically when configuring the F-I/O. You can change the number in the properties of the F-I/O. Alternatively you can change the name in the project tree via the context menu for the F-I/O DB.

You can find information on the assignment of number ranges in *Safety Administration Editor*, "Settings" tab (Page 45).

F-destination address

The F-destination address uniquely identifies the PROFIsafe destination (of the F-I/O). Therefore, the F-destination address must be unique network-wide and station-wide (see the following rules for address assignment).

To prevent incorrect parameter assignment, a station-wide unique F-destination address is automatically assigned during placement of the F-I/O in the work area of the device or network view.

To ensure a network-wide unique F-destination address assignment when multiple DP master systems and PROFINET IO systems are operated on one network, you must set the "Base for PROFIsafe addresses" parameter (in the properties of the F-CPU) in SIMATIC Safety F-systems differently, before placing the F-I/O in the various stations of a network.

Network-wide checking is only possible if the entire network is present in the project.

If you change the F-destination address, the station-wide uniqueness of the F-destination address is checked automatically. You yourself must ensure the network-wide uniqueness of the F-destination address.

You must set the F-destination address on the F-I/O using the DIP switch before installing the F-I/O.

Note

For the following fail-safe S7-300 signal modules, the F-destination address = the start address of the F-SM/8:

- SM 326; DI 8 x NAMUR (Order No. 6ES7326-1RF00-0AB0)
- SM 326; DO 10 x DC 24V/2A (Order No. 6ES7326-2BF01-0AB0)
- SM 336; AI 6 x 13 Bit (Order No. 6ES7336-1HE00-0AB0)

As long as the start address of the F-SM lies in the process image, the F-destination address is assigned automatically based on the "Base for PROFIsafe address" . Otherwise, the next free start address and F-destination address is assigned for these F-SMs beginning with 1.

Rules for address assignment

WARNING

The following applies to pure PROFIBUS subnets only:

The PROFIsafe destination address and, thus, the switch setting on the address switch of the F-I/O must be unique network-wide* and station-wide** (system-wide). For S7-300 F-SMs and ET 200S, ET 200eco, ET 200pro, and ET 200iSP F-modules, you can assign a maximum of 1022 different PROFIsafe destination addresses.

Exception: The F-I/O in different I-slaves may be assigned the same PROFIsafe destination address, as they are only addressed within the station, that is, by the F-CPU in the I-slave.

The following applies to Ethernet subnets and hybrid configurations of PROFIBUS and Ethernet subnets:

The PROFIsafe destination address and, thus, the address switch setting on the F-I/O have to be unique only*** within the entire Ethernet subnet, including all lower-level PROFIBUS subnets, and station-wide** (system-wide). For S7-300 F-SMs and ET 200S, ET 200eco, ET 200pro, and ET 200iSP F-modules, you can assign a maximum of 1022 different PROFIsafe destination addresses.

Exception: The F-I/O in different I-slaves/I-Devices may be assigned the same PROFIsafe destination address, as they are only addressed within the station, that is, by the F-CPU in the I-slave/I-Device.

The networked nodes of an Ethernet subnet are characterized by having IP addresses with the same subnet address, i.e., the IP addresses match in the digits that have the value "1" in the subnet mask.

Example:

IP address: 140.80.0.2.

Subnet mask: 255.255.0.0 = 11111111.11111111.00000000.00000000

Meaning: Bytes 1 and 2 of the IP address define the subnet; subnet address = 140.80.
(S002)

* A network consists of one or more subnets. "Network-wide" means beyond the boundaries of the subnet.

** "Station-wide" means for one device in the *hardware and network editor* (for example, a SIMATIC S7-300 or an I-slave)

*** Provided that no PROFINET IO communication over several Ethernet subnets is used.

Recommendation for address assignment

To allow you to easily determine whether the F-destination addresses are uniquely assigned, we recommend that you define a separate range for each F-CPU that includes the F-destination addresses of all F-I/O it reaches.

Then you only have to ensure that the ranges of the respective F-CPU's do not overlap, and the single test for uniqueness can be limited to each individual F-CPU.

The safety printout (see Printing project data (Page 187)) also indicates the ranges containing the F-destination addresses of the respective F-CPU.

Customizing the F-monitoring time for F-I/O

With the "Default F-monitoring time for central F-I/O" parameter in the properties of **the F CPU**, the monitoring of communication between the F-CPU and F-I/O is already configured.

You can customize the F-monitoring time in the properties of the F-I/O under "F-parameters". This may be necessary to prevent time monitoring functions from responding in the error-free case, if the F-I/O requires a longer F-monitoring time. For this purpose, activate the corresponding check box and assign an F-monitoring time.

 WARNING

It can only be ensured (from a fail-safe standpoint) that a signal level to be transferred will be captured on the sender side and transferred to the receiver if the signal level is pending for at least as long as the assigned monitoring time. (S018)
--

Additional information can be found in Monitoring and response times (Page 523).

Group diagnostics for fail-safe S7-300 signal modules

By disabling a channel in the properties of the module you also disable its group diagnostics for this channel.

Exception:

For the following S7-300 fail-safe signal modules

- SM 326; DI 8 x NAMUR (Order No. 6ES7326-1RF00-0AB0)
- SM 326; DO 10 x DC 24V/2A (Order No. 6ES7326-2BF01-0AB0)
- SM 336; AI 6 x 13Bit (Order No. 6ES7336-1HE00-0AB0)

the "Group diagnostics" parameter enables and disables the monitoring of channel-specific diagnostic messages of F-SMs (such as wire break and short circuit) to the F-CPU. You should disable group diagnostics for **unused** input or output channels.

 WARNING
For the following S7-300 fail-safe signal modules (F-SMs) with activated safety mode, "Group diagnostics" must be enabled for all connected channels: <ul style="list-style-type: none">• SM 326; DI 8 x NAMUR (Order No. 6ES7326-1RF00-0AB0)• SM 326; DO 10 x DC 24V/2A (Order No. 6ES7326-2BF01-0AB0)• SM 336; AI 6 x 13 Bit (Order No. 6ES7336-1HE00-0AB0) Check to verify that you have only disabled group diagnostics for these F-SMs for input and output channels that are actually unused. (S003)

Diagnostic interrupts can be enabled optionally.

Additional information

For detailed description of the **parameters**, refer to the help on the properties of the respective F-I/O and in the respective manual for the *F-I/O*.

2.5 Configuring fail-safe DP standard slaves and fail-safe standard I/O devices

Requirement

In order to use fail-safe DP standard slaves for SIMATIC Safety, these must be operated as standard slaves on PROFIBUS DP and support the PROFIsafe bus profile. Fail-safe DP standard slaves used in hybrid configurations on PROFIBUS DP and PROFINET IO based on IE/PB link must support the PROFIsafe bus profile in V2 mode.

In order to use fail-safe standard I/O devices for SIMATIC Safety, the standard devices must be on PROFINET IO and support the PROFIsafe bus profile in V2 mode.

Configuration with GSD files

As is the case in a standard system, the basis for configuring fail-safe DP standard slaves/standard I/O devices is the device specification in the GSD file (generic station description file).

A GSD file contains all of the properties of a DP standard slave or standard I/O device. For fail-safe DP standard slaves/standard I/O devices, portions of the specification are protected by a CRC.

The GSD files are supplied by the device manufacturers.

Protection of the data structure of the device in GSD files

The only GSD files supported are those that satisfy the requirements for protection defined as of *PROFIsafe Specification V2.0* using a CRC stored in this file ("desired value" for F_IO_StructureDescCRC).

While the hardware configuration is compiled, the data structure described in the GSD file is checked. If an error is detected, you should clarify whether the GSD file provided by the device manufacturer contains the desired value for F_IO_StructureDescCRC.

Procedure for configuring with GSD files

You import the GSD files in your project (see *Help on STEP 7 Professional "GSD files"*).

1. Select the fail-safe DP standard slave/standard I/O device in the "Hardware catalog" task card and connect it to the relevant subnet in the network view.
2. Select the fail-safe DP standard slave/standard I/O device and insert the necessary F-modules, if this does not occur automatically.
3. Select the relevant F-module and open the "Properties" tab in the inspector window.

Channel-granular passivation is not supported for fail-safe DP standard slaves/standard I/O devices.

Additional information

You will find the description of the parameters in the Help on fail-safe DP standard slaves and standard I/O devices.

Safety Administration Editor

Overview

The *Safety Administration Editor* supports you in the following tasks:

- Displaying of status of the safety program
- Displaying of collective F-signature
- Displaying of status of safety mode
- Creating and organizing of F-runtime groups
- Displaying information on the F-blocks
- Specifying/changing access protection
- Specifying/changing general settings for the safety program

Description	Status	Offline signature	Online signature
Collective F-signature	●	4EA3EE13	4EA3EE13

The *Safety Administration Editor* is divided into the following areas:

- **General**

Under "General", the status of safety mode, the safety program, and the collective F-signature are displayed to you. Further information on the "General" area can be obtained in ""General" tab (Page 41)".
- **F-runtime groups**

A safety program consists of one or two F-runtime groups. Under "F-runtime groups", you determine the blocks and properties of an F-runtime group.

General information on the F-runtime groups can be obtained under "Structure of the safety program (Page 56)".

Information on the generation of F-runtime groups can be obtained under "Defining F- Runtime Groups (Page 65)".
- **F-blocks**

Under "F-blocks", you will find information on the F-blocks used in your safety program and their properties . Further information on the "F-blocks" area can be obtained under ""F-blocks" tab (Page 44)".
- **Access protection**

Under "Access protection", you can set up, change, or revoke the password for the safety program. Access protection is mandatory for productive operation. Further information on access protection can be found under "Setting up, changing and revoking access permission for the safety program (Page 51)".
- **Settings**

Under "Settings", you set the parameters for the safety program. For information on the settings for your safety program, refer to ""Settings" tab (Page 45)".

Procedure for calling the Safety Administration Editor

Requirement

The *Safety Administration Editor* is visible as a line in the project tree, if you have configured a CPU as an F-CPU in the project, i.e., the "F-capability activated" option must be selected (in the properties of the F-CPU).

To call the *Safety Administration Editor*, proceed as follows:

1. Open the folder for your F-CPU in the project tree.
2. Double-click on "Safety administration" or right-click and select the corresponding context menu for the *Safety Administration Editor*.

Result

The *Safety Administration Editor* for your F-CPU is opened in the work area.

3.1 "General" tab

"Safety mode status"

The "Safety mode status" shows the current status of safety mode. The prerequisite is an existing online connection to the selected F-CPU.

The following statuses are possible:

- "Safety mode is activated"
Safety mode for the selected F-CPU is activated.
- "The safety mode is not activated"
Safety mode for the selected F-CPU is deactivated.
- "F-CPU is in STOP"
The selected F-CPU is in STOP mode.
- "Unknown"
The safety mode status for the selected F-CPU could not be determined.

Possible reasons:

- The online connection to the selected F-CPU could not be established or was interrupted.
- The safety program was not called.
- "F-runtime group was not called"
The main safety block of at least one F-runtime group was not called.
- "(No online connection)"
An online connection to the selected F-CPU does not exist.

The display of the status is updated whenever the status of the selected F-CPU changes or you click on the "General" tab in online mode once again.

"Disable safety mode"

For existing online connection and active safety mode operation, you have the option of using the "Disable safety mode" button to disable safety mode for the selected F-CPU. Safety mode can be deactivated only for the entire safety program and not for individual F-runtime groups.

Proceed as follows:

1. Click the "Disable safety mode" button.
2. Enter the password for the safety program in the dialog window and confirm with "OK".
3. A corresponding dialog will display the collective F-signature.
4. Using the displayed collective F-signature, verify that you have selected the desired F-CPU.
5. If you have selected the correct F-CPU, confirm the dialog with "Yes".

Result:Safety mode for the selected F-CPU is deactivated.

You have the option to prevent the deactivation of safety mode. To do so, clear "Safety mode can be disabled" in the "Settings" area of the *Safety Administration Editor*.

"Safety program status"

"Safety program status" displays the current status of your online and offline program.

The following statuses are possible:

- Consistent
- Inconsistent
- Modified

If no connection to the online program could be established, the message "(no online connection) " will be shown.

"Program signature"

For a non-existing online connection

"Program signature" displays the offline collective F-signature and the "Timestamp" displays the time of the last compilation process.

For an existing online connection

For an existing online connection, "Program signature" displays the status of the safety program and the online and offline collective F-signatures.

The meaning of the symbols in the "Status" column are described in the following table.

Status	Meaning
	The online and offline collective F-signatures match, and a password was assigned for the online and offline safety programs.
	The online and offline collective F-signatures do <i>not</i> match or no password was assigned for a safety program.
—	The safety program status could not be determined.

See also

Deactivating Safety Mode (Page 190)

3.2 "F-blocks" tab

Overview

The "F-Blocks" area helps you in the following tasks:

- Displaying the F-blocks used in your safety programs.
- Displaying the F-blocks used in the F-runtime groups.
- Displaying additional information about the F-blocks.

Displayed information

The following information is displayed for F-blocks in offline mode:

- Has the F-block been compiled and used?
- Function of F-block in the safety program
- Offline signature
- Time stamp of the last change

The following information is displayed for F-blocks in online mode:

- Status (whether block has the same time stamp online and offline)
- Function of F-block in the safety program
- Offline signature
- Online signature

The F-blocks are hierarchically displayed as in the "Program blocks" folder.

The description of the symbols in the "Status" column can be found in "Comparing Safety Programs (Page 183)".

Note

During the offline-online comparison, the comparison statuses may occasionally differ between the *comparison editor* and status display in the *Safety Administration Editor*. The decisive status is the result of the comparison in the *comparison editor*, since this is the only comparison that takes into account the contents of the F-blocks.

Filter function



Using the filter function, you can select whether you want to view all F-blocks of a certain F-runtime group or the entire safety program.

- Select "All F-blocks " from the drop-down list to view all F-blocks.
- Select an F-runtime group from the drop-down list to see all F-blocks of this F-runtime group.

3.3 "Settings" tab

"Number range of the F-system blocks"

The number ranges assigned here are used by the F-System for new, automatically generated F-blocks. When creating new F-system block, the F-system selects the next available number from the number ranges.

At this point, you can select whether the number ranges are managed by the F-system or if a fixed range specified by you is used.

- "System managed"

The number ranges are managed automatically by the F-system, depending on the F-CPU used. The F-system selects an available number range. The start and end ranges of the number ranges are displayed.

By default, the entire range of numbers that is provided by the F-CPU is used.

- "Fixed range"

You can select the start and end ranges of the number ranges from the available range. The available range depends on the F-CPU used.

An invalid number range selection is indicated by an error message.

Changes will become valid only during the next compilation. After modification of the number range, numbers previously assigned for F-system blocks are moved to the new number range, if required. The F-I/O DBs are not affected by this.

"Versions used in safety program"

Select the version to be used in your safety program for:

- Automatically generated F-blocks
- F-system blocks
- F-I/O access

"Local data used in safety program"

You use this parameter to specify the amount of temporary local data (in bytes) that is available for the call hierarchy below the main safety block.

The setting applies to each F-runtime group of a safety program. Additional information on F-runtime groups can be found in Structure of the safety program (Page 56).

The **minimum quantity setting** is determined by the local data requirement of the F-blocks generated automatically when the safety program is compiled.

For this reason, you must provide at least 440 bytes. However, the local data requirement for the automatically added F-blocks may be higher depending on the local data requirement of the F-blocks you created with FBD or LAD.

Therefore, provide as much local data as possible. If there is not enough local data available for the automatically added F-blocks (440 bytes or more), the safety program will be compiled nevertheless.

Data in automatically added F-DBs are then used instead of local data. However, this increases the runtime of the F-runtime group(s). You will receive a notice when the automatically added F-blocks require more local data than configured.

WARNING

The calculated maximum runtime of the F-runtime group using the Excel file for response time calculation (<http://support.automation.siemens.com/WW/view/en/49368678/133100>) is no longer correct in this case because the calculation assumes sufficient availability of F-local data.

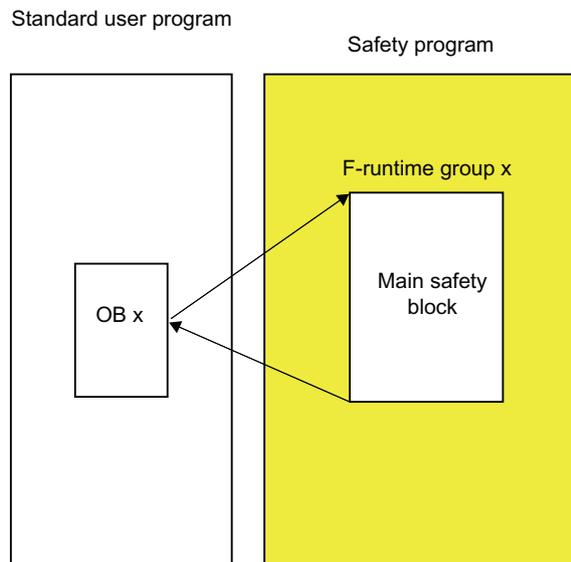
In this case, use the value you configured for the maximum cycle time of the F-runtime group (F-monitoring time) as the maximum runtime of the F-runtime group when calculating the maximum response times in the event of an error and for any runtimes of the standard system using the above-mentioned Excel file. (S004)

The maximum quantity setting depends on:

- Local data requirement of the main safety block and the higher-level standard user program. For this reason, you should call the main safety blocks directly in OBs (cyclic interrupt OBs, whenever possible), and additional local data should not be declared in these cyclic interrupt OBs.
- Maximum volume of local data of the utilized F-CPU (see Technical Specifications in the product information for the utilized F-CPU). For S7-400 F-CPU, you can configure the local data for each priority class. Therefore, assign the largest possible local data volume for the priority classes in which the safety program (the main safety blocks) will be called (e.g., OB 35).

Maximum possible amount of local data as a function of local data requirement of main safety block and higher-level standard user program

Case 1: Main safety block called directly from OBs

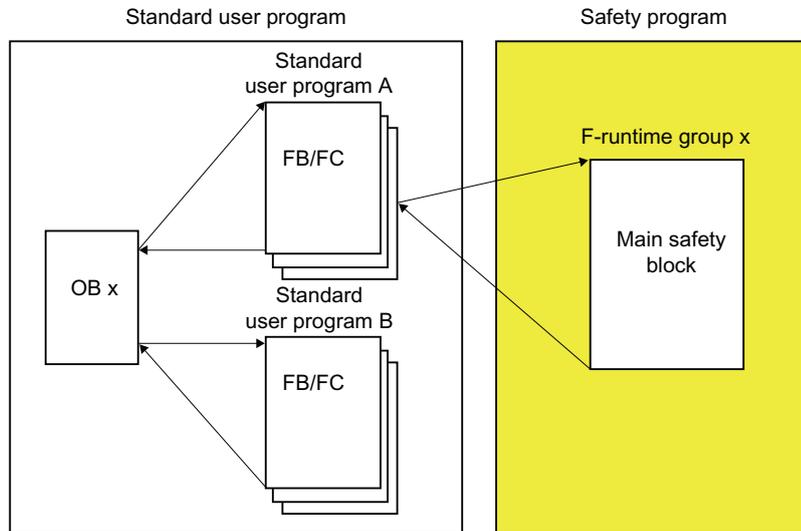


Set the "Local data used in safety program" parameter to the maximum amount of local data of the utilized F-CPU, minus the local data requirement of the main safety block (if the main safety block has 2 F-runtime groups, use the largest local data requirement), and minus the local data requirement of the calling OBx (if there are 2 F-runtime groups, use the OB with the larger local data requirement).

Note: If you have not declared any temporary local data in the main safety blocks and calling OBx, the local data requirement of the main safety blocks is 6 bytes and the local data requirement of the calling OBx is 26 bytes. You can derive the local data requirement of the main safety blocks and calling OBx from the program structure.

Select the utilized F-CPU in the project tree and then "Tools > Call structure". The table gives the local data requirement in the path or for the individual blocks (see also the help on *STEP 7 Professional*).

Case 2: Main safety block not called directly from OBs



Set the "Local data used in safety program" parameter to the value calculated for Case 1, minus the local data requirement of the standard user program A (if the standard user program A has 2 F-runtime groups, use the largest local data requirement).

Note: You can derive the local data requirement of the standard user program A from the program structure.

Select the utilized F-CPU in the project tree and then "Tools > Call structure". The table gives the local data requirement in the path or for the individual blocks (see also the help on *STEP 7 Professional*).

"Advanced settings"

If you deselect the "Safety mode can be disabled" option, you can prevent the deactivation of safety mode for a safety program.

When you change the setting for this option, you must recompile the safety program and download it to the F-CPU.

Access protection

Access protection is necessary for productive operation

For access to the SIMATIC Safety F-system, access protection is mandatory for productive operation.

For test purposes, commissioning, etc. no access protection is necessary at first. This means that you can execute all offline and online actions without access protection, i.e., without password prompt.

WARNING

Access to the SIMATIC Safety F-system without access protection is intended for test purposes, commissioning, etc., when the plant is not in productive operation mode. You must guarantee the safety of the plant through other organizational measures, for example, restricted access to certain areas.

Before you transition into productive operation mode, you must have set up and activated access protection. (S005)

4.1 Overview of Access Protection

Introduction

You can protect access to the SIMATIC Safety F-system by two password prompts: one for the safety program and another for the F-CPU.

Password for the safety program

The password for the safety program is available in two forms:

- The offline password is part of the safety program in the offline project on the programming device or PC.
- The online password is part of the safety program in the F-CPU.

Password for the F-CPU

The access protection is set at the F-CPU level. The online password is used to identify the F-CPU.

Overview of password assignment and prompt

The following table provides an overview of the access permissions for the F-CPU and the safety program.

The sections below show you how to assign the passwords and how to set up, change, and cancel access permissions for the F-CPU and the safety program.

	Password for F-CPU	Password for safety program
Assignment	In the <i>hardware and network editor</i> , during configuration of the F-CPU, inspector window, in "Settings" tab below "Protection", corresponding safety level, e.g., "Write protection for fail-safe blocks"	In the <i>Safety Administration Editor</i> under "Access protection"
Prompt	<p>If you do not have access permission for the safety program (Page 51):</p> <ul style="list-style-type: none"> • When the safety program is downloaded in its entirety • When F-blocks with F-attribute are downloaded and deleted 	<p>If you have assigned a password and this has not yet been entered since the project was opened, or you do not have access permission for the safety program (Page 51):</p> <p>Offline password e.g.:</p> <ul style="list-style-type: none"> • When the password is changed • When changing and deleting F-runtime groups • When inserting/moving/rename/deleting F-blocks in the project tree <p>Online password e.g., when deactivating safety mode (the password must always be entered, even if access permission for the safety program is still valid)</p>
Validity	Once the correct password has been entered, access is authorized until the project is closed in <i>STEP 7 Professional</i> or until access permission is canceled with "Online > Delete access rights".	Once the correct password has been entered, access is authorized until the project is closed in <i>STEP 7 Professional</i> or until access permission is canceled in the <i>Safety Administration Editor</i> .

4.2 Setting up, changing and revoking access permission for the safety program

Procedure for assigning the password for the safety program

Use the following procedure to assign the password for the safety program:

1. Open the folder for your F-CPU in the project tree.
2. Select "Safety Administration" and select "Go to protection" in the context menu.
Alternatively, double-click on "Safety Administration". The *Safety Administration Editor* of the F-CPU will open.
3. Select "Access Protection" in the area navigation.
4. Under "Offline safety program protection", click the "Define" button and enter the password for the safety program in the following dialog in the "New password" and "Confirm password" fields.
5. Confirm the assigned password with "OK".

Note

You cannot define the online password separately; the assigned offline password is applied. The online password is used to identify the F-CPU. Following a change to the offline password, the online and offline passwords may differ until the next time the offline safety program is loaded to the F-CPU.

Note

Use different passwords for the F-CPU and the safety program to optimize access protection.

NOTICE

If access protection is not used to limit access to the programming device or PC to only those persons who are authorized to modify the safety program, the following organizational measures must be taken to ensure the effectiveness of password protection at the programming device or PC:

- Only authorized personnel may have access to the password.
- Authorized personnel must explicitly cancel the access permission for the safety program before leaving the programming device or PC. If this is not strictly implemented, a screen saver equipped with a password accessible only to authorized personnel must also be used.

Changing the password for the safety program

You may change the safety program password as long as you have the necessary access permissions. This is also carried out in the "Access protection" area by clicking the "Change" button and following the same steps as in Windows, i.e., enter the old password, and then enter the new password twice).

Revoking the password for the safety program

You can revoke the password for a safety program, i.e., the safety program is then no longer protected by a password.

Proceed as follows:

1. Open the folder for your F-CPU in the project tree.
2. Select "Safety Administration" and select "Go to protection" in the context menu.

Alternatively, double-click on "Safety Administration". The *Safety Administration Editor* of the F-CPU will open.

3. Select "Access Protection" in the area navigation.
4. Click the "Change" button.
5. Under "Old password", enter the password for the safety program.
6. Click "Revoke" and then on "OK".

Logging in to the safety program

Log in to the safety program as follows:

1. Open the folder for your F-CPU in the project tree.
2. Select "Safety Administration" and select "Go to protection" in the context menu.

Alternatively, double-click on "Safety Administration". The *Safety Administration Editor* of the F-CPU will open.

3. Select "Access Protection" in the area navigation.
4. Enter the password for the safety program in the "Password" input field.
5. Select the "Login" button.

Validity of access permission for a safety program

If access permission for a safety program was granted through the entry of the password, this remains until the project is closed. If *STEP 7 Professional* is closed, then a project that is still open is automatically closed and a granted access permission is reset.

Canceling access permission for the safety program

The access permission for the safety program can be canceled as follows:

- By clicking the "Log off" button in the "Access protection" area in the "*Safety Administration Editor*".
- In the context menu for the *Safety Administration Editor* context menu (access by right-clicking).
- By using the lock symbol in the line of the *Safety Administration Editor*.

The user will then be prompted to enter the password for the safety program again the next time an action requiring a password is performed. To "cancel" access permission when using the online Modify function, the connection to the F-CPU must first be terminated (e.g., via the "Online > Go offline" menu command or the corresponding button in the toolbar).

The access permission for the safety program is reset automatically, if the project or *STEP 7 Professional* was closed.

Displaying the validity of access permission

The validity of the access permission is displayed in the project tree as follows:

- The access permission is valid, if the lock symbol in the line of the *Safety Administration Editor* is shown unlocked.
- The access permission is not available, if the lock symbol shows a closed lock.
- If no lock symbol is shown, no password was assigned.

4.3 Setting up access permission for the F-CPU

Receiving access permission for the F-CPU

You receive access permission for the F-CPU - depending on the configured protection level - by entering the password for the F-CPU prior to performing an action requiring a password.

Procedure for assigning a password for the F-CPU

You assign the password for the F-CPU when configuring the F-CPU (see Configuring the F-CPU (Page 29)).

You arrive there directly, if you click the link "Go to the "Protection" area of the F-CPU" in the "Access protection" area in the *Safety Administration Editor*.

Changing the password for the F-CPU

For the new password to become valid after a password change for the F-CPU, you must download the changed configuration into the F-CPU. If necessary, you must enter the "old" password for the F-CPU for this load operation. The F-CPU must be in STOP mode.

Validity/cancelation of the access permission for the F-CPU

Access permission for the F-CPU remains valid until the project is closed in *STEP 7 Professional* or canceled using the "Online > Delete access rights" menu command.

WARNING

If access protection is not used to limit access to the programming device or PC to only those persons who are authorized to modify the safety program, the following organizational measures must be taken to ensure the effectiveness of the password protection for the F-CPU at the programming device or PC:

- Only authorized personnel may have access to the password.
- Authorized personnel must explicitly cancel the access permission for the F-CPU before leaving the programming device or PC by closing *STEP 7 Professional* or via the "Online > Delete access rights" menu. If this is not strictly implemented, a screen saver equipped with a password accessible only to authorized personnel must also be used. (S006)

Programming

5.1 Overview of Programming

Introduction

A safety program consists of F-blocks that you create using the FBD or LAD programming language and F-blocks that are automatically added. Fault control measures are automatically added to the safety program you create, and additional safety-related tests are performed. Moreover, you have the option to incorporate special ready-made safety functions in the form of instructions into your safety program.

An overview of the following is given below:

- The structure of the safety program
- The fail-safe blocks
- Differences in the programming of the safety program with FBD/LAD compared to programming of standard user programs

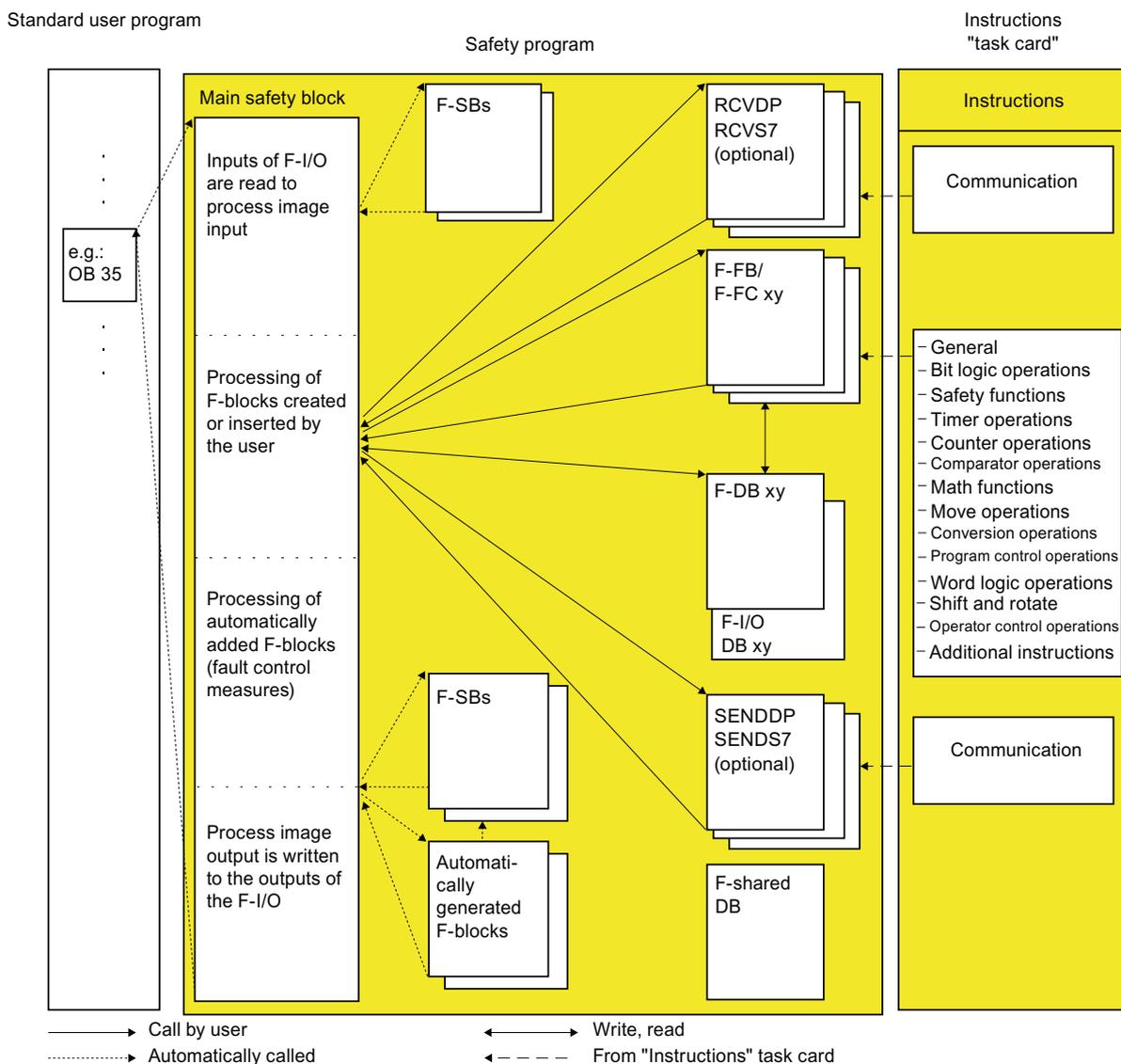
5.1.1 Structure of the safety program

Representation of program structure

The figure below shows the schematic structure of a safety program or an F-runtime group. For structuring purposes, a safety program consists of one or two F-runtime groups.

Each F-runtime group contains:

- F-blocks that are created by you using FBD or LAD or are inserted from the project library or global libraries
- F-blocks that are added automatically (F-system blocks (F-SBs), automatically generated F-blocks, F-shared DB, and F-I/O DBs)



Main safety block

The main safety block is the first F-block of the safety program that you program yourself. During compiling, it is supplemented by additional F-system blocks (F-SBs) and automatically generated F-blocks.

You must assign the main safety block to an F-runtime group (Page 65).

F-runtime groups

To improve handling, a safety program consists of one or two "F-runtime groups". An F-runtime group is a logical construct of several related F-blocks that is formed internally by the F-system.

An F-runtime group consists of the following:

- a main safety block (an F-FB/F-FC that you assign to the OB)
- Possible additional F-FBs or F-FCs that you program using F-FBD or F-LAD, as needed and call from the main safety block
- One or more F-DBs, as needed
- F-I/O DBs
- F-blocks from the project library or shared libraries
- F-system blocks F-SBs
- Automatically generated F-blocks

Structuring of the safety program in two F-runtime groups

You can divide your safety program into two F-runtime groups. By arranging for portions of the safety program (one F-runtime group) to run in a faster priority class, you achieve faster safety circuits with short response times.

5.1.2 Fail-Safe Blocks

F-blocks of an F-runtime group

The following table shows the F-blocks that you use in an F-runtime group:

F-block	Function
Main safety block	The first step in programming of the safety program is the main safety block. The main safety block is an F-FC or F-FB (with instance DB), which, by being called by a standard block (recommendation: OB 35), becomes the main safety block.
F-FB/F-FC	Both in the main safety block as well as additional F-FBs and F-FCs, you can perform the following: <ul style="list-style-type: none"> • Program the safety program with the instructions available for F-blocks in FBD or LAD • Call other created F-FBs/F-FCs for structuring the safety program • Insert F-blocks from the project library or shared libraries
F-DB	Optional fail-safe data blocks that can be read- and write-accessed within the entire safety program.
F-I/O DB	An F-I/O DB is automatically generated for each F-I/O when it is configured. You can or you must access the tags of the F-I/O DB in conjunction with F-I/O accesses.

Instructions for the safety program

In the "Instructions" task card, you find all instructions that you can use for programming the safety program.

You will find instructions that you know from the standard user program, such as bit logic operations, mathematical functions, functions for program control, and word logic operations.

Moreover, there are instructions with safety functions, e.g., for two-hand monitoring, discrepancy analysis, muting, emergency STOP, safety door monitoring, and feedback loop monitoring.

Additional information

For a detailed description of the instructions for the safety program, refer to Overview of instructions (Page 219).

5.1.3 Restrictions in the programming languages FBD/LAD

LAD and FBD programming languages

The user program in the F-CPU typically consists of a standard user program and a safety program. The standard user program is created using standard programming languages such as STL, LAD, or FBD.

For the safety program, LAD or FBD may be used with certain restrictions in the instructions and the applicable data types and operand areas.

Supported instructions

You will find the supported instructions in the description of the instructions (starting from Overview of instructions (Page 219)).

Note

Preconnection of enable input EN or evaluation of enable output ENO is not possible.

Supported data types and parameter types

Only the following elementary data types are supported:

- BOOL
- INT
- WORD
- DINT
- DWORD
- TIME

Non-permitted data and parameter types

The following types are **not** permitted:

- All types not listed in "Supported data types and parameters types". (e.g., BYTE, REAL)
- Complex data types (e.g., STRING, ARRAY, STRUCT, UDT)
- Parameter types (e.g., BLOCK_FB, BLOCK_DB, ANY)

Supported operand areas

The system memory of an F-CPU is divided into the same operand areas as the system memory of a standard CPU. You can access the operand areas listed in the table below from within the safety program.

Note that you can access data in the safety program only via the declared data type for the data element/tag as follows:

- Data of data type BOOL only bit by bit
- Data of data type INT only word by word
- Data of data type WORD only word by word
- Data of data type DINT only double word by double word
- Data of data type DWORD only double word by double word
- Data of data type TIME only double word by double word

Example: You can access input channels of data type BOOL in the process image input of F-I/O using the "input (bit)" unit only.

For reasons of clarity, you should always access operand areas in a safety program using symbolic names.

Table 5- 1 Supported operand areas

Operand area	Access via units of the following size:	S7 Notation	Description
Process image input			
<ul style="list-style-type: none"> • Of F-I/O 	Input (bit) Input word Input double-word	I IW ID	At the beginning of each main safety block, the F-CPU reads the inputs from the F-I/O and saves the values to the process image input. Input channels can only be accessed read-only. Therefore, a transfer to IN_OUT parameters of an F-FB or F-FC is also not permitted.
<ul style="list-style-type: none"> • Of standard I/O 	Input (bit) Input word Input double-word	I IW ID	At the beginning of each OB 1 cycle, the F-CPU reads the inputs from the standard I/O and saves the values to the process image input. Also bear in mind the update times when using process image partitions. Input channels of standard I/O can only be accessed read-only. Therefore, a transfer to IN_OUT parameters of an F-FB or F-FC is also not permitted. In addition, a process-specific plausibility check is required.

Operand area	Access via units of the following size:	S7 Notation	Description
Process image output			
<ul style="list-style-type: none"> Of F-I/O 	Output (bit) Output word Output double-word	Q QW QD	<p>In the safety program, the values for the outputs of the F-I/O are calculated and stored in the process image output. At the end of the main safety block, the F-CPU writes the calculated output values to the outputs of the F-I/O. Output channels can only be accessed write-only.</p> <p>Therefore, a transfer to IN_OUT parameters of an F-FB or F-FC is also not permitted.</p>
<ul style="list-style-type: none"> Of standard I/O 	Output (bit) Output word Output double-word	Q QW QD	<p>In the safety program, the values for the outputs of the standard I/O are also calculated, if necessary, and stored in the process image output. At the beginning of the next OB 1 cycle, the F-CPU writes the calculated output values to the outputs of the standard I/O. Also bear in mind the update times when using process image partitions.</p> <p>Output channels of standard I/O are write-only channels.</p> <p>Therefore, a transfer to IN_OUT parameters of an F-FB or F-FC is also not permitted.</p>
Bit memory	Bit memory (bit) Memory word Memory double word	M MW MD	<p>This area is used for data exchange with the standard user program.</p> <p>In addition, read access requires a process-specific plausibility check.</p> <p>A particular element of the bit memory can be either read- or write-accessed in the safety program.</p> <p>Therefore, a transfer to IN_OUT parameters of an F-FB or F-FC is also not permitted.</p> <p>Note that bit memory can only be used for connecting the standard user program and the safety program; it must not be used as a buffer for F-data.</p>

Operand area	Access via units of the following size:	S7 Notation	Description
Data blocks			
<ul style="list-style-type: none"> F-DB 	Data bit Data word Data double word	DBX DBW DBD	Data blocks store information for the program. They can either be defined as global data blocks such that all F-FBs, F-FCs, or main safety blocks can access them or assigned to a particular F-FB or main safety block (instance DB). In doing so, a tag of a global DB can be accessed only from one F-runtime group and an instance DB can be accessed only from the F-runtime group in which the associated F-FB/instruction is called.
<ul style="list-style-type: none"> DB 	Data bit Data word Data double word	DBX DBW DBD	This area is used for data exchange with the standard user program. In addition, read access requires a process-specific plausibility check. For a data element of a DB, either read access or write access is possible in the safety program. Therefore, a transfer to IN_OUT parameters of an F-FB or F-FC is also not permitted. Note that data of a DB can only be used for transferring data between the standard user program and the safety program; DBs must not be used as a buffer for F-data.
Temporary local data	Local data bit Local data word Local data double word	L LW LD	This memory area holds the temporary data of a block (or F-block) while this block is being executed. The local data stack also provides memory for transferring block parameters and for saving intermediate results.

Non-permitted operand areas

Access via units other than those listed in the table above is **not** permitted. The same applies to access to operand areas not listed, in particular:

- Data blocks that were automatically added
Exception: certain data in the F-I/O DB (Page 82) and the F-shared DB (Page 107) of the safety program
- I/O area: Inputs
- I/O area: Outputs

Boolean constants "0" and "1"

If you require Boolean constants "0" (FALSE) and "1" (TRUE) in your safety program to assign parameters during block calls, you can access the "VKE0" and "VKE1" tags in the F-shared DB using fully qualified DB access ("F_GLOBDB".VKE0 and "F_GLOBDB".VKE1, respectively).

Operand area of temporary local data: Particularities

Note

Note when using the operand area of temporary local data that the first access of a local data element in a main safety block/F-FB/F-FC must always be a write access. This initializes the local data element.

Make sure that the initialization of the local data element is **not** skipped over by JMP, JMPN, or RET instructions (branching).

The "local data bit" should be initialized with the Assign ("=") (FBD) or ("-()") (LAD) instruction. Assign the local data bit a signal state of "0" or "1" as a Boolean constant.

Local data bits cannot be initialized with the Flip Flop (SR, RS), Set Output (S) or Reset Output (R) instructions.

The F-CPU can go to STOP if this is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"
- "Safety program: internal CPU fault; internal error information: 404"

"Fully qualified DB access"

The initial access to data of a data block in an F-FB/F-FC **must** always be a "fully qualified DB access," or it must be preceded by the "OPN" instruction. This also applies to the initial access to data of a data block after a jump label.

Example of "fully qualified DB access":

Assign a name for the F-DB, e.g., "F_Data_1". Use the names assigned in the declaration of the F-DB instead of the absolute addresses.

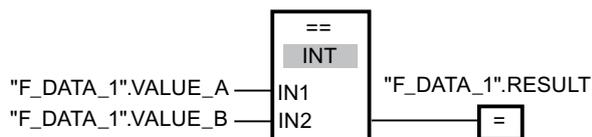


Figure 5-1 Example with fully-qualified access and symbolic names

Example of "non-fully-qualified DB access":

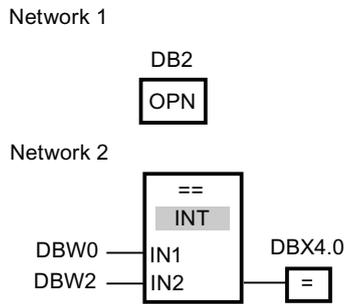


Figure 5-2 Example without fully-qualified access and symbolic names

Access to instance DBs

You can also access instance DBs of F-FBs with fully qualified access, e.g., for transfer of block parameters. It is not possible to access static data in instance DBs of other F-FBs.

Note that accessing instance DBs of F-FBs that are not called in the safety program can cause the F-CPU to go to STOP mode.

5.2 Defining F-Runtime Groups

5.2.1 Rules for F-Runtime Groups of the Safety Program

Rules

Note the following:

- The channels of an F-I/O can only be accessed from one F-runtime group.
- Tags of the F-I/O DB of an F-I/O can only be accessed from one F-runtime group and only from the F-runtime group from which the channels of this F-I/O are accessed (if access is made).
- F-FBs can be used in more than one F-runtime group but they must be called with different instance DBs.
- Instance DBs can only be accessed from the F-runtime group in which the associated F-FB is called.
- Individual tags of F-DBs (except the F-shared DB) can only be used in one F-runtime group (however, an F-DB can be used in more than one F-runtime group).
- A DB for F-runtime group communication can be read and write accessed by the F-runtime group to which it was assigned as "DB for runtime group communication", but only read-accessed by the "receiver" F-runtime group.
- The F-communication DB can only be accessed from one F-runtime group.
- F-blocks must not be called directly in an OB; rather, they must be inserted into a F-runtime group.
- For optimal use of temporary local data, you must call the F-runtime group (the main safety block) directly in an OB (cyclic interrupt OB, if possible); you should not declare any additional temporary local data in this cyclic interrupt OB.
- Within a cyclic interrupt OB, the F-runtime group should be executed **before** the standard user program; that is, it should be called at the very beginning of the OB, so that the F-runtime group is always called at fixed time intervals, regardless of how long it takes to process the standard user program.

For this reason, the cyclic interrupt OB should also not be interrupted by higher priority interrupts.

- A main safety block may only be called once. In addition, you must not call it with different instance DBs. Multiple calls are not permitted and can cause the F-CPU to go to STOP mode.
- The process image input and output of standard I/O, bit memory, and data of DBs of the standard user program may be accessed from more than one F-runtime group. (See also Data exchange between standard user program and safety program (Page 105))
- F-FCs can generally be called in more than one F-runtime group.

Note

You can improve performance by writing sections of the program that are not required for the safety function in the standard user program.

When determining which elements to include in the standard user program and which to include in the safety program, you should keep in mind that the standard user program can be modified and downloaded to the F-CPU more easily. In general, changes in the standard user program do not require an acceptance test.

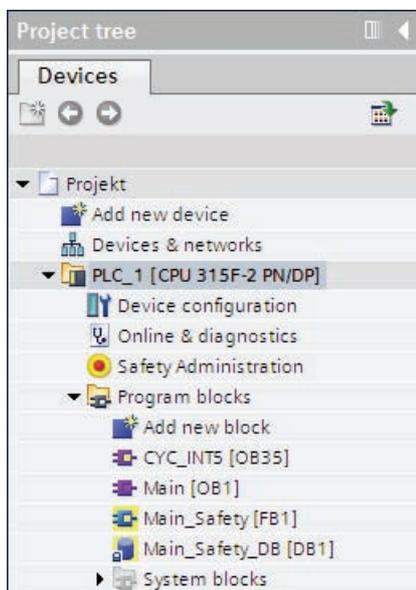
5.2.2 Procedure for Defining an F-Runtime Group

Requirements

In your project you have selected an F-CPU from the "Hardware catalog" task card and inserted it into the *hardware and network editor*. In the "Properties" tab of the F-CPU, the "F-capability activated" check box is selected (default setting).

F-runtime group created by default

STEP 7 Safety Advanced V11 inserts F-blocks for an F-runtime group in the project tree by default. You see the (F-)blocks of the F-runtime group (CYC_INT5 [OB 35], Main-Safety [FB 1], and Main-Safety_DB [DB1]) in the project tree when you open the "Program blocks" folder.



Procedure for defining an F-runtime group

To define an F-runtime group, proceed as follows:

1. Open the *Safety Administration Editor* by double-clicking in the project tree.
2. Select "F-runtime groups" in the area navigation.

Result: The work area for defining an F-runtime group with the (default) settings for F-runtime group 1 is opened.

The screenshot shows the 'Add F-runtime group' dialog box. At the top, there is a title bar 'Add F-runtime group' and a descriptive text: 'A F-runtime group consists of a block (cyclic interrupt OB (OB3x), FB or FC) that calls a main safety block (FB or FC). Additional user safety functions must then be called from this main safety block. [More...](#)'. Below this is a button 'Add new F-runtime group'. The main area is titled 'F-runtime group 1 [RTG1]'. It contains three dropdown menus: 'Calling block' (CYC_INT5 [OB35]), 'Main safety block' (Main_Safety [FB1]), and 'I-DB for main safety block' (Main_Safety_DB [DB1]). An arrow labeled 'Calls' points from the 'Calling block' to the 'Main safety block'. Below these are the 'F-runtime group parameters': 'Execution time of the calling block CYC_INT5 [OB35]: 100 ms', 'Maximum cycle time of the F-runtime group 1 [RTG1]: 200 ms', and 'DB for F-runtime group communication: (none)'. At the bottom left is a button 'Delete F-runtime group'.

3. Specify the block in which the main safety block is to be called.

Cyclic interrupt OB 35 is suggested here by default. The advantage of using cyclic interrupt OBs is that they interrupt the cyclic program execution in OB 1 of the standard user program at fixed time intervals; that is, the safety program is called and executed at fixed time intervals in a cyclic interrupt OB.

In this input field, you can select only those blocks that were created in the LAD, FBD, or STL programming language. If you select a block here, the call is inserted automatically into the selected block and, if necessary, removed from a previously selected block.

If you want to call the main safety block in a block that was created in another programming language, you must program this call yourself. The input field is then not editable (grayed out), and you can change the call only in the calling block and not the Safety Administration Editor.

4. Assign the desired main safety block to the F-runtime group. If the main safety block is an FB, you must also assign an instance DB.

Main-Safety [FB1] and Main-Safety_DB [DB1] are suggested by default.

5. The F-CPU monitors the F-cycle time in the F-runtime group. For "Maximum cycle time of F-runtime group", enter the maximum permitted time between two calls of this F-runtime group (maximum of 1000 ms).

 **WARNING**

The F-runtime group call interval is monitored for the maximum value; that is, monitoring is performed to determine whether the call is executed often enough, but not whether it is executed too often. For this reason, fail-safe timers must be implemented using the TP, TON, or TOF instructions (Page 453) from the "Instructions" task card and not using counters (OB calls). (S007)

6. If one F-runtime group is to provide data for evaluation to another F-runtime group of the safety program, assign a DB for F-runtime group communication. Select an F-DB for "DB for F-runtime group communication". (See also Safety-Related Communication between F-Runtime Groups of a Safety Program (Page 69))
7. **Create an additional F-runtime group**, by clicking the "Add new F-runtime group" button.
8. Assign an F-FB or F-FC as the main safety block to a calling block. This F-FB or F-FC is automatically generated in the project tree, if not already present.
9. If the main safety block is an F-FB, then assign an instance DB to the main safety block. The instance DB is generated automatically in the project tree.
10. Execute the above-mentioned steps 3 and 4 to complete generation of the second F-runtime group.

5.2.3 Safety-Related Communication between F-Runtime Groups of a Safety Program

Safety-related communication between F-runtime groups

Safety-related communication can take place between the two F-runtime groups of a safety program. That is, fail-safe data can be provided by one F-runtime group in an F-DB and read in another F-runtime group.

Note

A DB for F-runtime group communication can be read and write accessed by the F-runtime group to which it was assigned as "DB for runtime group communication", while it can only be read-accessed by the "receiver" F-runtime group.

Tip: You can improve performance by structuring your safety program in such a way that as few data as possible are exchanged between the F-runtime groups.

Procedure for defining a DB for F-runtime group communication

You define the DB for F-runtime group communication in the work area "F-runtime groups". Proceed as follows:

1. Click "F-runtime groups" in "*Safety Administration Editor*".
2. Select an existing F-DB in the "DB for F-runtime group communication" field or assign a new one.
3. Assign a symbolic name for the F-DB.

Up-to-dateness of data when reading from another F-runtime group

Note

The data read from another F-runtime group are as up-to-date as they were when the F-runtime group providing the data finished processing before the start of the F-runtime group reading the data.

If the furnished data undergo multiple changes while the F-runtime group furnishing the data is being processed, the F-runtime group reading the data always receives the last change.

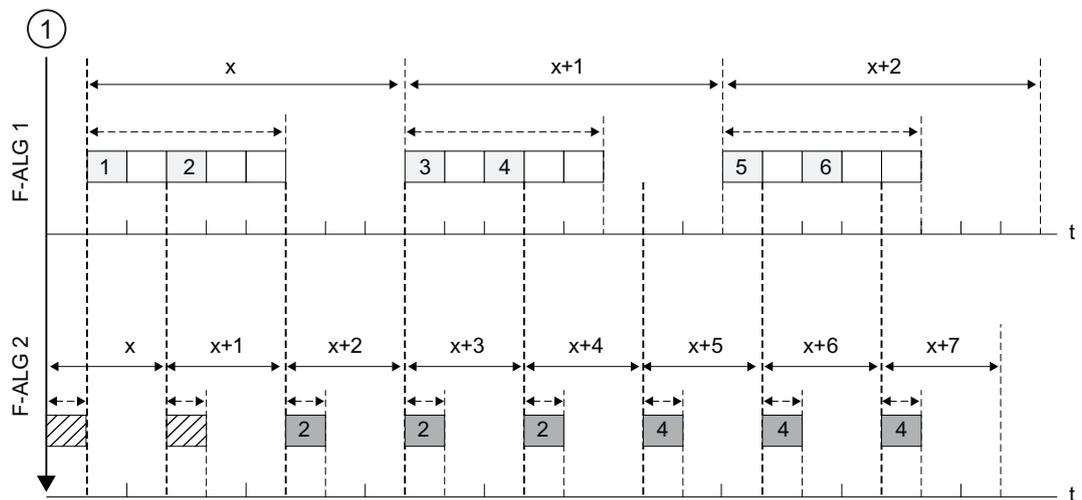
Assignment of fail-safe values

After a startup of the F-system, fail-safe values are made available to the receiving F-runtime group (for example, F-runtime group 2) having read access to data in the DB for F-runtime group communication of another F-runtime group . These are the values you specified as initial values in the DB for F-runtime group communication of F-runtime group 1.

F-runtime group 2 reads the fail-safe values the first time it is called. The second time F-runtime group 2 is called, it reads the latest data if F-runtime group 1 has been processed completely between the two calls of F-runtime group 2. If F-runtime group 1 has not been processed completely, F-runtime group 2 continues to read the fail-safe values until F-runtime group 1 is completely processed.

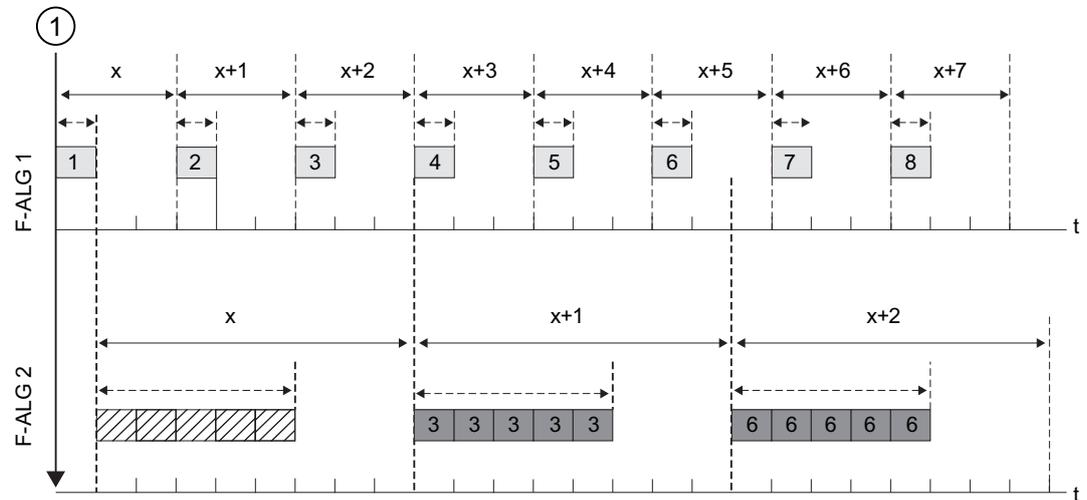
The behavior is illustrated in the two figures below.

Reading data from F-runtime group 1 that has a longer OB cycle and lower priority than F-runtime group 2



- ① Startup of F-system
- ↔ Cycle time of the OB in which the F-runtime group is called.
- ⋯ Runtime of the F-runtime group
- 1 ... Data element of F-runtime group 1, written to DB for F-runtime group communication of F-runtime group 1
- 2 data element of F-runtime group 2, read in DB for F-runtime group communication of F-runtime group 1
- ▨ Initial value in the DB for F-runtime group communication

Reading of data from F-runtime group 1 that has a shorter OB cycle and higher priority than F-runtime group 2



- ① Startup of F-system
- ↔ Cycle time of the OB in which the F-runtime group is called.
- ⋯ Runtime of the F-runtime group
- 1 ... Data element of F-runtime group 1, written to DB for F-runtime group communication of F-runtime group 1
- 2 ... Data element of F-runtime group 2, read in DB for F-runtime group communication of F-runtime group 1
- ▨ Initial value in the DB for F-runtime group communication

F-runtime group providing the data is not processed

Note

If the F-runtime group whose DB for F-runtime group communication supplies the data to be read is not processed (main safety block of the F-runtime group is not called), the F-CPU goes to STOP mode. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- Error in safety program: cycle time exceeded
- Number of the relevant main safety block (of F-runtime group that is not processed)
- Current cycle time in ms: "0"

5.2.4 Deleting an F-runtime group

Deleting an F-runtime group

To delete an F-runtime group, proceed as follows:

1. Click on "F-runtime group 1" or "F-runtime group 2" in the area navigation of the *Safety Administration Editor*.
2. Click the "Delete F-runtime group " button in the work area shown below.
3. Confirm the dialog with "Yes".
4. For your changes to become effective, compile the safety program (Page 169) (menu command "Edit > Compile").

The assignment of the F-blocks to an F-runtime group (to the calling block of the main safety block) is deleted in the project tree. However, the F-blocks continue to exist.

Note

If you want to delete your safety program, delete all F-blocks outside the System blocks folder in the project tree.

F-blocks that do not allow deletion are deleted by recompiling the safety program or deactivating the F-capability for the F-CPU (see Configuring the F-CPU (Page 29)).

5.2.5 Modifying an F-runtime group

Modifying an F-runtime group

You can make the following changes for each F-runtime group of your safety program in the associated "F-runtime groups" work area:

- Specify another block as the calling block of the main safety block
- Specify another F-FB or F-FC as main safety block
- Enter a different or new I-DB for the main safety block
- Change the value for the maximum cycle time of the F-runtime group
- Specify another DB as a data block for the F-runtime group communication

5.3 Creating F-blocks in FBD / LAD

5.3.1 Creating F-blocks

Introduction

In order to create F-FBs, F-FCs, and F-DBs for the safety program, you should follow the same basic procedure as for standard blocks. In the following, only the deviations from the procedure for standard blocks are presented.

Creating F-FBs, F-FCs, and F-DBs

You create F-blocks in the same way as for standard blocks. Proceed as follows:

1. Double-click on "Add new block" under "Program blocks" in the project tree.
2. In the dialog that appears, specify the type, name, and language and select the "Create F-block" check box. (If you do not select the check box, a standard block will be created in the project tree.)
3. After the dialog is confirmed, the F-block is opened in the *Program editor*.

Note the following

Note the following important instructions

Note

The main safety block must not contain any parameters, as they cannot be provided with actual values when the block is called.

NOTICE

Editing the instance DB of F-FBs is not permitted online or offline and can cause the F-CPU to go to STOP mode.

Note

Accesses to static parameters of instance DBs of other F-FBs are not permitted.

Note

Output parameters of F-FCs must always be initialized.

The F-CPU can go to STOP if this is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
 - "Data corruption in the safety program prior to output to partner F-CPU"
 - "Safety program: internal CPU fault; internal error information: 404"
-

Note

If you wish to assign an operand from the data area (data block) to a formal parameter of an F-FC as an actual parameter, you must use fully qualified DB access.

Note

Note that access to the input parameters in an F-FB/F-FC is read-only, while access to the output parameters is write-only.

Use an in/out parameter if you wish to have both read and write access.

Note

For greater clarity, assign unique symbolic names to the F-blocks you have created. These symbolic names appear in the project tree and in the block interface. Symbolic names are assigned in the same way as for standard blocks.

If you have inserted an instruction or an F-FB as a multi-instance, you must note the following:

- You must not use static data of the multi-instance as an operand of an instruction or actual parameter of an F-FB/F-FC.
- You may use inputs and outputs of the multi-instance only in the F-FB in which the multi-instance is declared as operand of an instruction or actual parameter of an F-FB / F-FC.

Copying F-blocks

You can copy F-FBs, F-FCs, and F-DBs in exactly the same way as blocks of the standard user program.

Exception:

Blocks from the "Program blocks > System blocks > STEP 7 Safety > Compiler blocks" folder must not be copied

5.3.2 Use libraries

Introduction

You have the option to store F-blocks in global libraries or the project library.

Global libraries

In global libraries, you can store, among others, F-blocks that you would like to reuse across projects. You must explicitly create global libraries. Additional information on global libraries can be found in the *help on STEP 7 Professional* in "Using libraries".

Working with global libraries

If you want to store an F-block in a global library for reuse not just as a "copy template only" but also as an already tested and, if applicable, approved safety function, you must note the following:

- Document the signature and, if applicable, the initial value signature of the F-block after you have finished compiling, testing, and, if applicable, approving.

For documentation purposes, you can create a safety printout of the safety program you used to create the F-block.

- When using the F-block in another safety program, make sure that the signature and, if applicable, the initial value signature of the F-block are unchanged.

For F-blocks with instructions for which a version is indicated in the "Version" column of the "Instructions" task card, you must also note the following:

- Document the versions of these instructions that were set at the time the F-block was compiled and on whose basis the F-block was tested and, if applicable, acceptance tested.

You can use the safety printout for documentation purposes.

- When using the F-block in another safety program, make sure that the documented versions are set for these instructions in the "Instructions" task card.
- If you want to set different versions than the documented versions for these instructions, you must check yourself to determine whether the F-block still works with these versions as intended.

To do so, identify any differences between the versions based on the online help on the instructions.

If necessary, you must arrange for retesting and acceptance testing of the F-block.

Additional information on instruction versions can be found in the *help on STEP 7 Professional* in "Basics for instruction versions".

Project library

In the project library, you can store elements that you would like to use repeatedly within the project. The project library is created and stored automatically along with the project. Additional information on project libraries can be found in the *help on STEP 7 Professional* in "Using libraries".

Working with the project library

Proceed as described in "Working with global libraries".

5.4 Programming startup protection

Introduction

 WARNING
<p>When an F-CPU is switched from STOP to RUN mode, the standard user program starts up in the usual way. When the safety program is started up, all data blocks with F-attribute are initialized with the values from the load memory - as is the case during a cold restart. This means that saved error information is lost.</p> <p>The F-system automatically reintegrates the F-I/O.</p> <p>An operation error or an internal error can also trigger a startup of the safety program with the values from the load memory. If your process does not allow such a startup, you must program a restart/startup prevention in the safety program. The output of process data must be blocked until manually enabled. This enable must not occur until it is safe to output process data and faults have been corrected. (<i>S008</i>)</p>

Example of restart/startup prevention

In order to implement a restart/startup prevention, it must be possible to detect a startup. To detect a startup, you declare a tag of data type BOOL with initial value "TRUE" in an F-DB.

Block the output of process data when this tag has the value "1," for example, by passivating F-I/O using the PASS_ON tag in the F-I/O DB.

To manually enable the output of process data, you reset this tag by means of a user acknowledgment.

See also

F-I/O DB (Page 82)

Implementing User Acknowledgment in the Safety Program of the F-CPU of a DP Master or IO Controller (Page 99)

Implementing user acknowledgment in the safety program of the F-CPU of a I-slave or I-device (Page 102)

F-I/O access

6.1 F-I/O access

Overview

The following describes how you can access the F-I/O and the special characteristics you must consider when programming this access.

Access via the process image

As with standard I/O, you access F-I/O (e.g., S7-300 F-SMs) via the **process image** (PII and PIQ). Direct I/O access is not permitted. The channels of an F-I/O can only be accessed from one F-runtime group.

The update of the process image input is done at the start of the main safety block. The process image output is updated at the end of the main safety block (see Structure of the safety program (Page 56)). For additional information on updating the process image, refer to the note in Data Transfer from the Safety Program to the Standard User Program (Page 105).

The actual communication between the F-CPU (process image) and the F-I/O for the purpose of updating the process image takes place in the background using a special safety protocol in accordance with PROFIsafe.

Note

Due to the special safety protocol, the F-I/O occupies a larger area of the process image than is required for the channels that are actually present on the F-I/O. To find out the area of the process image where the channel data (user data) are stored, refer to the relevant *manuals for the F-I/O*. When the process image is accessed in the safety program, only the channels that are actually present are permitted to be accessed.

Note that for certain F-I/O (such as S7-300 F-SMs and ET 200S fail-safe modules), a "1oo2 evaluation of the sensors can be specified. To find out which of the channels combined by the "1oo2 evaluation of the sensors" you can access in the safety program, refer to the relevant manuals for the F-I/O.

See also

Safety-Related I-Slave-Slave Communication - F-I/O Access (Page 153)

6.2 Process Data or Fail-Safe Values

When are fail-safe values used?

The safety function requires that fail-safe values (0) be used instead of process data for passivation of the entire F-I/O or individual channels of an F-I/O in the following cases. This applies both to digital channels (data type BOOL) as well as for analog channels (data type INT (WORD) or DINT (DWORD)):

- When the F-system starts up
- When errors occur during safety-related communication (communication errors) between the F-CPU and F-I/O using the safety protocol in accordance with PROFIsafe
- When F-I/O faults and channel faults occur (such as wire break, short circuit, and discrepancy errors)
- As long as you enable passivation of the F-I/O with PASS_ON = 1 in the F-I/O DB (see below)

Fail-safe value output for F-I/O/channels of an F-I/O

When **passivation** occurs for an **F-I/O with inputs**, the F-system provides the safety program with fail-safe values (0) in the PII instead of the process data pending at the fail-safe inputs of the F-I/O.

The overflow or underflow of a channel of the **SM 336; AI 6 x 13Bit** or **SM 336; F-AI 6 x 0/4 ... 20 mA HART** is recognized by the F-system as an F-I/O/channel fault. The fail-safe value 0 is provided in place of 7FFF_H (for overflow) or 8000_H (for underflow) in the PII for the safety program.

If you want to process fail-safe values in the safety program other than "0" for an F-I/O with inputs for **analog channels of data type INT (WORD) or DINT (DWORD)**, then you can assign individual fail-safe values for QBAD/QBAD_I_xx/QBAD_O_xx = 1.

WARNING

For an F-I/O with digital input channels (data type BOOL), the value provided in the PII must always be processed in the safety program, regardless of QBAD/QBAD_I_xx/QBAD_O_xx. (S009)

When passivation occurs in an F-I/O with outputs, the F-system transfers fail-safe values (0) to the fail-safe outputs instead of the output values in the PIQ provided by the safety program. The F-system overwrites the associated PIQ with fail-safe values (0).

Reintegration of F-I/O/channels of an F-I/O

The switchover from fail-safe values (0) to process data (**reintegration of an F-I/O**) takes place **automatically** or following **user acknowledgment** in the F-I/O DB. The reintegration method depends on the following:

- The reason for passivation of the F-I/O or channels of the F-I/O
- Parameters you have assigned for the F-I/O DB (Page 82)

Note

Note that for channel faults in the F-I/O, channel-level passivation takes place if configured accordingly in the *hardware and network editor*. For the concerned channels, fail-safe values (0) are output.

Reintegration after channel faults reintegrates all channels whose faults were eliminated; faulty channels remain passivated.

See also

Configuring the F-I/O (Page 33)

6.3 F-I/O DB

Introduction

An F-I/O DB is automatically generated for each F-I/O (in safety mode) when the F-I/O is configured in the *hardware and network editor*. The F-I/O DB contains tags that you can or must evaluate or write to in the safety program. It is not permitted to change the initial values of the tags directly in the F-I/O DB. When an F-I/O is deleted, the associated F-I/O DB is also deleted.

Use of access to an F-I/O DB

You access tags of the F-I/O DB for the following reasons:

- For reintegration of F-I/O after communication errors, F-I/O faults, or channel faults
- If you want to passivate the F-I/O as a function of particular states of your safety program (for example, group passivation)
- For reassignment of parameters for fail-safe DP standard slaves/standard I/O devices
- If you want to evaluate whether fail-safe values or process data are output

Tags of an F-I/O DB

The following table shows the tags of an F-I/O DB:

	Tag	Data type	Function	Initial value
Tags that you can or must write	PASS_ON	BOOL	1=enable passivation	0
	ACK_NEC	BOOL	1=acknowledgment for reintegration required in the event of F-I/O or channel faults	1
	ACK_REI	BOOL	1=acknowledgment for reintegration	0
	IPAR_EN	BOOL	Tag for parameter reassignment of fail-safe DP standard slaves/standard I/O devices or, in the case of SM 336; F-AI 6 x 0/4 ... 20 mA HART, for enabling HART communication	0
Tags that you can evaluate	PASS_OUT	BOOL	Passivation output*	1
	QBAD	BOOL	1=Fail-safe values are output*	1
	ACK_REQ	BOOL	1=acknowledgment requirement for reintegration	0
	IPAR_OK	BOOL	Tag for parameter reassignment of fail-safe DP standard slaves/standard I/O devices or, in the case of SM 336; F-AI 6 x 0/4 ... 20 mA HART, for enabling HART communication	0
	DIAG	BYTE	Service information	
	QBAD_I_xx	BOOL	1=fail-safe values are output to input channel xx	1
	QBAD_O_x x	BOOL	1=fail-safe values are output to output channel xx	1

* For explanations see section below "PASS_OUT/QBAD/QBAD_I_xx/QBAD_O_xx"

PASS_ON

The PASS_ON tag allows you to enable passivation of an F-I/O, for example, as a function of particular states in your safety program.

Using the PASS_ON tag in the F-I/O DB, you can passivate F-I/O; channel-level passivation is not possible.

As long as PASS_ON = 1, **passivation** of the associated F-I/O occurs.

ACK_NEC

If an F-I/O fault is detected by the F-I/O, **passivation** of the relevant F-I/O occurs.. If channel faults are detected, the relevant channels are passivated if channel-level passivation is configured. If passivation of the entire F-I/O is configured, all channels of the relevant F-I/O are passivated. Once the F-I/O fault or channel fault has been eliminated, **reintegration** of the relevant F-I/O occurs, depending on ACK_NEC:

- With ACK_NEC = 0, you can assign an **automatic reintegration**.
- With ACK_NEC = 1, you can assign a **reintegration by user acknowledgment**.

WARNING

Assignment of the ACK_NEC = 0 tag is only allowed if automatic reintegration is permitted for the relevant process from a safety standpoint. (S010)

Note

The initial value for ACK_NEC is 1 following creation of the F-I/O DB. If you do not require automatic reintegration, you must not write ACK_NEC.

ACK_REI

When the F-system detects a communication error or an F-I/O fault for an F-I/O, the relevant F-I/O are passivated. If channel faults are detected, the relevant channels are passivated if channel-level passivation is configured. If passivation of the entire F-I/O is configured, all channels of the relevant F-I/O are passivated. **Reintegration** of the F-I/O/channels of the F-I/O after elimination of faults requires a **user acknowledgment** with a positive edge at the ACK_REI tag of the F-I/O DB:

- After every communication error
- After F-I/O faults or channel faults when ACK_NEC = 1 is assigned

Reintegration after channel faults reintegrates all channels whose faults were eliminated.

Acknowledgment is not possible until tag ACK_REQ = 1.

In your safety program, you must provide a user acknowledgment by means of the ACK_REI tag for each F-I/O.

 WARNING
--

For the user acknowledgement, you must interconnect the ACK_REI tag of the F-I/O DB with a signal generated by an operator input. An interconnection with an automatically generated signal is not allowed. (S011)
--

Note

Alternatively, you can use the "ACK_GL" instruction to carry out reintegration of the F-I/O following communication errors or F-I/O/channel faults (ACK_GL: Global acknowledgment of all F-I/O in an F-runtime group (STEP 7 Safety Advanced V11) (Page 295)).

IPAR_EN

The IPAR_EN tag corresponds to the iPar_EN_C tag in the PROFIsafe bus profile, PROFIsafe Specification V1.20 and higher.

Fail-safe DP standard slaves/standard I/O devices

To find out when this tag must be set or reset when parameters of fail-safe DP standard slaves/standard I/O devices are reassigned, consult the PROFIsafe Specification V1.20 or higher or the documentation for the fail-safe DP standard slave/standard I/O device.

Note that IPAR_EN = 1 does not trigger passivation of the relevant F-I/O.

If passivation is to occur when IPAR_EN = 1, you must also set tag PASS_ON = 1.

HART communication with SM 336; F-AI 6 x 0/4 ... 20 mA HART

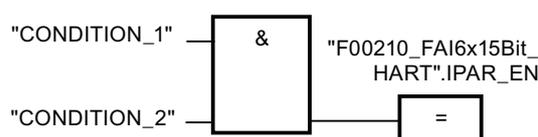
If you set the IPAR_EN tag to "1" while parameter "HART_TOR" = "switchable" is assigned, the HART communication for the SM 336; F-AI 6 x 0/4 ... 20 mA HART is enabled. Setting this tag to "0" disables the HART communication. The F-SM acknowledges the enabled or disabled HART communication with tag IPAR_OK = 1 or 0.

Enable HART communication only when your system is in a status, in which any reassignment of parameters for an associated HART device can be done without any risk.

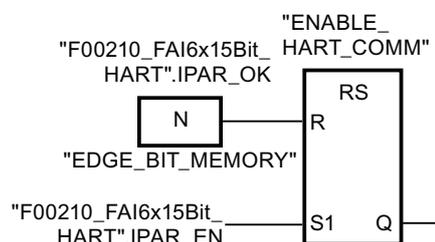
If you want to evaluate the "Enable HART communication" status in your safety program, e.g., for the purpose of programming interlocks, you must build up the information as shown in the following example. This is necessary to ensure that the information is properly available even if communication errors occur while the HART communication is enabled with IPAR_EN = 1. Only change the status of the IPAR_EN tag during this evaluation if there is no passivation due to a communication error or F-I/O/channel fault (PASS_OUT = 0).

Example of enabling HART communication

Network 1: Enable HART communication



Network 2: Determination of Enable HART communication



Further information on HART communication with SM 336; F-AI 6 x 0/4 ... 20 mA HART can be found in the Automation System S7-300, ET 200M Distributed I/O System manual, Fail-Safe Signal Modules (<http://support.automation.siemens.com/WW/view/en/19026151>) manual and in the online help on the module.

PASS_OUT/QBAD/QBAD_I_xx/QBAD_O_xx

If you have configured channel-level passivation for the F-I/O, then PASS_OUT = 1 and QBAD = 1 indicate that at least one channel was passivated. QBAD_I_xx and QBAD_O_xx indicate the input and output channels that were passivated.

If you have configured passivation of the entire F-I/O, tags PASS_OUT = 1 and QBAD = 1 indicate that passivation of the entire F-I/O has occurred.

The **F-system** sets PASS_OUT, QBAD, QBAD_I_xx, and QBAD_O_xx = 1, as long as the fail-safe value is used instead of process data for the associated F-I/O or individual channels of the F-I/O.

If you enable passivation via PASS_ON = 1, however, only QBAD, QBAD_I_xx, and QBAD_O_xx = 1 are set. PASS_OUT does not change its value during passivation via PASS_ON = 1. PASS_OUT can therefore be used for group passivation of additional F-I/O.

ACK_REQ

When the F-system detects a communication error or an F-I/O fault or channel fault for an F-I/O, the relevant F-I/O or individual channels of the F-I/O are passivated. ACK_REQ = 1 signals that **user acknowledgment** is required for reintegration of the relevant F-I/O or channels of the F-I/O.

The F-system sets ACK_REQ = 1 as soon as the fault has been eliminated and user acknowledgment is possible. For channel-level passivation, the F-system sets ACK_REQ = 1 as soon as the channel fault is corrected. User acknowledgement is possible for this fault. Once acknowledgment has occurred, the F-system resets ACK_REQ to 0.

Note

For F-I/O with outputs, acknowledgment after F-I/O faults or channel faults may only be possible some minutes after the fault has been eliminated due to necessary test signal inputs (see *F-I/O manuals*).

IPAR_OK

The IPAR_OK tag corresponds to the iPar_OK_S tag in the PROFIsafe bus profile, PROFIsafe Specification V1.20 and higher.

Fail-safe DP standard slaves/standard I/O devices

To find out how to evaluate this tag when parameters of fail-safe DP standard slaves or standard I/O devices are reassigned, consult the PROFIsafe Specification V1.20 or higher or the documentation for the fail-safe DP standard slave/standard I/O device.

HART communication with SM 336; F-AI 6 x 0/4 ... 20 mA HART

See section "IPAR_EN"

DIAG

The DIAG tag provides non-fail-safe information (1 byte) about errors or faults that have occurred for service purposes. You can read out this information by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program. DIAG bits are saved until you perform an acknowledgment with the ACK_REI tag or until automatic reintegration takes place.

Note

In the safety program, you can use the MOVE instruction to assign this tag to a standard data element.

Structure of DIAG

Bit no.	Assignment	Possible error causes	Remedies
Bit 0	Timeout detected by F-I/O	The PROFIBUS/PROFINET connection between F-CPU and F-I/O is faulty. The value of the F-monitoring time for the F-I/O is set too low. The F-I/O are receiving invalid parameter assignment data or	<ul style="list-style-type: none"> Check the PROFIBUS/PROFINET connection and ensure that there are no external sources of interference. Check the parameter assignment of the F-I/O. If necessary, set a higher value for the monitoring time. Recompile the hardware configuration, and download it to the F-CPU. Recompile the safety program. Check the diagnostics buffer of the F-I/O. Turn the power of the F-I/O off and back on.
		Internal F-I/O fault or	Replace F-I/O
		Internal F-CPU fault	Replace F-CPU
Bit 1	F-I/O fault or channel fault detected by F-I/O	See <i>F-I/O manuals</i>	See <i>F-I/O manuals</i>
Bit 2	CRC error or sequence number error detected by F-I/O	See description for Bit 0	See description for Bit 0
Bit 3	Reserved	—	—
Bit 4	Timeout detected by F-system	See description for Bit 0	See description for Bit 0
Bit 5	Sequence number error detected by F-system	See description for Bit 0	See description for Bit 0
Bit 6	CRC error detected by F-system	See description for Bit 0	See description for Bit 0
Bit 7	Reserved	—	—

See also

- Configuring the F-I/O (Page 33)
- After F-I/O / channel faults (Page 94)
- Group passivation (Page 97)
- MOVE: Move value (STEP 7 Safety Advanced V11) (Page 327)

6.4 Accessing F-I/O DB Variables

Name of the F-I/O DB

An F-I/O DB is automatically generated for each F-I/O when the F-I/O is configured in the *hardware and network editor* and a name is entered for this in the block interface at the same time.

The name is formed by combining the fixed prefix "F", the start address of the F-I/O, and the names entered in the properties for the F-I/O in the *hardware and network editor* (example: F00005_4_8_F_DI_DC24V).

Changing the name and number of the F-I/O DB

The name and number of the F-I/O DB are assigned automatically when the F-I/O are configured. You can change the number in the "Properties" tab of the associated F-I/O. You can also change the name and number in the project tree.

Rule for accessing tags of the F-I/O DB

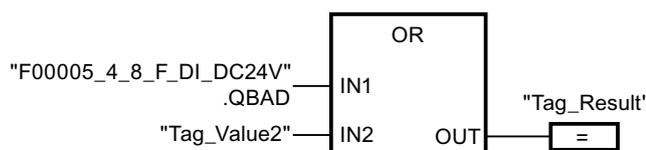
Tags of the F-I/O DB of an F-I/O can only be accessed from one F-runtime group and only from the F-runtime group from which the channels of this F-I/O are accessed (if access is made).

"Fully qualified DB access"

You can access the tags of the F-I/O DB with "fully qualified DB access" (that is, by specifying the name of the F-I/O DB and by specifying the name of the tag).

Example of evaluating the QBAD tag

Network 4: fully qualified access to the QBAD tag



6.5 Passivation and reintegration of F-I/O

Overview

In the following you will find information on passivation and reintegration of F-I/O.

Signal sequence charts

The signal sequences presented below represent typical signal sequences for the indicated behavior.

Actual signal sequences and, in particular, the relative position of the status change of individual signals can deviate from the given signal sequences within the scope of known "fuzzy" cyclic program execution factors, depending on the following:

- The F-I/O used
(F-I/O with inputs, F-I/O with outputs, F-I/O with inputs and outputs, S7-300 F-SMs, ET 200S F-modules, ET 200eco F-modules, ET 200pro F-modules, ET 200iSP F-modules, or fail-safe DP standard slaves/standard I/O devices, version of PROFIsafe bus profile for the F-I/O and F-CPU)
- The cycle time of the OB in which the associated F-runtime group is called
- The target rotation time of the PROFIBUS DP or the update time of the PROFINET IO

Note

The signal sequences shown refer to the status of signals in the user's safety program. If the signals are evaluated in the standard user program before or after the safety program is called in the same OB, the status change of the signals can be displaced by one cycle.

Contrary to what is shown in the signal charts, status changes between process data and fail-safe values that are transmitted to the fail-safe outputs ("To Outputs" signal sequence) can occur before the status change of the associated QBAD signal, if necessary. The timing of the status change is dependent on whether F-I/O with outputs or F-I/O with inputs and outputs were used.

6.5.1 After startup of F-system

Behavior after a startup

After a startup of the F-system, communication between the F-CPU and F-I/O must be established in accordance with the PROFIsafe safety protocol. During this time, **passivation** of the entire F-I/O occurs.

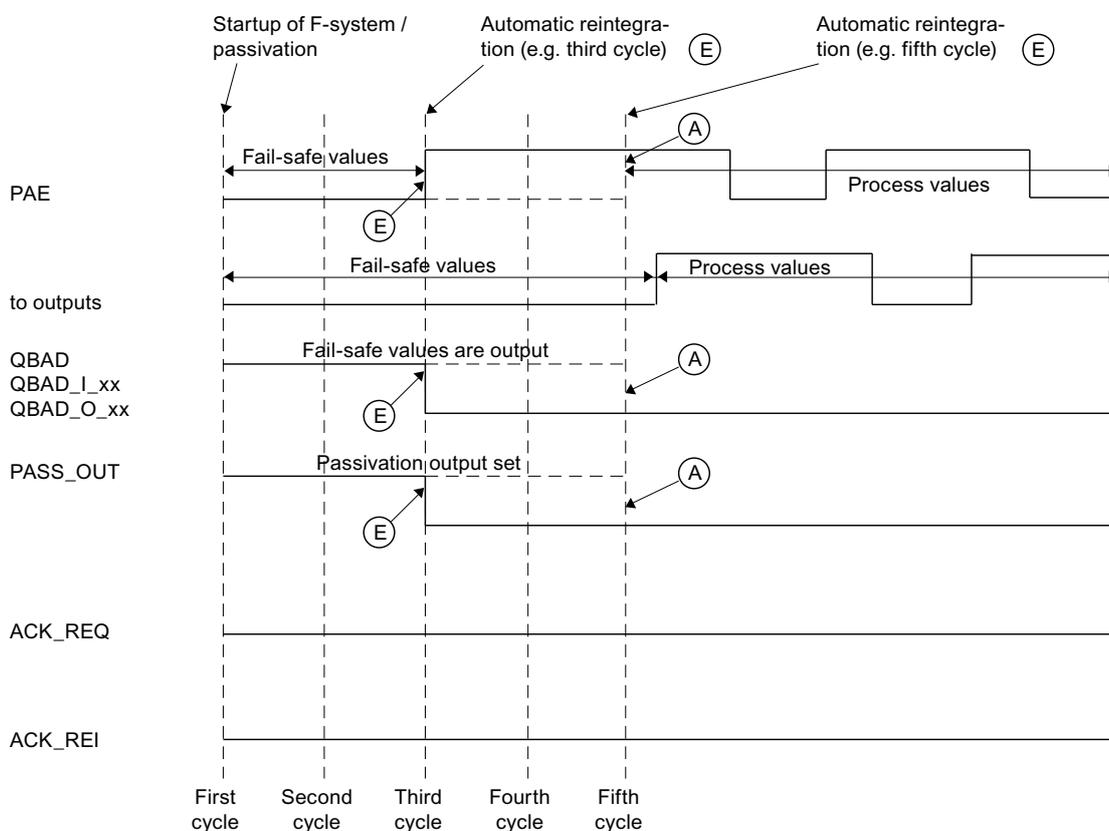
While fail-safe values are being used, tags QBAD, PASS_OUT, QBAD_I_xx, and QBAD_O_xx = 1.

Reintegration of F-I/O

Reintegration of the F-I/O, that is, the provision of process data in the PII or the transfer of process data provided in the PIQ to the fail-safe outputs, takes place **automatically no sooner than** the second cycle of the F-runtime group after startup of the F-system; this happens regardless of the setting for the ACK_NEC tag. Depending on the F-I/O you are using and the cycle time of the F-runtime group and PROFIBUS DP/PROFINET IO, several cycles of the F-runtime group can elapse before reintegration occurs.

If communication between the F-CPU and F-I/O takes longer to establish than the F-monitoring time set in the properties for the F-I/O, automatic reintegration does not take place.

Signal sequence for passivation and reintegration of F-I/O after F-system startup



- (E) for F-I/O with inputs
- (A) for F-I/O with outputs or F-I/O with inputs and outputs

WARNING

When an F-CPU is switched from STOP to RUN mode, the standard user program starts up in the usual way. When the safety program is started up, all data blocks with F-attribute are initialized with the values from the load memory - as is the case during a cold restart. This means that saved error information is lost.

The F-system automatically reintegrates the F-I/O.

A data handling error or an internal error can also trigger a startup of the safety program with the values from the load memory. If your process does not allow such a startup, you must program a restart/startup protection in the safety program: The output of process data must be blocked until manually enabled. This enable must not occur until it is safe to output process data and faults have been corrected. (S008)

See also

- Programming startup protection (Page 76)
- after communication errors (Page 92)

6.5.2 After communication errors

Behavior after communication errors

If the F-system detects an error during safety-related communication (communication error) between the F-CPU and an F-I/O in accordance with the PROFIsafe safety protocol, **passivation** of the relevant F-I/O occurs.

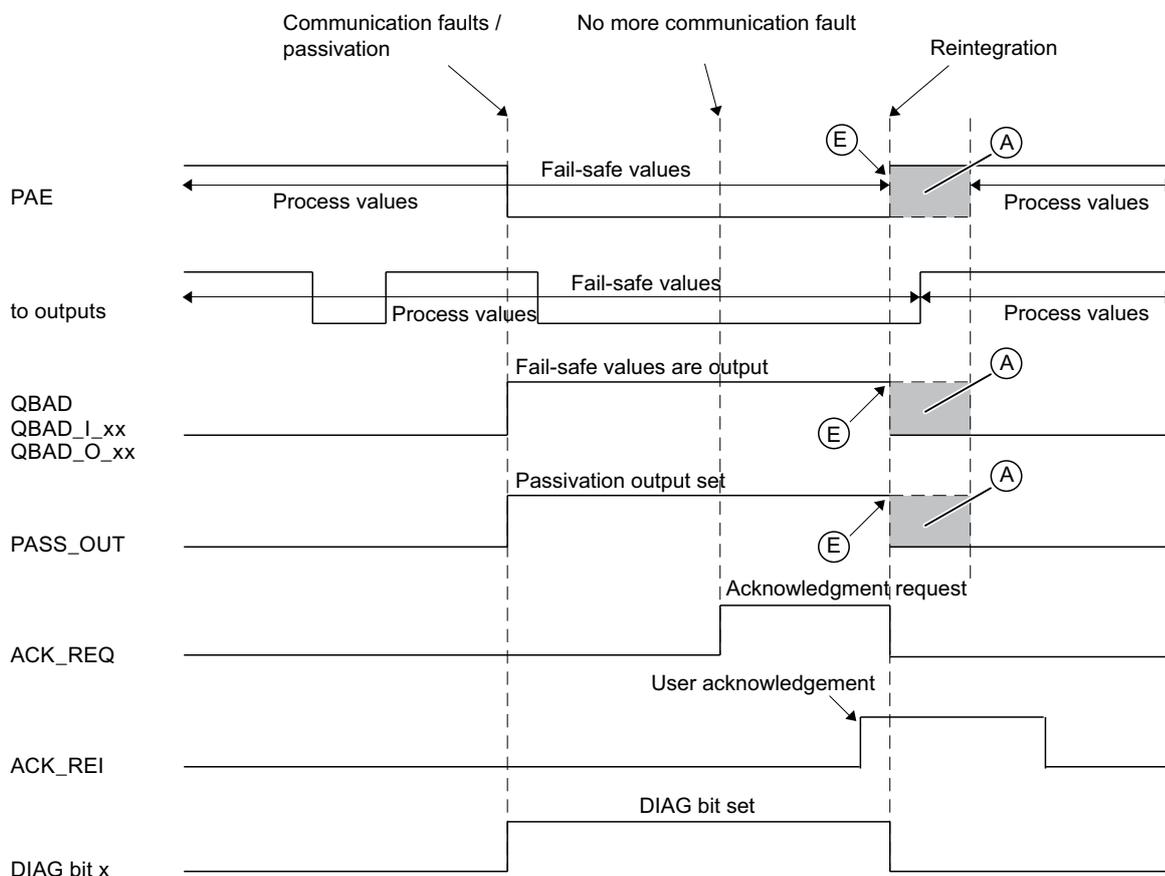
While fail-safe values are being used, tags QBAD, PASS_OUT, QBAD_I_xx, and QBAD_O_xx = 1.

Reintegration of F-I/O

Reintegration of the relevant F-I/O, that is, provision of process data in the PII or transfer of process data provided in the PIQ to the fail-safe outputs, takes place only when the following occurs:

- All communication errors have been eliminated and the F-system has set tag ACK_REQ = 1
- A **user acknowledgment** with a positive edge has occurred:
 - At the ACK_REI tag of the F-I/O DB (Page 82) or
 - At the ACK_REI_GLOB input of the "ACK_GL" instruction (ACK_GL: Global acknowledgment of all F-I/O in an F-runtime group (STEP 7 Safety Advanced V11) (Page 295))

Signal sequence for passivation and reintegration of F-I/O after communication errors



- Ⓔ for F-I/O with inputs
- Ⓐ for F-I/O with outputs and F-I/O with inputs and outputs (Signal sequence depending on the F-I/O used)

See also

Implementing User Acknowledgment in the Safety Program of the F-CPU of a DP Master or IO Controller (Page 99)

Implementing user acknowledgement in the safety program of the F-CPU of a I-slave or I-device (Page 102)

6.5.3 After F-I/O / channel faults

Behavior after F-I/O faults

If the F-system detects an F-I/O fault (parameter assignment error, excess temperature, for example), a **passivation** of all relevant F-I/O occurs.

While fail-safe values are being used, tags QBAD, PASS_OUT, QBAD_I_xx, and QBAD_O_xx = 1.

Behavior after channel faults

If the F-system detects an F-I/O fault (e.g., short circuit, overload, discrepancy error, wire break), the response of the F-system depends on how the "Behavior after channel faults" parameter for the F-I/O is configured.

If you have configured channel-level passivation, the relevant channels of the F-I/O are passivated. While fail-safe values (0) are being used, tags QBAD, PASS_OUT, and QBAD_I_xx and QBAD_O_xx of the relevant channels = 1.

If you have configured passivation of the entire F-I/O, passivation occurs just like after F-I/O errors (see above).

Reintegration of F-I/O

Reintegration of the relevant F-I/O or the relevant channels of the F-I/O, that is, provision of process data in the PII or transfer of process data provided in the PIQ to the fail-safe outputs, takes place only when the following occurs:

- All F-I/O faults or channel faults have been eliminated.

If you have configured channel-level passivation for the F-I/O, the relevant channels are reintegrated once the fault is corrected; any faulty channels remain passivated.

Reintegration takes place as follows, depending on your setting for the ACK_NEC tag:

- When ACK_NEC = 0, **automatic reintegration** takes place as soon as the F-system detects that the fault has been eliminated. For F-I/O with inputs, reintegration takes place right away. For F-I/O with outputs or F-I/O with inputs and outputs, depending on the F-I/O you are using, reintegration can take place several minutes after completion of necessary test signal inputs, which are used by the F-I/O to determine that the fault has been eliminated.
- With ACK_NEC = 1, reintegration takes place only as a result of a user acknowledgement with a positive edge at the ACK_REI tag of the F-I/O DB or at the ACK_REI_GLOB input of the "ACK_GL" instruction. Acknowledgment can be made as soon as the F-system detects that the fault has been eliminated and tag ACK_REQ = 1 has been set.

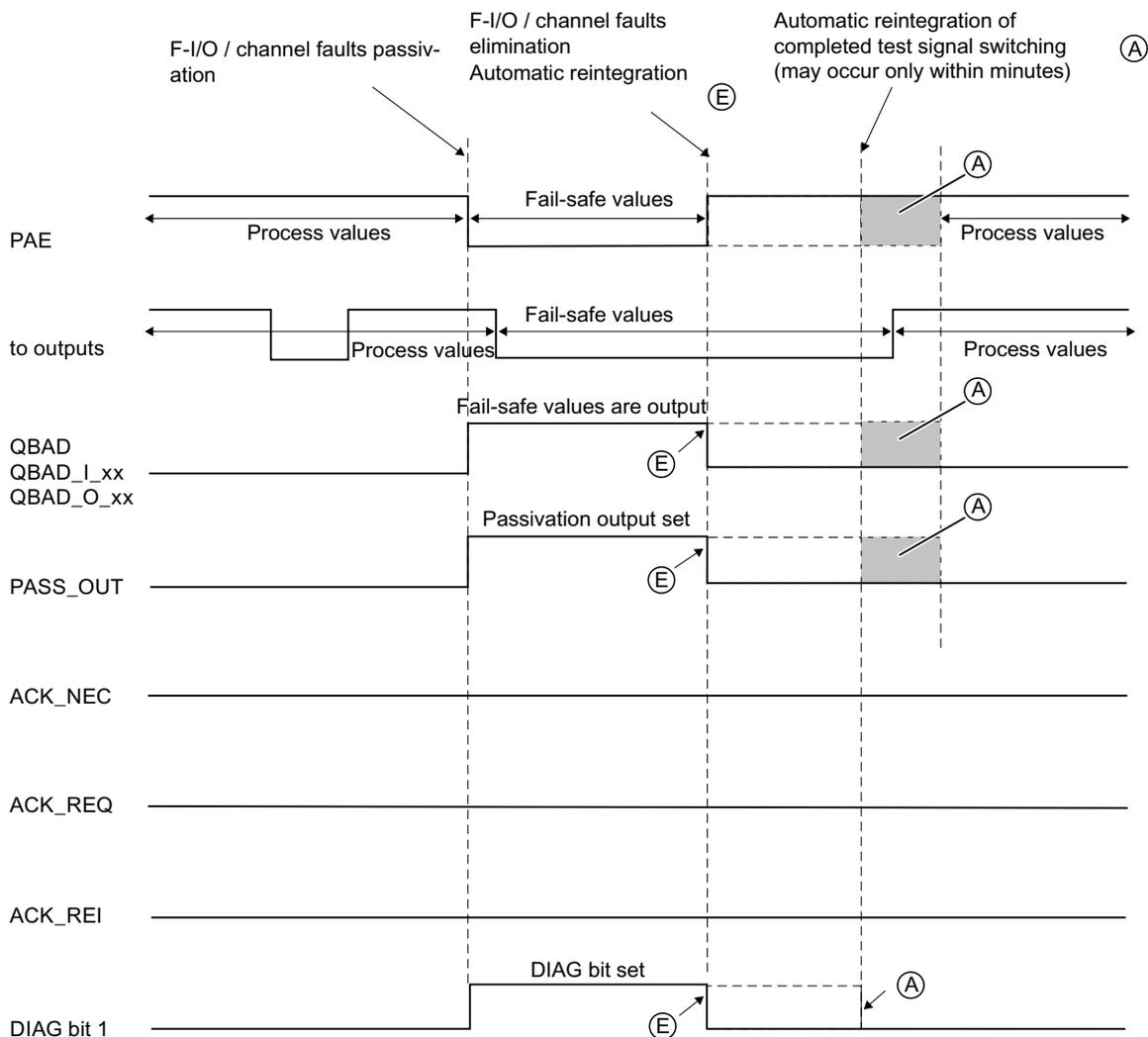
WARNING

Following a power failure of the F-I/O lasting less than the assigned F-monitoring time for the F-I/O, automatic reintegration can occur regardless of your setting for the ACK_NEC tag, as described for the case when ACK_NEC = 0.

If for this case, automatic reintegration is not permitted for the relevant process, you must program startup protection by evaluating tags QBAD or QBAD_I_xx and QBAD_O_xx or PASS_OUT.

In the event of a power failure of the F-I/O lasting longer than the specified F-monitoring time for the F-I/O, the F-system detects a communication error. (S012)

Signal sequence for passivation and reintegration of F-I/O after F-I/O faults or channel faults when ACK_NEC = 0 (for passivation of entire F-I/O after channel faults)



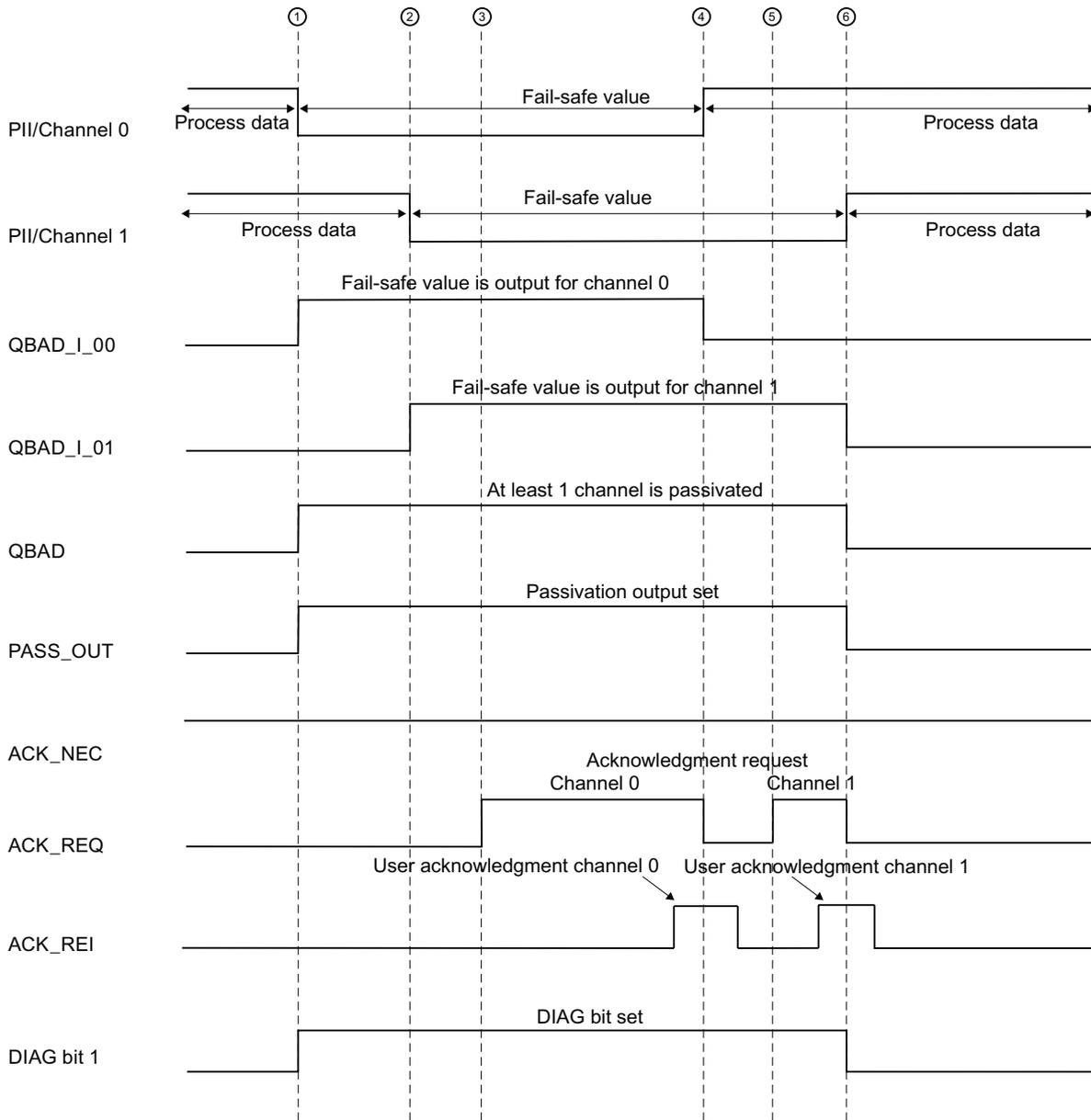
- (E) for F-I/O with inputs
- (A) for F-I/O with outputs or F-I/O with inputs and outputs (Signal sequence depends on the F-I/O used)

Signal sequence for passivation and reintegration of F-I/O after F-I/O faults and channel faults when ACK_NEC = 1 (for passivation of entire F-I/O after channel faults)

For the signal sequence for passivation and reintegration of F-I/O after F-I/O faults or channel faults when ACK_NEC = 1 (initial value), see Passivation and reintegration of F-I/O (Page 89).

Signal sequence for passivation and reintegration of F-I/O after channel faults when ACK_NEC = 1 (for channel-level passivation)

Example for an F-I/O with inputs:



- ① Channel fault for channel 0/passivation of channel 0
- ② Channel fault for channel 1/passivation of channel 1
- ③ Channel fault eliminated for channel 0

- ④ Reintegration of channel 0
- ⑤ Channel fault eliminated for channel 1
- ⑥ Reintegration of channel 1

See also

Programming startup protection (Page 76)

after communication errors (Page 92)

Implementing User Acknowledgment in the Safety Program of the F-CPU of a DP Master or IO Controller (Page 99)

Implementing user acknowledgment in the safety program of the F-CPU of a I-slave or I-device (Page 102)

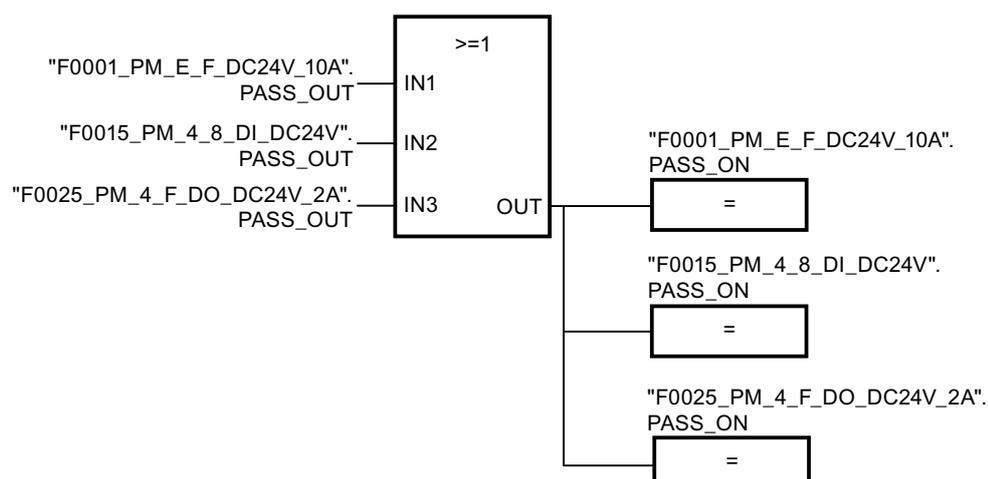
6.5.4 Group passivation**Programming a group passivation**

If you want to enable passivation of additional F-I/O when an F-I/O or a channel of an F-I/O is passivated by the F-system, you can use the PASS_OUT/PASS_ON tags to perform **group passivation** of the associated F-I/O.

Group passivation by means of PASS_OUT/PASS_ON can, for example, be used to force simultaneous reintegration of all F-I/O after startup of the F-system.

For group passivation, you must OR all PASS_OUT tags of the F-I/O in the group and assign the result to all PASS_ON tags of the F-I/O in the group.

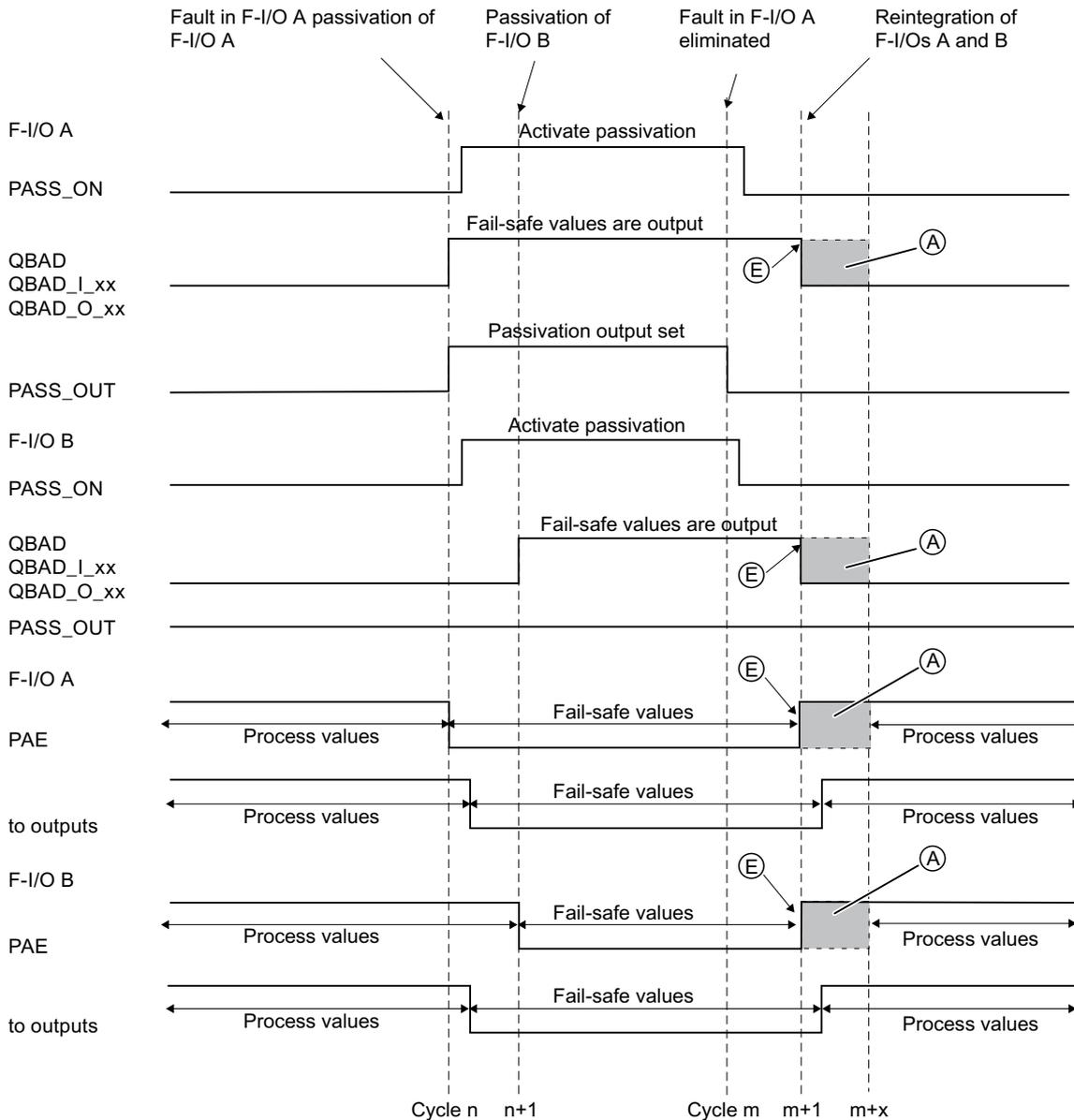
During use of fail-safe values (0) due to group passivation by means of PASS_ON = 1, the QBAD, QBAD_I_xx, and QBAD_O_xx tags of the F-I/O in the group = 1.

Example of group passivation

Reintegration of F-I/O

Reintegration of F-I/O passivated by group passivation occurs **automatically**, if a reintegration **automatic** or **through user acknowledgment** occurs for the F-I/O that triggered the group passivation (PASS_OUT = 0).

Signal sequence for group passivation



- Ⓔ for F-I/O with inputs
- Ⓐ for F-I/O with outputs and F-I/O with inputs and outputs (signal sequence depending on the F-I/O used)

Implementation of user acknowledgment

7.1 Implementing User Acknowledgment in the Safety Program of the F-CPU of a DP Master or IO Controller

Options for user acknowledgment

You have the following options for implementing a user acknowledgment:

- An acknowledgment key that you connect to an F-I/O with inputs
- An operator control and monitoring system

User acknowledgment by means of acknowledgment key

Note

When you implement user acknowledgment by means of acknowledgment key, and a communication error, an F-I/O fault, or a channel fault occurs in the F-I/O to which the acknowledgment key is connected, then it will not be possible to acknowledge the reintegration of this F-I/O.

This "blocking" can only be removed by a STOP-to-RUN transition of the F-CPU.

Consequently, it is recommended that you also provide for an acknowledgment by means of an operator control and monitoring system, in order to acknowledge reintegration of an F-I/O to which an acknowledgment key is connected.

User acknowledgment by means of an operator control and monitoring system

For implementation of a user acknowledgement by means of an operator control and monitoring system, the ACK_OP: Fail-safe acknowledgment (STEP 7 Safety Advanced V11) (Page 351) instruction is required.

Procedure for programming user acknowledgment by means of an operator control and monitoring system

1. Select the "ACK_OP" instruction in the "Instructions" task card and place it in your safety program. The acknowledgment signal is available at the OUT output of ACK_OP for purposes of evaluating the user acknowledgments.
2. On your operator control and monitoring system, set up a field for manual entry of an "acknowledgment value" of "6" (first step in acknowledgment) and an "acknowledgment value" of "9" (second step in acknowledgment) in the instance DB of ACK_OP (input IN).

or

Assign function key 1 to transfer an "acknowledgment value" of "6" (first step in acknowledgment) and function key 2 to transfer an "acknowledgment value" of "9" (second step in acknowledgment) in the instance DB of ACK_OP (input IN).

3. Optional: on your operator control and monitoring system, evaluate output Q in the instance DB of F_ACK_OP to indicate the time frame within which the second step in acknowledgment must occur or to indicate that the first step in acknowledgment has already occurred.

If you want to perform a user acknowledgment exclusively from a programming device or PC using the watch table (monitor/modify tag) without having to deactivate safety mode, then you must transfer an operand (memory word or DBW of a DB of the standard user program) at input IN when calling ACK_OP. You can then transfer "acknowledgment values" "6" and "9" on the programming device or PC by modifying the memory word or DBW of a DB. The memory word or DBW of a DB must not be written by the program.

Note

If you interconnect input IN with a memory word or DBW of a DB, this memory word or DBW of a DB may be used only in an F-runtime group at the ACK_OP instruction and not in another F-runtime group.

WARNING

The two acknowledgment steps must **not** be triggered by one single operation, for example, by automatically storing them along with the time conditions in one program and using one function key to trigger them!

Having two separate acknowledgement steps prevents erroneous triggering of an acknowledgement by your non-fail-safe operator control and monitoring system. (S013)

⚠ WARNING

If your operator control and monitoring system can access multiple F-CPU's that use the ACK_OP instruction for fail-safe acknowledgment, or if you have networked operator control and monitoring systems and F-CPU's (with ACK_OP), you must be sure that the correct F-CPU is in fact being addressed **before** executing the two acknowledgment steps:

- In each F-CPU, store a network-wide unique name for the F-CPU in a DB of your standard user program.
- In your operator control and monitoring system, set up a field from which you can read out the F-CPU name from the DB online before executing the two acknowledgment steps.
- Optional: in your operator control and monitoring system, set up a field to permanently store the F-CPU name. Then, you can determine whether the intended F-CPU is being addressed by simply comparing the F-CPU name read out online with the permanently stored name. (S014)

Example of procedure for programming a user acknowledgment for reintegrating an F-I/O

1. Optional: set the ACK_NEC tag in the respective F-I/O DB (Page 82) to "0" if automatic reintegration (without user acknowledgment) is to take place after an F-I/O fault or a channel fault.

⚠ WARNING

Assignment of the ACK_NEC = 0 tag is only allowed if automatic reintegration is permitted for the relevant process from a safety standpoint. (S010)

2. Optional: Evaluate the QBAD or QBAD_I_xx and QBAD_O_xx or DIAG tags in the respective F-I/O DB to trigger an indicator light, if applicable, in the event of an error, and/or generate error messages on your operator control and monitoring system in your standard user program by evaluating QBAD or QBAD_I_xx and QBAD_O_xx or DIAG; these messages can be evaluated before performing the acknowledgment operation. Alternatively, you can evaluate the diagnostic buffer of the F-CPU.
3. Optional: Evaluate the ACK_REQ tag in the respective F-I/O DB, for example, in the standard user program or on the operator control and monitoring system, to query or to indicate whether user acknowledgment is required.
4. Assign the input of the acknowledgment key or the OUT output of the ACK_OP instruction to the ACK_REI tag in the respective F-I/O DB or the ACK_REI_GLOB input of the ACK_GL instruction (see above).

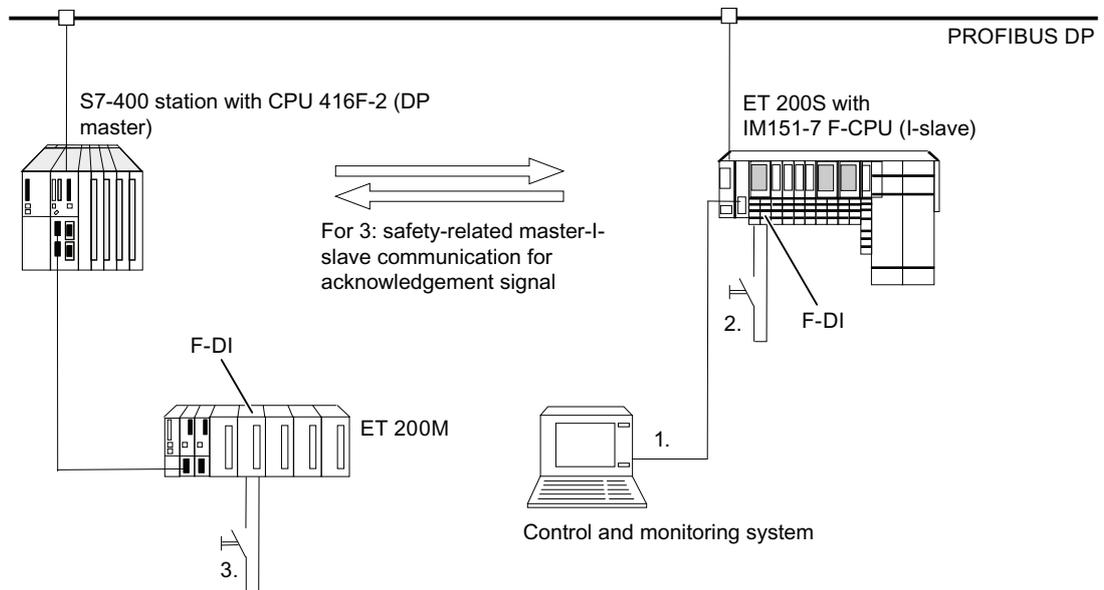
7.2 Implementing user acknowledgment in the safety program of the F-CPU of a I-slave or I-device

Options for user acknowledgment

You can implement a user acknowledgment by means of:

- An operator control and monitoring system that you can use to access the F-CPU of the I-slave/I-Device
- An acknowledgment key that you connect to an F-I/O with inputs that is assigned to the F-CPU of the I-slave/I-Device
- An acknowledgment key that you connect to an F-I/O with inputs that is assigned to the F-CPU of the DP master/IO Controller

These three options are illustrated in the figure below.



1. User acknowledgement by means of an operator control and monitoring system that you can use to access the F-CPU of the I-slave/I-Device

For implementation of a user acknowledgment by means of an operator control and monitoring system that you can use to access the F-CPU of the I-slave/I-Device, the ACK_OP: Fail-safe acknowledgment (STEP 7 Safety Advanced V11) (Page 351) instruction is required.

Programming procedure

Follow the procedure described in "Implementing User Acknowledgment in the Safety Program of the F-CPU of a DP Master or IO Controller (Page 99)" under "Programming Procedure ...".

From your operator control and monitoring system, you can then directly access the instance DB of ACK_OP in the I-slave/I-Device.

2. User acknowledgment by means of an acknowledgment key at an F-I/O with inputs that is assigned to the F-CPU of the I-slave/I-Device

Note

In the event of a communication error, F-I/O fault, or channel fault in the F-I/O to which the acknowledgment key is connected, an acknowledgment for reintegration of this F-I/O is no longer possible.

This "blocking" can only be removed by a STOP-to-RUN transition of the F-CPU of the I-slave/I-Device.

Consequently, it is recommended that you also provide for an acknowledgment by means of an operator control and monitoring system that you can use to access the F-CPU of the I-slave/I-Device, in order to acknowledge reintegration of an F-I/O to which an acknowledgment key is connected (see 1).

3. User acknowledgment by means of acknowledgment key at an F-I/O with inputs that is assigned to the F-CPU of the DP master/IO Controller

If you want to use the acknowledgment key that is assigned to the F-CPU at the DP master/IO Controller to also perform a user acknowledgment in the safety program of the F-CPU of an I-slave/I-Device, you must transmit the acknowledgment signal from the safety program in the F-CPU of the DP master/IO Controller to the safety program in the F-CPU of the I-slave/I-Device by means of safety-related master-I-slave/IO Controller-I-Device communication.

Programming procedure

1. Place the SENDDP (Page 356) instruction in the safety program in the F-CPU of the DP master/IO Controller.
2. Place the RCVDP (Page 356) instruction in the safety program in the F-CPU of the I-slave/I-Device.
3. Supply an input SD_BO_xx of SENDDP with the input of the acknowledgment key.
4. The acknowledgment signal for evaluating user acknowledgments is now available at the corresponding RD_BO_xx output of RCVDP.

The acknowledgment signal can now be read in the program sections in which further processing is to take place with fully qualified access directly in the associated instance DB (for example, "RCVDP_DB".RD_BO_02).

5. Supply the corresponding input SUBBO_xx of RCVDP with the fail-safe value "VKE0" so that an unintentional user acknowledgment is not triggered before communication is established the first time after startup of the sending and receiving F-systems, or in the event of a safety-related communication error. "VKE0" is available in the F-shared-DB. At input SUBBO_xx, enter "F_GLOBDB".VKE0 fully qualified.

Note

If a communication error, an F-I/O fault, or a channel fault occurs at the F-I/O to which the acknowledgment key is connected, then an acknowledgment for reintegration of this F-I/O will no longer be possible.

This "blocking" can only be removed by a STOP-to-RUN transition of the F-CPU of the DP master/IO Controller.

Consequently, it is recommended that you also provide for an acknowledgment by means of an operator control and monitoring system that you can use to access the F-CPU of the DP master/IO Controller, in order to acknowledge reintegration of the F-I/O to which an acknowledgment key is connected.

If a safety-related master-I-slave/IO Controller-I-Device communication error occurs, the acknowledgment signal cannot be transmitted, and an acknowledgment for reintegration of safety-related communication is no longer possible.

This "blocking" can only be removed by a STOP-to-RUN transition of the F-CPU of the I-slave/I-Device.

Consequently, it is recommended that you also provide for an acknowledgment by means of an operator control and monitoring system that you can use to access the F-CPU of the I-slave/I-Device, in order to acknowledge reintegration of the safety-related communication for transmission of the acknowledgment signal (see 1).

Data exchange between standard user program and safety program

8

8.1 Data Transfer from the Safety Program to the Standard User Program

Data transfer from the safety program to the standard user program

The standard user program can read all data of the safety program, for example, through symbolic (fully qualified) accesses to the following:

- Instance DBs of the F-FBs
- F-DBs (for example, "Name F_DB".Signal_1)
- Process image input and process image output of F-I/O (for example, "Emergency_Stop_Button_1" (I 5.0))

Note

The process image input of F-I/O is updated not only at the start of the main safety block, but also by the standard operating system.

To find out the standard operating system update times, refer to the *help on STEP 7 Professional* under "Process image input and output". For F-CPU's that support process image partitions, also bear in mind the update times when process image partitions are used. For this reason, when the process image input of F-I/O is accessed in the standard user program, it is possible to obtain different values than in the safety program. The differing values can occur due to:

- Different update times
- Use of fail-safe values in the safety program

To obtain the same values in the standard user program as in the safety program, you must not access the process image input in the standard program until after execution of an F-runtime group. In this case, you can also evaluate the QBAD or QBAD_I_xx tag in the associated F-I/O DB in the standard user program, in order to find out whether the process image input is receiving fail-safe values (0) or process data. When using process image partitions, also make sure that the process image is not updated by the standard operating system or by the UPDAT_PI instruction between execution of an F-runtime group and evaluation of the process image input in the standard user program.

In addition, you can write safety program data directly to the standard user program (see also table of supported operand areas in: Restrictions in the programming languages FBD/LAD (Page 59)).

8.1 Data Transfer from the Safety Program to the Standard User Program

Bit memory

In order to write safety program data directly to the standard user program, you can also write to bit memory in the safety program. However, written bit memory must not be read in the safety program itself.

	From standard user program		From safety program	
	Read-access	Write-access	Read-access	Write-access
Bit memory	Permitted	Permitted	Bit memory can either be read-accessed <i>or</i> write-accessed	

Data block

In order to write safety program data directly to the standard user program (e.g., DIAG output of the SENDDP instruction), you can write to data blocks of the standard user program from the safety program. However, a written data element must not be read in the safety program itself.

	From standard user program		From safety program	
	Read-access	Write-access	Read-access	Write-access
DB	Permitted	Permitted	A data element in the DB can be read-accessed <i>or</i> write-accessed	
F-DB	Permitted	Not permitted	Permitted	Permitted

Process image output

The process image output (PIQ) of standard I/O can be written to in the safety program, e.g., for display purposes. The PIQ must not be read in the safety program.

		From standard user program		From safety program	
		Read-access	Write-access	Read-access	Write-access
Process image of standard I/O	PII	Permitted	Permitted	Permitted	Not permitted
	PIQ	Permitted	Permitted	Not permitted	Permitted
Process image of F-I/O	PII	Permitted	Not permitted	Permitted	Not permitted
	PIQ	Permitted	Not permitted	Not permitted	Permitted

8.1.1 F-Shared DB

Function

The F-shared data block is a fail-safe data block that contains all of the shared data of the safety program and additional information needed by the F-system. The F-shared DB is automatically inserted when the hardware configuration is compiled.

Using its name F_GLOBDB, you can evaluate certain data of the safety program in the standard user program.

Reading an F-shared DB in the standard user program

You can read out the following information in the F-shared DB in the standard user program or on an operator control and monitoring system:

- Operating mode: safety mode or deactivated safety mode ("MODE" tag)
- Error information "Error occurred when executing safety program" ("ERROR" tag)
- Collective F-signature ("F_PROG_SIG" tag)
- Compilation date of the safety program ("F_PROG_DAT" tag, DATE_AND_TIME data type)

You use fully qualified access to access these tags (e.g. ""F_GLOBDB".MODE").

8.2 Data Transfer from Standard User Program to Safety Program

Data transfer from standard user program to safety program

As a basic principle, only fail-safe data or fail-safe signals from fail-safe I/O and other safety programs (in other F-CPUs) can be processed in the safety program, since standard data and signals are not safe.

If you nevertheless have to process data from the standard user program in the safety program, you can evaluate either bit memory from the standard user program, data of a standard DB, or the process image input (PII) of standard I/O in the safety program (see table of supported operand areas in: Restrictions in the programming languages FBD/LAD (Page 59)).

 WARNING
Because these data are not generated safely, you must carry out additional process-specific validity checks in the safety program to ensure that no dangerous states can arise. If bit memory, data of a standard DB, or an input of standard I/O is used in both F-runtime groups, you must perform the validity check separately in each F-runtime group. (S015)

To facilitate the checks, all signals from the standard user program that are evaluated in the safety program are included when the safety program is printed out.

Bit memory

In order to process data of the standard user program in the safety program, you can also read bit memory in the safety program. However, read bit memory must not be written in the safety program itself.

	From standard user program		From safety program	
	Read-access	Write-access	Read-access	Write-access
Bit memory	Permitted	Permitted	Bit memory can be read-accessed <i>or</i> write-accessed	

Data block

In order to process data of the standard user program in the safety program, you can read data blocks of the standard user program in the safety program. However, a read data element must not be written in the safety program itself.

	From standard user program		From safety program	
	Read-access	Write-access	Read-access	Write-access
DB	Permitted	Permitted	A data element in the DB can be read-accessed <i>or</i> write-accessed	
F-DB	Permitted	Not permitted	Permitted	Permitted

Process image input

You can read the process image input (PII) of standard I/O in the safety program. The PII must not be written in the safety program.

		From standard user program		From safety program	
		Read-access	Write-access	Read-access	Write-access
Process image of standard I/O	PII	Permitted	Permitted	Permitted	Not permitted
	PIQ	Permitted	Permitted	Not permitted	Permitted
Process image of F-I/O	PII	Permitted	Not permitted	Permitted	Not permitted
	PIQ	Permitted	Not permitted	Not permitted	Permitted

Examples: Programming validity checks

- Use Comparison (Page 468) instructions to check whether unsafe data from the standard user program exceed or fall below permitted upper and lower limits. You can then influence your safety function with the result of the comparison.
- For example, use the S: Set output (STEP 7 Safety Advanced V11) (Page 381), R: Reset output (STEP 7 Safety Advanced V11) (Page 380), or SR: Set/reset flip-flop (STEP 7 Safety Advanced V11) (Page 383) instructions with unsafe signals from the standard user program to allow a motor to be switched off, but not switched on.
- For switch-on sequences, for example, use the AND logic operation instruction to logically combine unsafe signals from the standard user program with switch-on conditions that you derive from fail-safe signals.

If you want to process unsafe data in the safety program, bear in mind that a sufficiently simple method of checking validity does not exist for all unsafe data.

Reading data from the standard user program that can change during runtime of an F-runtime group

When you want to read data from the standard user program (bit memory, data of a standard DB, or PII of standard I/O) in the safety program, and these data can be changed - either by the standard user program or an operator control and monitoring system – during runtime of the F-runtime group in which they are read (e.g., because your standard user program is processed by a higher-priority cyclic interrupt), you must use bit memory or data of a standard DB for this purpose. You must write the bit memory or data of a standard DB with the data from the standard user program immediately before calling the F-runtime group. You are then permitted to access only this bit memory or data of a standard DB in the safety program.

Also note that **clock memory** that you defined when configuring your F-CPU in the "Properties" tab can change during runtime of the F-runtime group, since clock memory runs asynchronously to the F-CPU cycle.

Note

The F-CPU can go to STOP if the information above is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
 - "Data corruption in the safety program prior to output to partner F-CPU"
 - "Safety Program: internal CPU fault; internal error information: 404"
-

Configuring and Programming Communication

9.1 Overview of communication

Introduction

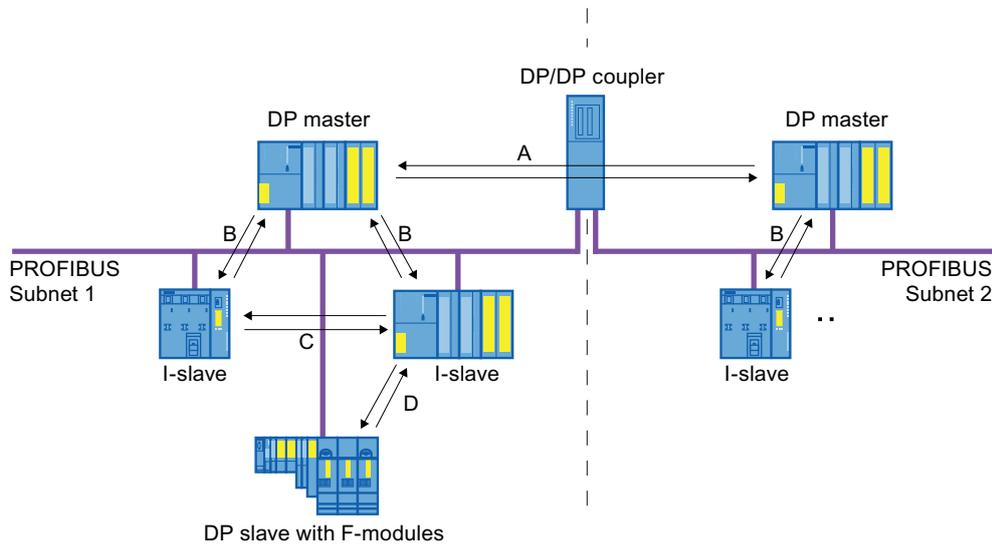
Here you receive an overview of the safety-related communication options in SIMATIC Safety F-systems.

Options for safety-related communication

Safety-related communication	On subnet	Additionally required hardware
I-slave-slave communication	PROFIBUS DP	—
Safety-related CPU-CPU communication:		
Master-master communication	PROFIBUS DP	DP/DP coupler
Master-master communication for S7 Distributed Safety	PROFIBUS DP	DP/DP coupler
Master-I-slave communication	PROFIBUS DP	—
I-slave-I-slave communication	PROFIBUS DP	—
IO controller-IO controller communication	PROFINET IO	PN/PN coupler
IO Controller-IO Controller communication for S7 Distributed Safety	PROFINET IO	PN/PN coupler
IO Controller-I-Device communication	PROFINET IO	—
IO Controller-I-slave communication	PROFINET IO and PROFIBUS DP	IE/PB Link
CPU-CPU communication via S7 connections	Industrial Ethernet	—
CPU-CPU communication to S7 Distributed Safety or S7 F Systems via S7 connections	Industrial Ethernet	—

Overview of safety-related communication via PROFIBUS DP

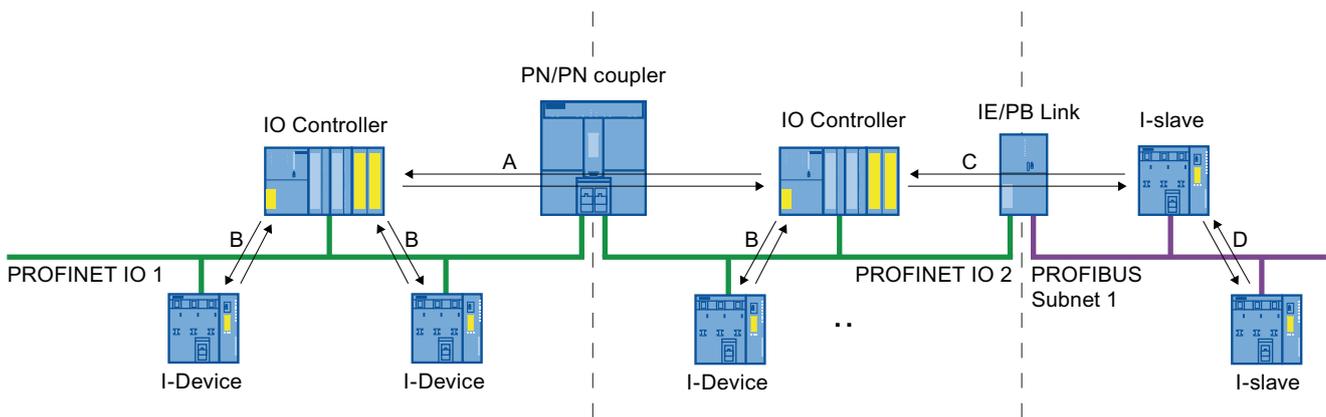
The figure below presents an overview of the four options for safety-related communication via PROFIBUS DP in SIMATIC Safety F-systems.



- A Safety-related master-master communication
- B Safety-related master-I-slave communication
- C Safety-related I-slave-I-slave communication
- D Safety-related I-slave-slave communication

Overview of safety-related communication via PROFINET IO

The figure below presents an overview of the three options for safety-related communication via PROFINET IO in SIMATIC Safety F-systems. If an IE/PB-link is used, safety-related communication is possible between assigned I-slaves.



- A Safety-related IO controller-IO controller communication
- B Safety-related IO Controller-I-Device communication
- C Safety-related IO controller-I-slave communication
- D Safety-related I-slave-I-slave communication

Safety-related CPU-CPU communication via PROFIBUS DP or PROFINET IO

In safety-related CPU-CPU communication, a fixed amount of fail-safe data of data types BOOL and INT is transmitted in a fail-safe manner between the safety programs in F-CPU's of DP masters/I-slaves or IO Controllers/I-Devices.

The data are transferred using the SENDDP instruction for sending and the RCVDP instruction for receiving. The data are stored in configured address areas/transfer areas of the devices.

Safety-related I-slave-slave communication via PROFIBUS DP

Safety-related I-slave-slave communication is possible with F-I/O in a DP slave that supports safety-related I-slave-slave communication, e.g., with all ET 200S F-modules and with all S7-300 fail-safe signal modules with IM 153-2, order no. 6ES7153-2BA01-0XB0 or higher, firmware version > V4.0.0.

Safety-related communication between the safety program of the F-CPU of an I-slave and F-I/O of a DP slave takes place using direct data exchange – same as in standard programs. The process image input is used to access the channels of the F-I/O in the safety program of the F-CPU of the I-slave.

Safety-related CPU-CPU communication via Industrial Ethernet

Safety-related CPU-CPU communication via Industrial Ethernet is possible by means of S7 connections, both from and to the following:

- S7-300 F-CPU's via the integrated PROFINET interface
- S7-400 F-CPU's via the integrated PROFINET interface or a CP 443-1 Advanced-IT

In safety-related communication via S7 connections, a specified amount of fail-safe data of data type BOOL, INT, WORD, DINT, DWORD, or TIME is transferred in a fail-safe manner between the safety programs of the F-CPU's linked by means of the S7 connection.

The data transfer makes use of the SENDS7 instruction for sending and the RCVS7 instruction for receiving. Data are exchanged using one F-DB ("F-communication DB") each on the sender and receiver sides.

Safety-related CPU-CPU communication to *S7 Distributed Safety* or *F Systems*

Safety-related communication from F-CPU's in *SIMATIC Safety V11* to F-CPU's in *S7 Distributed Safety* or *S7 F-systems* is possible.

9.2 Safety-related IO controller-IO controller communication

9.2.1 Configure safety-related IO controller-IO controller communication

Introduction

Safety-related communication between safety programs of the F-CPU's of IO Controllers takes place over a PN/PN coupler that you set up between the F-CPU's.

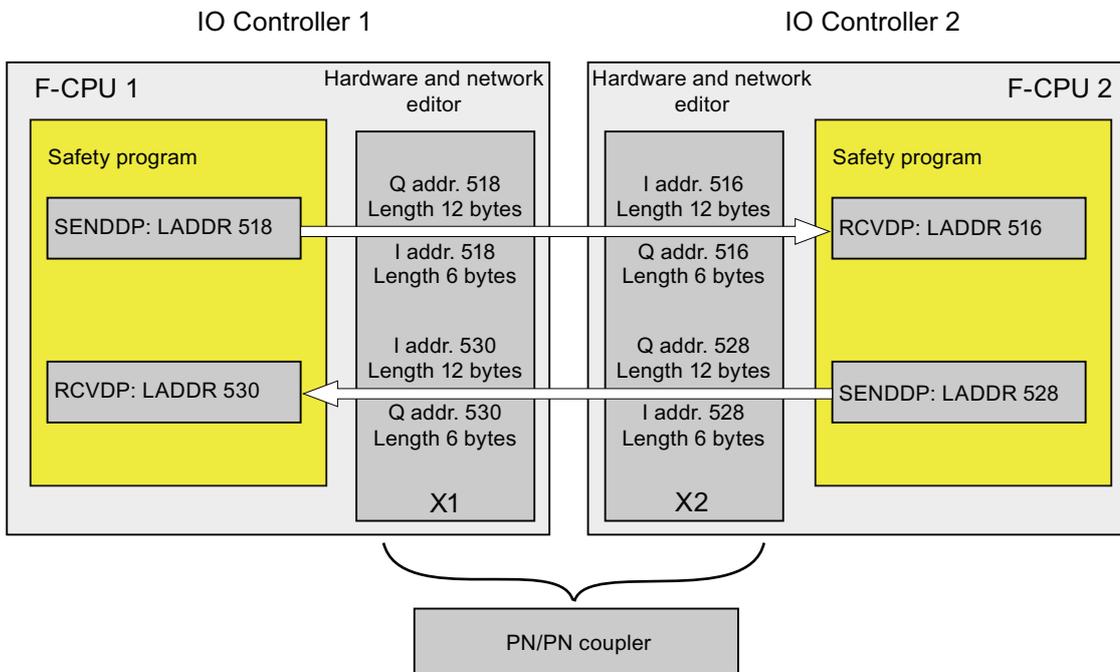
In the case of a CPU 416F-2 DP without an integrated PROFINET interface, you use a 443-1 Advanced-IT.

Note

Disable the "Data validity display DIA" parameter in the properties for the PN/PN coupler in the *hardware and network editor* (same as default setting). Otherwise, safety-related IO Controller-IO Controller communication is not possible.

Configuring address areas

You must configure one address area for output data and another address area for input data in the PN/PN coupler in the *hardware and network editor* for each safety-related communication connection between two F-CPU's via a PN/PN coupler. In the figure below, both of the F-CPU's are to be able to send and receive data (bidirectional communication).



Rules for defining the address areas

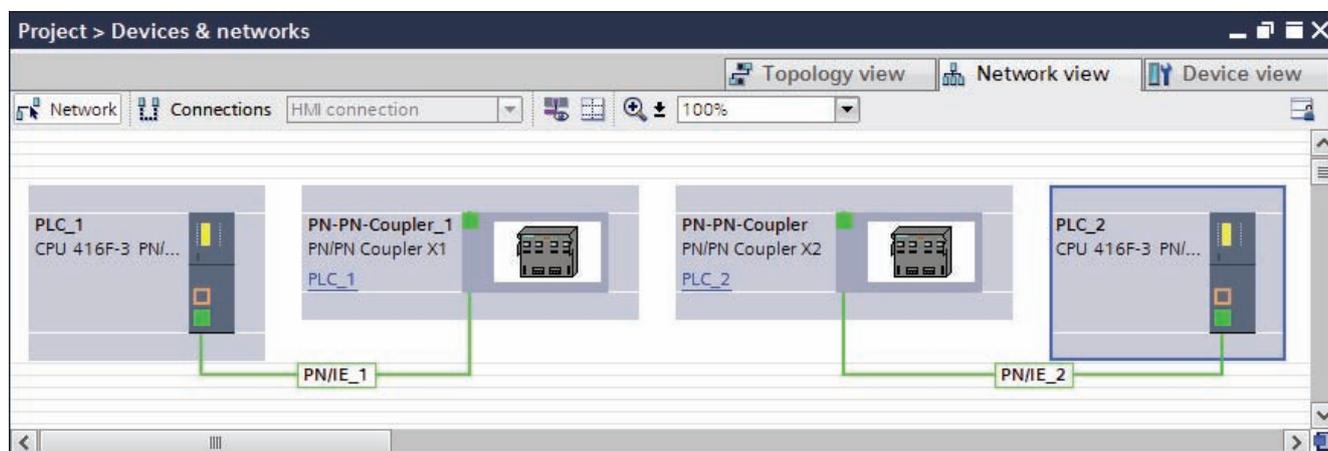
The address area for output data for the **data to be sent** must begin with the same start address as the associated address area for input data. A total of 12 bytes (consistent) is required for the address area for output data, while 6 bytes (consistent) are required for the address area for input data.

The address area for input data for the **data to be received** must begin with the same start address as the associated address area for output data. A total of 12 bytes (consistent) is required for the address area for input data, while 6 bytes (consistent) are required for the address area for output data.

Procedure for configuration

The procedure for configuring safety-related IO Controller-IO Controller communication is identical to that in the standard system. Proceed as follows:

1. Insert two F-CPUS from the "Hardware catalog" task card in the project.
2. Select PN/PN Coupler X1 from "Other field devices\PROFINET IO\Gateway\Siemens AG\PN/PN Coupler" in the "Hardware catalog" task card and insert it into the network view of the hardware and network editor.
3. Insert a PN/PN Coupler X2 into the network view.
4. Connect the PN interface of the F-CPU 1 with the PN interface of the PN/PN Coupler X1 and the PN interface of the F-CPU 2 with the PN interface of PN/PN Coupler X2.



5. For bidirectional communication connections, i.e., each F-CPU is to send and receive data, select the following modules from "IN/OUT" in the "Hardware catalog" task card (with filter activated), and insert them into the "Device overview" tab in the device view of PN/PN Coupler X1:
 - One "IN/OUT 6 bytes/12 bytes" module, and
 - One "IN/OUT 12 bytes/6 bytes" module

6. In the properties of the modules, assign the addresses outside the process image as follows:

For "IN/OUT 6 bytes/12 bytes" module, e.g.:

- Output addresses: Start address 518
- Input addresses: Start address 518

For "IN/OUT 12 bytes/6 bytes" module, e.g.:

- Output addresses: Start address 530
- Input addresses: Start address 530

Note

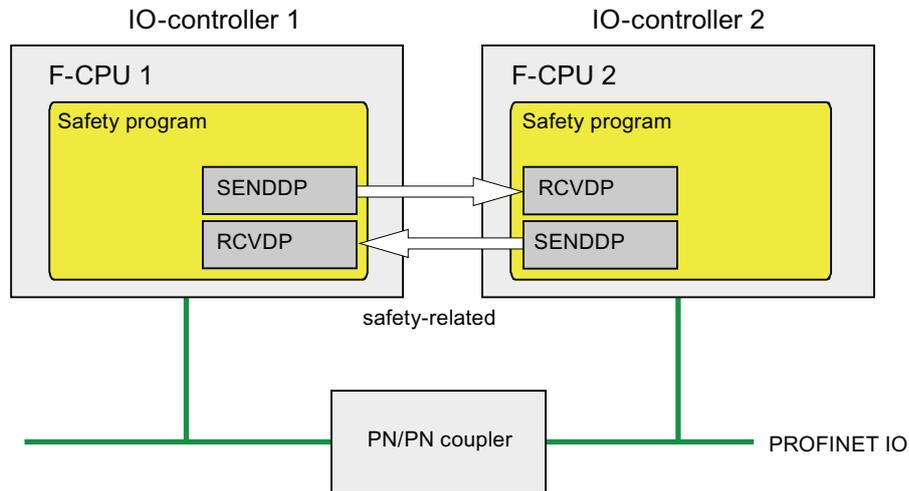
Make sure that the values you assign for the start addresses of the output and input data address areas are identical.

Module	Rack	Slot	I address	Q address	Type	Order no.	Firmware
PN-PN-Coupler_1	0	0	16378*		PN/PN Coupler X1	6ES7 158-3AD01-0XA0	V03.00.00
Interface	0	0 X1	16377*		PN-PN-Coupler		
IN/OUT 6 Bytes / 12 Bytes_1	0	1	518...523	518...529	IN/OUT 6 Bytes / 12 Bytes		
IN/OUT 12 Bytes / 6 Bytes_1	0	2	530...541	530...535	IN/OUT 12 Bytes / 6 Bytes		
	0	3					
	0	4					
	0	5					
	0	6					
	0	7					

7. Next, define the modules and addresses for PN/PN Coupler X2. To do so, follow the procedure just outlined. Use the same module types as those for X1. You must adapt the addresses correspondingly. (See figure at the beginning of "Configuring address areas").

9.2.2 Safety-related IO controller-IO controller communication via SENDDP and RCVDP

Communication via SENDDP and RCVDP instructions



Safety-related communication between the F-CPU's of the IO Controllers makes use of the SENDDP and RCVDP instructions for sending and receiving, respectively. They can be used to transfer a *fixed* amount of fail-safe data of data types BOOL and INT in a fail-safe manner.

You will find these instructions in the "Instructions" task card under "Communication". The RCVDP instruction **must** be called at the start of the main safety block. The SENDDP instruction **must** be called at the end of the main safety block.

Note that the send signals are not sent before calling the SENDDP instruction at the end of the relevant F-runtime group execution.

A detailed description of the SENDDP and RCVDP instructions is found in SENDDP and RCVDP: Send and receive data via PROFIBUS DP/PROFINET IO (STEP 7 Safety Advanced V11) (Page 510).

9.2.3 Program safety-related IO controller-IO controller communication

Requirement for programming

The address areas for input and output data for the PN/PN coupler must be configured.

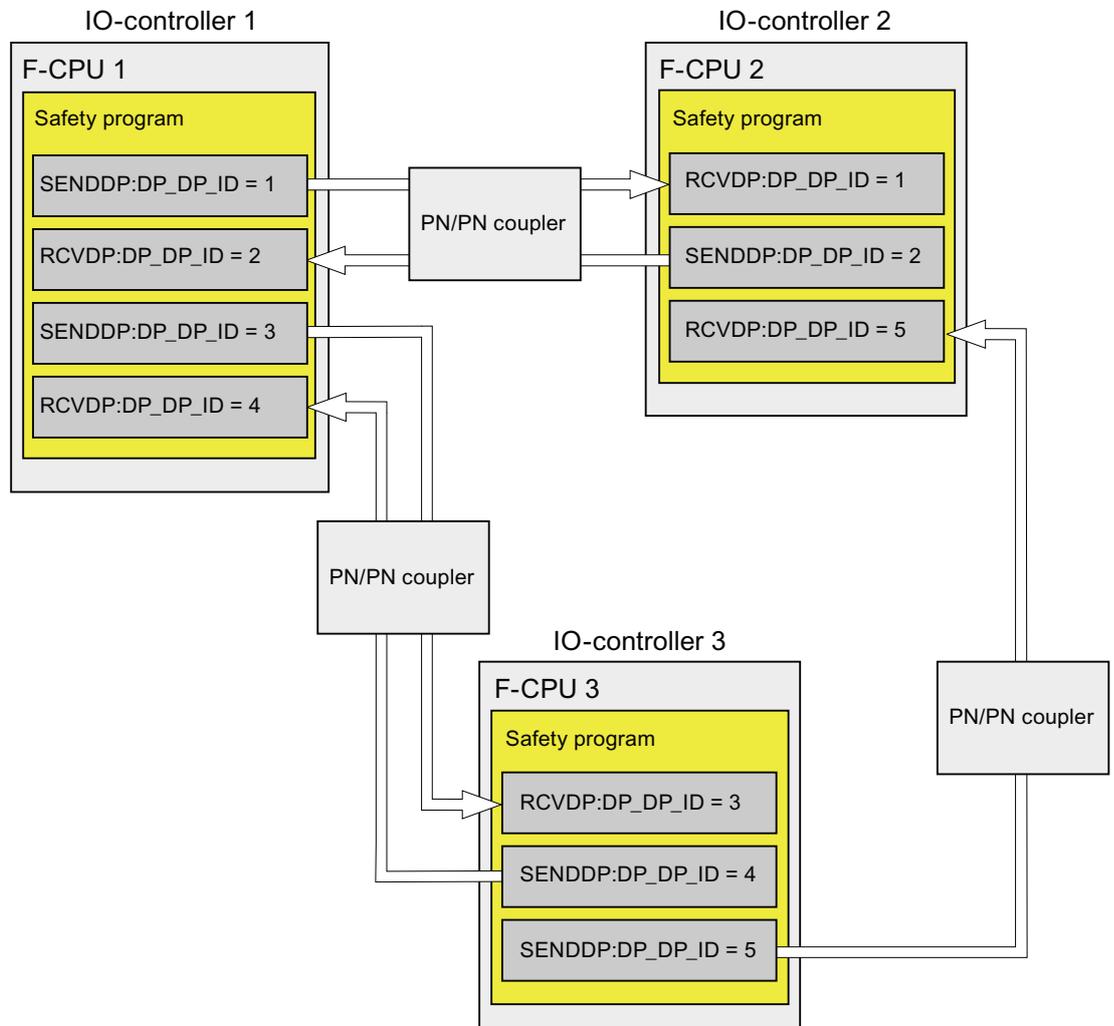
Programming procedure

You program safety-related IO Controller-IO Controller communication as follows:

1. In the safety program from which data are to be sent, call the SENDDP instruction (Page 510) for sending at the end of the main safety block.
2. In the safety program in which data are to be received, call the RCVDP instruction (Page 510) for receiving at the start of the main safety block.
3. Assign the start addresses of the output and input data address areas of the PN/PN coupler configured in the *hardware and network editor* to the respective LADDR inputs.

You must carry out this assignment for every communication connection for each of the F-CPU's involved.

- Assign the value for the respective address relationship to the DP_DP_ID inputs. This establishes the communication relationship between the SENDDP instruction in a F-CPU and the RCVDP instruction in the other F-CPU: The associated instructions receive the same value for DP_DP_ID.



WARNING

The value for each address relationship (input parameter DP_DP_ID; data type: INT) is user-defined; however, it must be unique from all other safety-related communication connections in the network. The uniqueness must be checked in the print-out of the safety program during acceptance testing of the safety program. Additional information can be found in Correctness of the communication configuration (Page 208).

You must supply inputs DP_DP_ID and LADDR with constant values when calling the instruction. Direct read or write access to the associated instance DB is not permitted in the safety program! (S016)

5. Supply the SD_BO_xx inputs of SENDDP with the send signals. To cut down on intermediate signals when transferring parameters, you can write the value directly to the instance DB of SENDDP using fully qualified access (for example, "Name SENDDP_1".SD_BO_02) before calling SENDDP.
6. Supply the RD_BO_xx outputs of RCVDP with the signals that you want to process further in other program sections or use fully qualified access to read the received signals directly in the associated instance DB in the program sections to be processed further (e.g., "Name RCVDP_1".RD_BO_02).
7. Supply the SUBBO_xx and SUBI_xx inputs of RCVDP with the fail-safe values that are to be output by RCVDP in place of the process data until communication is established for the first time after startup of the sending and receiving F-systems or in the event of an error in safety-related communication.

- Specification of constant fail-safe values:

For data of data type INT, you can enter constant fail-safe values directly as constants in the SUBI_xx input (initial value = "0"). If you want to specify a constant fail-safe value "TRUE" for data of data type BOOL, enter the "F_GLOBDB".VKE1 tag (fully qualified) in the SUBBO_xx input (initial value = "FALSE").

- Specification of dynamic fail-safe values:

If you want to specify dynamic fail-safe values, define a tag that you change dynamically through your safety program in an F-DB and specify this tag (fully qualified) in the SUBI_xx or SUBBO_xx input.

 **WARNING**

Note that your safety program for dynamically changing the tag for a dynamic fail-safe value can only be processed after the call of the RCVDP, because prior to the RCVDP call there must not be any network in the main safety block and at most there can be one other RCVDP instruction. You must therefore assign appropriate initial/actual values for these tags to be output by RCVDP in the first cycle after a startup of the F-system. (S017)

8. Configure the TIMEOUT inputs of the RCVDP and SENDDP instructions with the required monitoring time.

 **WARNING**

It can only then be ensured (from a fail-safe standpoint) that a signal level to be transferred will be captured on the sender side and transferred to the receiver if the signal level is pending for at least as long as the assigned monitoring time. (S018)

Information on calculating the monitoring times can be found in Monitoring and response times (Page 523).

9. Optional: evaluate the ACK_REQ output of the RCVDP instruction, for example, in the standard user program or on the operator control and monitoring system in order to query or to indicate whether user acknowledgment is required.
10. Supply the ACK_REI input of the RCVDP instruction with the acknowledgment signal for reintegration.

11. Optional: evaluate the SUBS_ON output of the RCVDP or SENDDP instruction, in order to query whether the RCVDP instruction is outputting the fail-safe values assigned in the SUBBO_xx and SUBI_xx inputs.
12. Optional: Evaluate the ERROR output of the RCVDP or SENDDP instruction, for example, in the standard user program or on the operator control and monitoring system in order to query or to indicate whether a communication error has occurred.
13. Optional: evaluate the SENDMODE output of the RCVDP instruction in order to query whether the F-CPU with the associated SENDDP instruction is in deactivated safety mode (Page 190).

9.2.4 Safety-related IO controller-IO controller communication - Limits for data transfer

Note

If the data quantities to be transmitted exceed the capacity of the SENDDP / RCVDP correlated instructions, a second (or third) SENDDP / RCVDP call can be used. This requires configuration of an additional connection via the PN/PN coupler. Whether or not this is possible with one single PN/PN coupler depends on the capacity restrictions of the PN/PN coupler.

9.3 Safety-related master-master communication

9.3.1 Configure safety-related master-master communication

Introduction

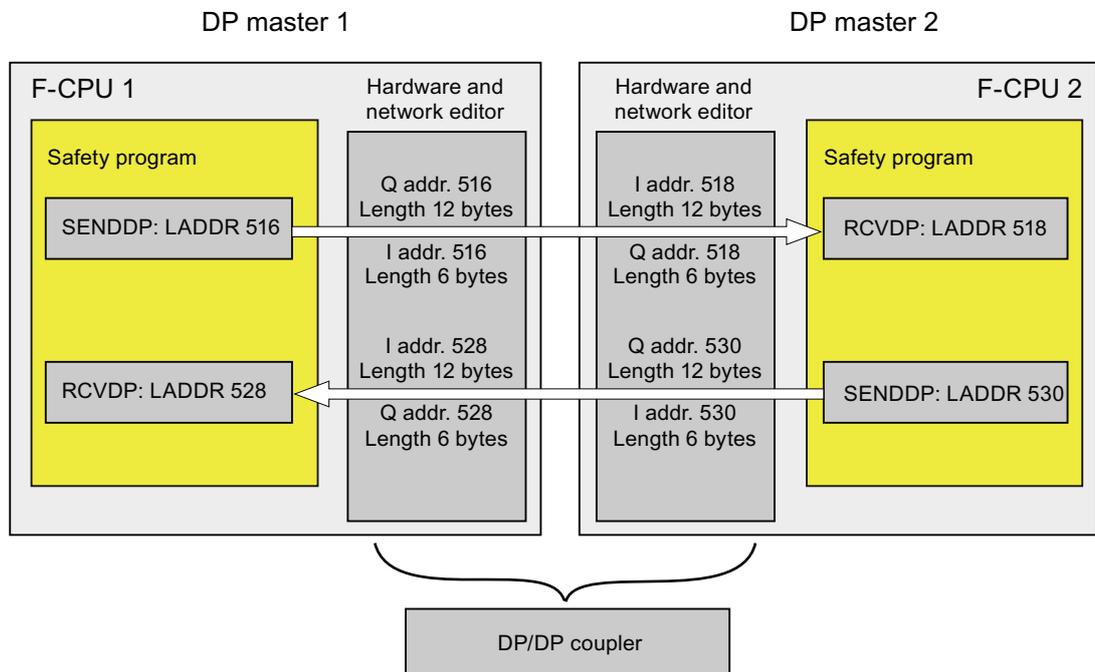
Safety-related communication between safety programs of the F-CPU's of DP masters takes place via a DP/DP coupler.

Note

Switch the data validity indicator "DIA" on the DIP switch of the DP/DP coupler to "OFF". Otherwise, safety-related CPU-CPU communication is not possible.

Configuring address areas

You must configure one address area for output data and another address area for input data in the DP/DP coupler in the *hardware and network editor* for each safety-related communication connection between two F-CPU's via a DP/DP coupler. In the figure below, both of the F-CPU's are to be able to send and receive data (bidirectional communication).



Rules for defining the address areas

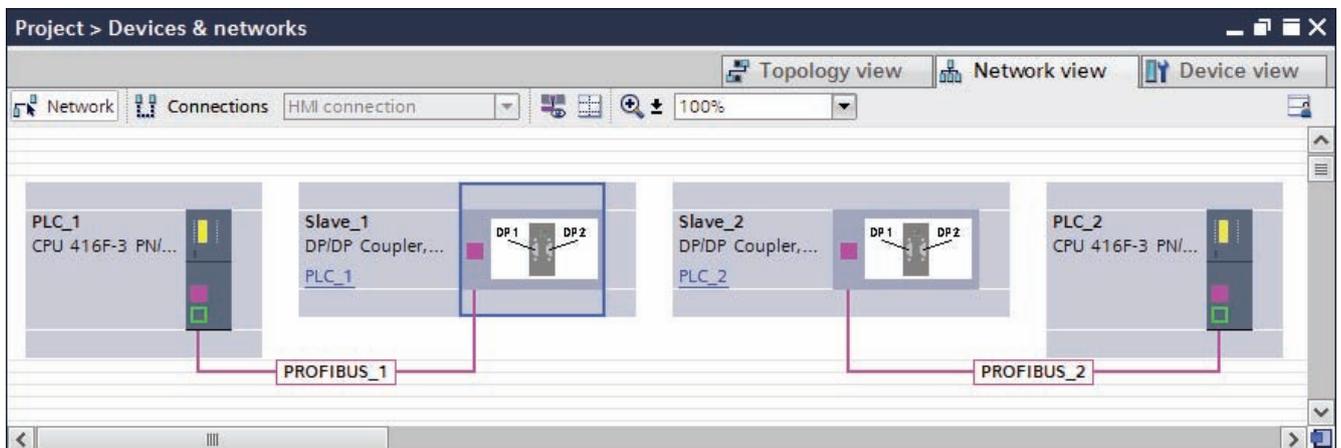
The address area for output data for the **data to be sent** must begin with the same start address as the associated address area for input data. A total of 12 bytes (consistent) is required for the address area for output data, while 6 bytes (consistent) are required for the address area for input data.

The address area for input data for the **data to be received** must begin with the same start address as the associated address area for output data. A total of 12 bytes (consistent) is required for the address area for input data, while 6 bytes (consistent) are required for the address area for output data.

Procedure for configuration

The procedure for configuring safety-related master-master communication is identical to that in the standard system. Proceed as follows:

1. Insert two F-CPUS from the "Hardware catalog" task card in the project.
2. Select the DP/DP coupler from "Other field devices\PROFIBUS DP\Gateways\Siemens AG\DP/DP Coupler" in the "Hardware catalog" task card and insert it into the network view of the hardware and network editor.
3. Insert a second DP/DP coupler.
4. Connect a DP interface of F-CPU 1 to the DP interface of a DP/DP coupler and a DP interface of F-CPU 2 to the DP interface of the other DP/DP coupler.



5. A free PROFIBUS address is assigned automatically in the properties of the DP/DP-coupler in the device view. You must set this address via a switch on the DP/DP coupler, either via the DIP switch on the device or in the configuration of the DP/DP coupler (see DP/DP Coupler (<http://support.automation.siemens.com/WW/view/en/1179382>) manual).
6. For bidirectional communication connections, i.e., each F-CPU is to send and receive data, select the following modules from the "Hardware catalog" task card (with filter activated). Insert them into the "Device overview" tab in the device view of the DP/DP coupler:
 - One "6 bytes I/12 Bytes Q consistent" module, and
 - One "12 bytes I/6 Bytes Q consistent" module

7. In the properties of the modules, assign the addresses outside the process image as follows:

For "6 bytes I/12 Bytes Q consistent" module, e.g.:

- Output addresses: Start address 516
- Input addresses: Start address 516

For "12 bytes I/6 Bytes Q consistent" module, e.g.:

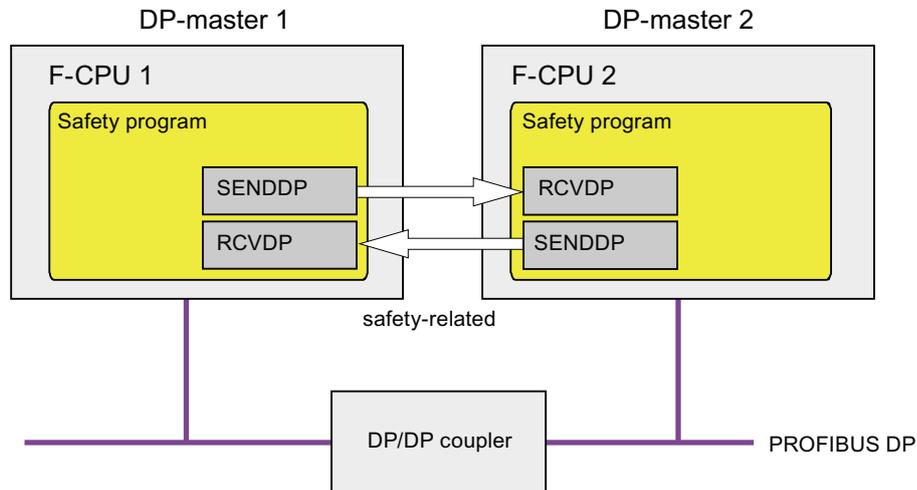
- Output addresses: Start address 528
- Input addresses: Start address 528

Device overview		Rack	Slot	I address	Q address	Type	Order no.	Firmware
Slave_1		0	0	2043*		DP/DP Coupler, Release 2	6ES7 158-0AD01-0XA0	B0
6 Bytes I/12 Bytes O consistent_1		0	1	516...521	516...527	6 Bytes I/12 Bytes O consistent		
12 Bytes I/6 Bytes O consistent_1		0	2	528...539	528...533	12 Bytes I/6 Bytes O consistent		
		0	3					
		0	4					
		0	5					
		0	6					

8. Next, define the addresses for the modules of F-CPU 2 with DP/DP coupler "Slave_2", as just described. Note that you have to adjust the addresses accordingly (see figure at the beginning of "Configuring address areas").

9.3.2 Safety-related master-master communication via SENDDP and RCVDP

Communication via SENDDP and RCVDP instructions



Safety-related communication between the F-CPU's of the DP masters makes use of the SENDDP and RCVDP instructions for sending and receiving, respectively. They can be used to transfer a *fixed* amount of fail-safe data of data types BOOL and INT in a fail-safe manner.

You will find these instructions in the "Instructions" task card under "Communication". The RCVDP instruction **must** be called at the start of the main safety block. The SENDDP instruction **must** be called at the end of the main safety block.

Note that the send signals are not sent before calling the SENDDP instruction at the end of the relevant F-runtime group execution.

A detailed description of the SENDDP and RCVDP instructions is found in SENDDP and RCVDP: Send and receive data via PROFIBUS DP/PROFINET IO (STEP 7 Safety Advanced V11) (Page 510).

9.3.3 Program safety-related master-master communication

Requirement for programming

The address areas for input and output data for the DP/DP coupler must be configured.

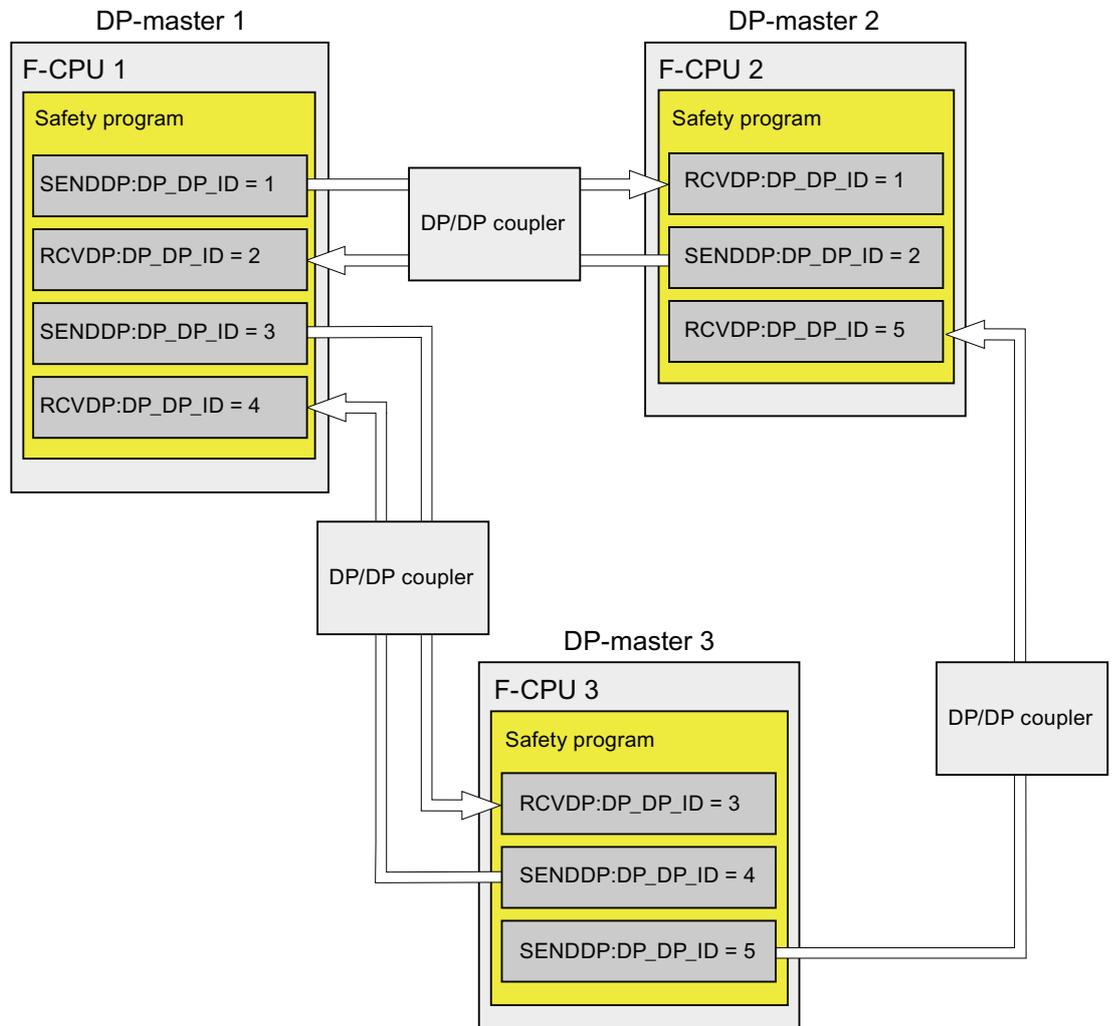
Programming procedure

You program safety-related master-master communication as follows:

1. In the safety program from which data are to be sent, call the SENDDP instruction (Page 510) for sending at the end of the main safety block.
2. In the safety program from which data are to be received, call the RCVDP instruction (Page 510) for receiving at the start of the main safety block.
3. Assign the start addresses of the address areas for output and input data of the DP/DP coupler configured in the *hardware and network editor* to the respective LADDR inputs.

You must carry out this assignment for every communication connection for each of the F-CPU's involved.

- Assign the value for the respective address relationship to the DP_DP_ID inputs. This establishes the communication relationship between the SENDDP instruction in a F-CPU and the RCVDP instruction in the other F-CPU: The associated instructions receive the same value for DP_DP_ID.



WARNING

The value for each address relationship (input parameter DP_DP_ID; data type: INT) is user-defined; however, it must be unique from all other safety-related communication connections in the network. The uniqueness must be checked in the print-out of the safety program during acceptance testing of the safety program. Additional information can be found in Correctness of the communication configuration (Page 208).

You must supply inputs DP_DP_ID and LADDR with constant values when calling the instruction. Direct read or write access to the associated instance DB is not permitted in the safety program! (S016)

5. Supply the SD_BO_xx inputs of SENDDP with the send signals. To cut down on intermediate signals when transferring parameters, you can write the value directly to the instance DB of SENDDP using fully qualified access (for example, "Name SENDDP_1".SD_BO_02) before calling SENDDP.
6. Supply the RD_BO_xx outputs of RCVDP with the signals that you want to process further in other program sections or use fully qualified access to read the received signals directly in the associated instance DB in the program sections to be processed further (for example, "Name RCVDP1".RD_BO_02).
7. Supply the SUBBO_xx and SUBI_xx inputs of RCVDP with the fail-safe values that are to be output by RCVDP in place of the process data until communication is established for the first time after startup of the sending and receiving F-systems or in the event of an error in safety-related communication.

- Specification of constant fail-safe values:

For data of data type INT, you can enter constant fail-safe values directly as constants in the SUBI_xx input (initial value = "0"). If you want to specify a constant fail-safe value for data of data type BOOL, enter the "F_GLOBDB".VKE1 tag (fully qualified) in the SUBBO_xx input (initial value = "FALSE").

- Specification of dynamic fail-safe values:

If you want to specify dynamic fail-safe values, define a tag that you change dynamically through your safety program in an F-DB and specify this tag (fully qualified) in the SUBI_xx or SUBBO_xx input.

 **WARNING**

Note that your safety program for dynamically changing the tag for a dynamic fail-safe value can only be processed after the call of the RCVDP, because prior to the RCVDP call there must not be any network in the main safety block and at most there can be one other RCVDP instruction. You must therefore assign appropriate initial/actual values for these tags to be output by RCVDP in the first cycle after a startup of the F-system. (S017)

8. Configure the TIMEOUT inputs of the RCVDP and SENDDP instructions with the required monitoring time.

 **WARNING**

It can only then be ensured (from a fail-safe standpoint) that a signal level to be transferred will be captured on the sender side and transferred to the receiver if the signal level is pending for at least as long as the assigned monitoring time. (S018)

Information on calculating the monitoring times can be found in Monitoring and response times (Page 523).

9. Optional: evaluate the ACK_REQ output of the RCVDP instruction, for example, in the standard user program or on the operator control and monitoring system in order to query or to indicate whether user acknowledgment is required.
10. Supply the ACK_REI input of the RCVDP instruction with the acknowledgment signal for reintegration.

11. Optional: evaluate the SUBS_ON output of the RCVDP or SENDDP instruction, in order to query whether the RCVDP instruction is outputting the fail-safe values assigned in the SUBBO_xx and SUBI_xx inputs.
12. Optional: evaluate the ERROR output of the RCVDP or SENDDP instruction, for example, in the standard user program or on the operator control and monitoring system in order to query or to indicate whether a communication error has occurred.
13. Optional: evaluate the SENDMODE output of the RCVDP instruction in order to query whether the F-CPU with the associated SENDDP instruction is in deactivated safety mode (Page 190).

9.3.4 Safety-related master-master communication: Limits for data transfer

Note

If the data quantities to be transmitted exceed the capacity of the SENDDP / RCVDP correlated instructions, a second (or third) SENDDP / RCVDP call can be used. This requires configuration of an additional connection via the DP/DP coupler. Whether or not this is possible with one single DP/DP coupler depends on the capacity restrictions of the DP/DP coupler.

9.4 Safety-related communication between I/O-controller and I-device

9.4.1 Configuring safety-related communication between I/O-controller and I-device

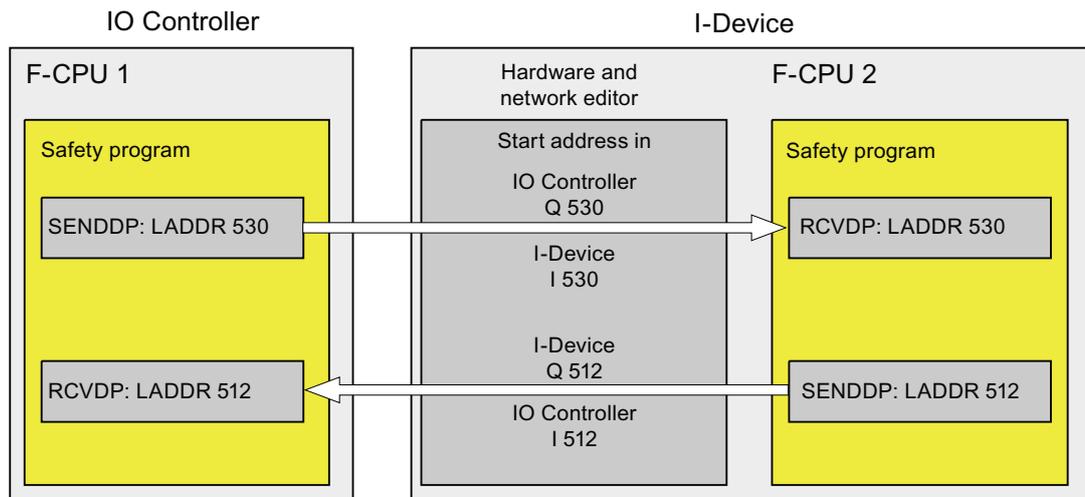
Introduction

Safety-related communication between the safety program of the F-CPU of an IO Controller and the safety program(s) of the F-CPU(s) of one or more I-Devices takes place over IO Controller-I-Device connections (**F-CD**) in PROFINET IO, same as in standard systems.

You do not need any additional hardware for IO Controller-I-Device communication.

Configuring transfer areas

For every safety-related communication connection between two F-CPU's, you must configure transfer areas in the *hardware and network editor*. In the figure below, both of the F-CPU's are to be able to send and receive data (bidirectional communication).



You assign the start addresses of the transfer areas to the LADDR parameter of the corresponding SENDDP and RCVDP instructions in the safety programs.

Procedure for configuration

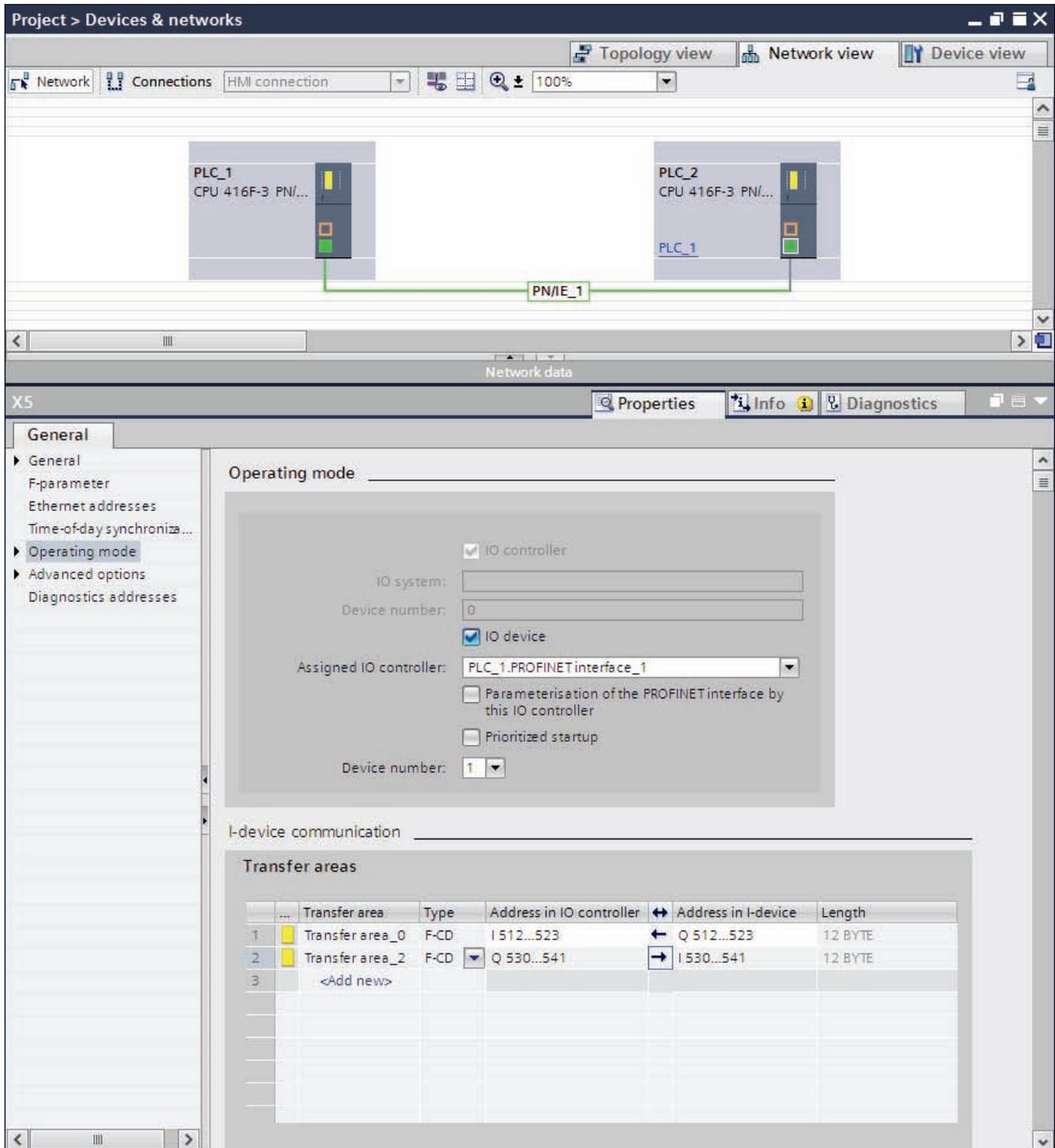
The procedure for configuring safety-related IO Controller/IO Controller communication is identical to that in the standard system. Proceed as follows:

1. Insert two F-CPUS from the "Hardware catalog" task card in the project.
2. Activate the "IO Device" mode for F-CPU 2 in the properties of its PN interface and assign this PN interface to a PN interface of F-CPU 1.
3. Select the PROFINET interface of F-CPU 2. Under "Transfer areas", you create an F-CD connection (type "F-CD") for sending to the IO controller (←). The F-CD connection is shown in yellow in the table and the address areas in the I-Device and IO Controller assigned outside of the process image are displayed.

In addition, an acknowledgement connection is created automatically for each F-CD connection. (see "Transfer area details").

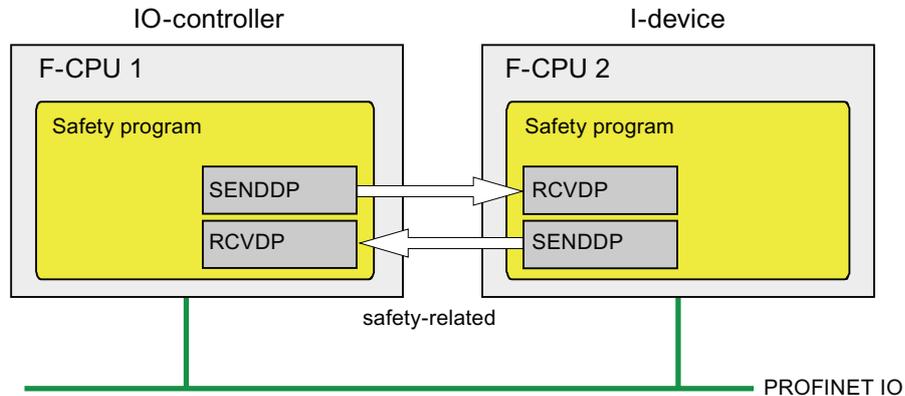
4. Create an additional F-CD connection for receiving from the IO Controller.

- In the transfer area you just created, click the arrow in order to change the transfer direction to receiving from IO Controller (→).



9.4.2 Safety-related IO controller-I/O device communication via SENDDP and RCVDP

Communication via SENDDP and RCVDP instructions



Safety-related communication between the F-CPU's of the IO Controller and an I-Device makes use of the SENDDP and RCVDP instructions for sending and receiving, respectively. They can be used to transfer a *fixed* amount of fail-safe data of data types BOOL and INT in a fail-safe manner.

You will find these instructions in the "Instructions" task card under "Communication". The RCVDP instruction **must** be called at the start of the main safety block. The SENDDP instruction **must** be called at the end of the main safety block.

Note that the send signals are sent only after calling the SENDDP instruction at the end of the relevant F-runtime group execution.

A detailed description of the SENDDP and RCVDP instructions is found in SENDDP and RCVDP: Send and receive data via PROFIBUS DP/PROFINET IO (STEP 7 Safety Advanced V11) (Page 510).

9.4.3 Programming safety-related IO controller I-device communication

Requirement for programming

The transfer areas must be configured.

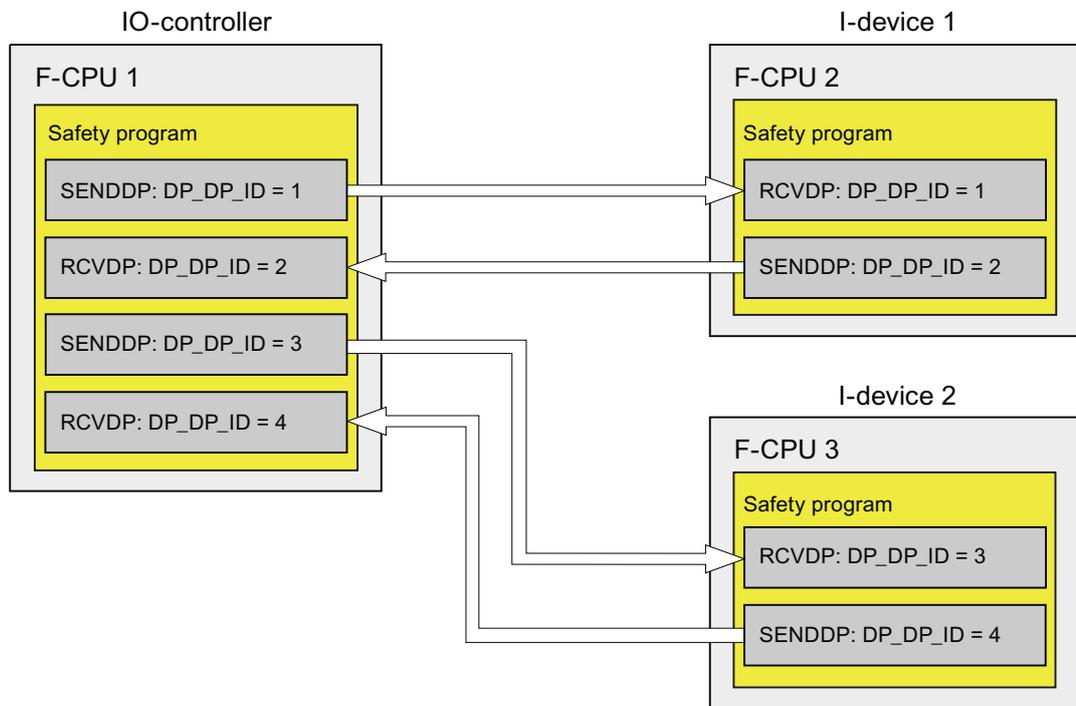
Programming procedure

The procedure for programming safety-related IO-Controller-I-Device communication is the same as for programming safety-related IO-Controller-IO-Controller communication (see Program safety-related IO controller-IO controller communication (Page 118)).

The assignment of the start addresses of the transfer areas for the LADDR parameter of the SENDDP/RCVDP instructions can be obtained from the following table.

Instruction	Start address LADDR	
	From row	From column
SENDDP in the IO Controller	→	Address in the IO Controller
RCVDP in the IO Controller	←	Address in the IO Controller
SENDDP in the I-Device	←	Address in the IO Device
RCVDP in the I-Device	→	Address in the IO Device

The figure below contains an example of how to specify the address relationships for the inputs of the SENDDP and RCVDP instructions for four safety-related IO-Controller-I-Device communication relationships.



 WARNING
The value for each address relationship (input parameter DP_DP_ID; data type: INT) is user-defined; however, it must be unique from all other safety-related communication connections in the network. The uniqueness must be checked in the print-out of the safety program during acceptance testing of the safety program. Additional information can be found in Correctness of the communication configuration (Page 208).
You must supply inputs DP_DP_ID and LADDR with constant values when calling the instruction. Direct read or write access to the associated instance DB is not permitted in the safety program! (S016)

 WARNING
It can only be ensured (from a fail-safe standpoint) that a signal level to be transferred will be captured on the sender side and transferred to the receiver if the signal level is pending for at least as long as the assigned monitoring time. (S018)

Information on calculating the monitoring times can be found in Monitoring and response times (Page 523).

9.4.4 Safety-related IO-Controller-IO-Device communication - Limits for data transfer

Limits for data transfer

If the amount of data to be transferred is greater than the capacity of the related SENDDP/RCVDP instructions, you can use additional SENDDP/RCVDP instructions. Configure additional transfer areas for this purpose. Remember the maximum limit of 1440 bytes of input data or 1440 bytes of output data for transfer between an I-Device and a IO Controller.

The following table shows the amount of output and input data assigned in safety-related communication connections:

Safety-related communication	Communication connection	Assigned input and output data			
		In the IO Controller		In the I-Device	
		Output data	Input data	Output data	Input data
IO Controller-I-Device	Sending: I-Device 1 to IO Controller	6 bytes	12 bytes	12 bytes	6 bytes
	Receiving: I-Device 1 from IO Controller	12 bytes	6 bytes	6 bytes	12 bytes

Consider all additional configured safety-related and standard communication connections (transfer areas of type F-CD and CD) for the maximum limit of 1440 bytes of input data or 1440 bytes of output data for transfer between an I-Device and an IO Controller . In addition, data are assigned for internal purposes such that the maximum limit may be reached sooner.

When the limit is exceeded, a corresponding error message is displayed.

9.5 Safety-related master-I-slave communication

9.5.1 Configuring safety-related master-I-slave communication

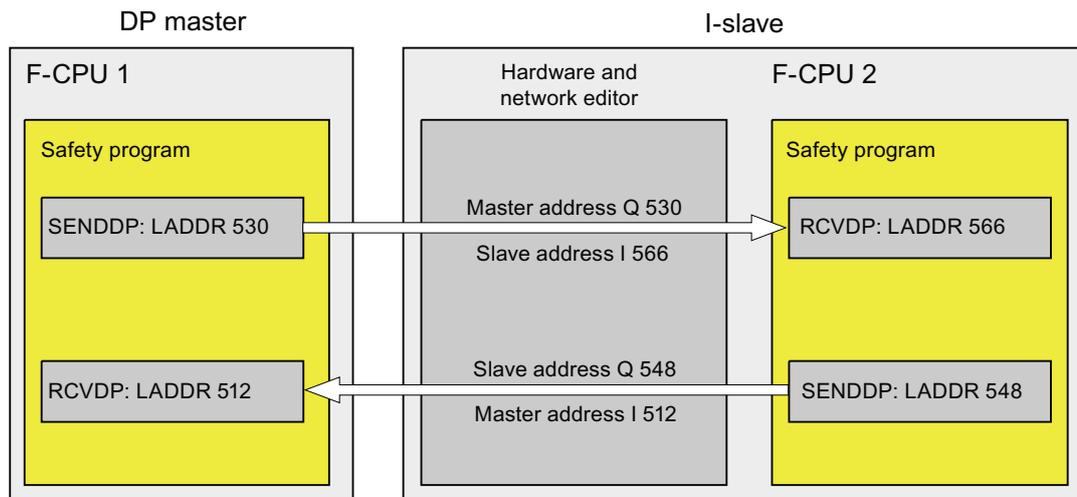
Introduction

Safety-related communication between the safety program of the F-CPU of a DP master and the safety program(s) of the F-CPU(s) of one or more I-slaves takes place over master-I-slave connections (**F-MS**), as in standard systems.

You do not need any additional hardware for the master-I-slave communication.

Configuring transfer areas

For every safety-related communication connection between two F-CPU's, you must configure transfer areas in the *hardware and network editor*. In the figure below, both of the F-CPU's are to be able to send and receive data (bidirectional communication).



You assign the start addresses of the transfer areas to the LADDR parameter of the corresponding SENDDP and RCVDP instructions in the safety programs.

Procedure for configuration

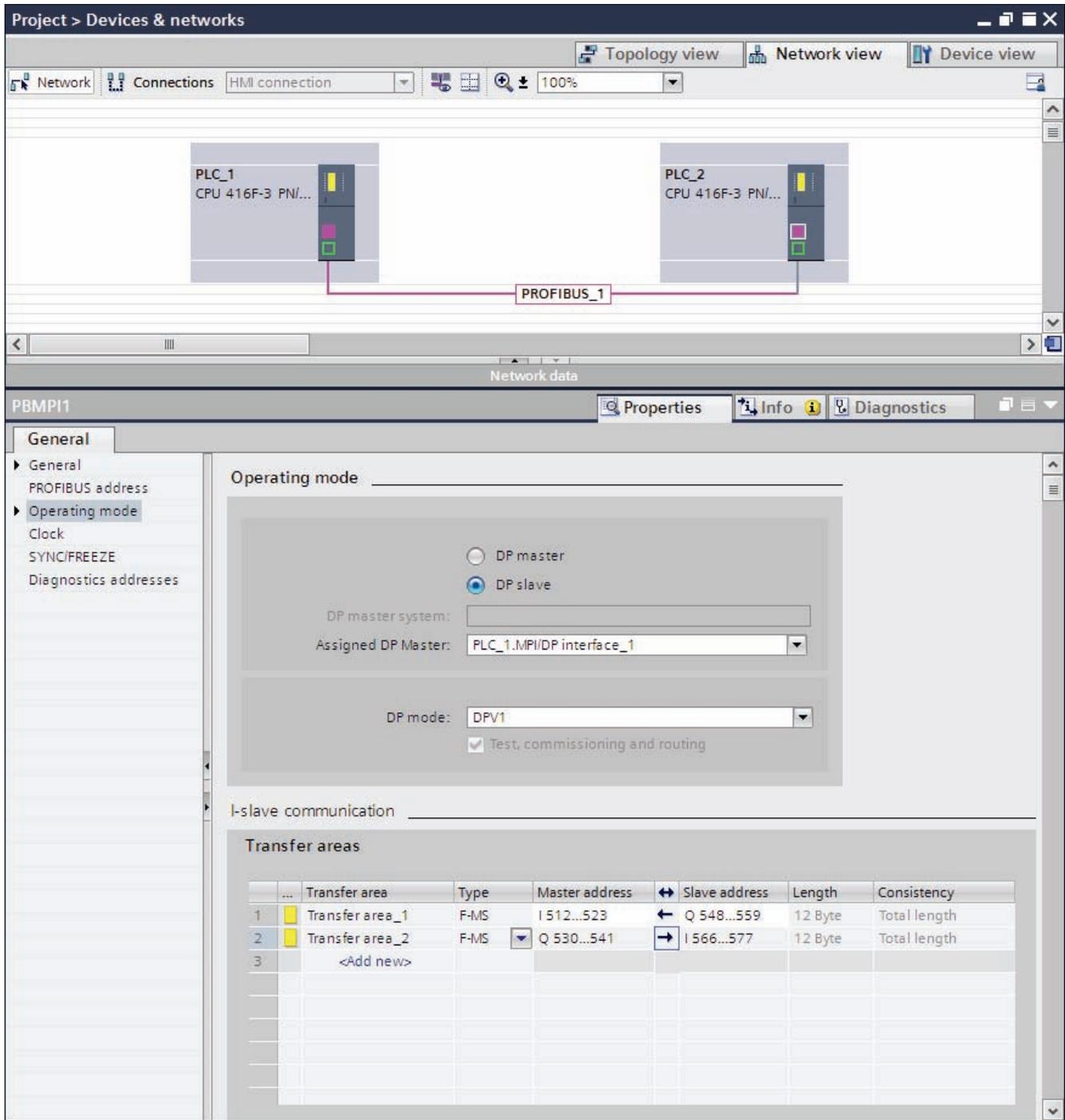
The procedure for configuring safety-related master-I-slave communication is identical to that in the standard system. Proceed as follows:

1. Insert two F-CPU from the "Hardware catalog" task card in the project.
2. Activate the "DP slave" mode (I-slave) for F-CPU 2 in the properties of its DP interface and assign this DP interface to a DP interface of F-CPU 1.
3. Select the PROFIBUS interface of F-CPU 2. Under "Transfer areas", you create an F-MS connection (type "F-MS") for sending to the DP master (←). The F-MS connection is shown in yellow in the table and the address areas in the I-slave and DP master assigned outside of the process image are displayed.

In addition, an acknowledgement connection is created automatically for each F-MS connection. (see "Transfer area details").

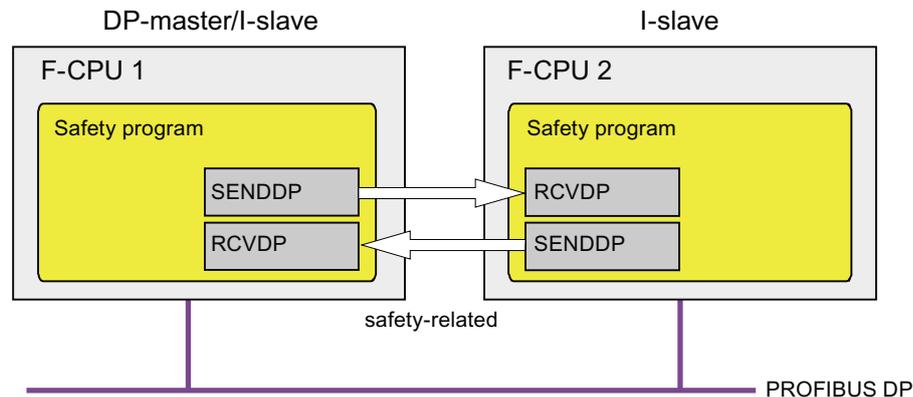
4. Create an additional F-MS connection for receiving from the DP master.

- In the transfer area you just created, click the arrow in order to change the transfer direction to receiving from DP master (→).



9.5.2 Safety-related master-I-slave or I-slave-I-slave communication via SENDDP and RCVDP

Communication via SENDDP and RCVDP instructions



Safety-related communication between the F-CPU of the DP master and an I-slave or between the F-CPU of several I-slaves makes use of the SENDDP and RCVDP instructions for sending and receiving, respectively. They can be used to transfer a *fixed* amount of fail-safe data of data types BOOL and INT in a fail-safe manner.

You will find these instructions in the "Instructions" task card under "Communication". The RCVDP instruction **must** be called at the start of the main safety block. The SENDDP instruction **must** be called at the end of the main safety block.

Note that the send signals are sent only after calling the SENDDP instruction at the end of the relevant F-runtime group execution.

A detailed description of the SENDDP and RCVDP instructions is found in SENDDP and RCVDP: Send and receive data via PROFIBUS DP/PROFINET IO (STEP 7 Safety Advanced V11) (Page 510).

9.5.3 Program the safety-related master-I-slave or I-slave-I-slave communication

Requirements

The transfer areas must be configured.

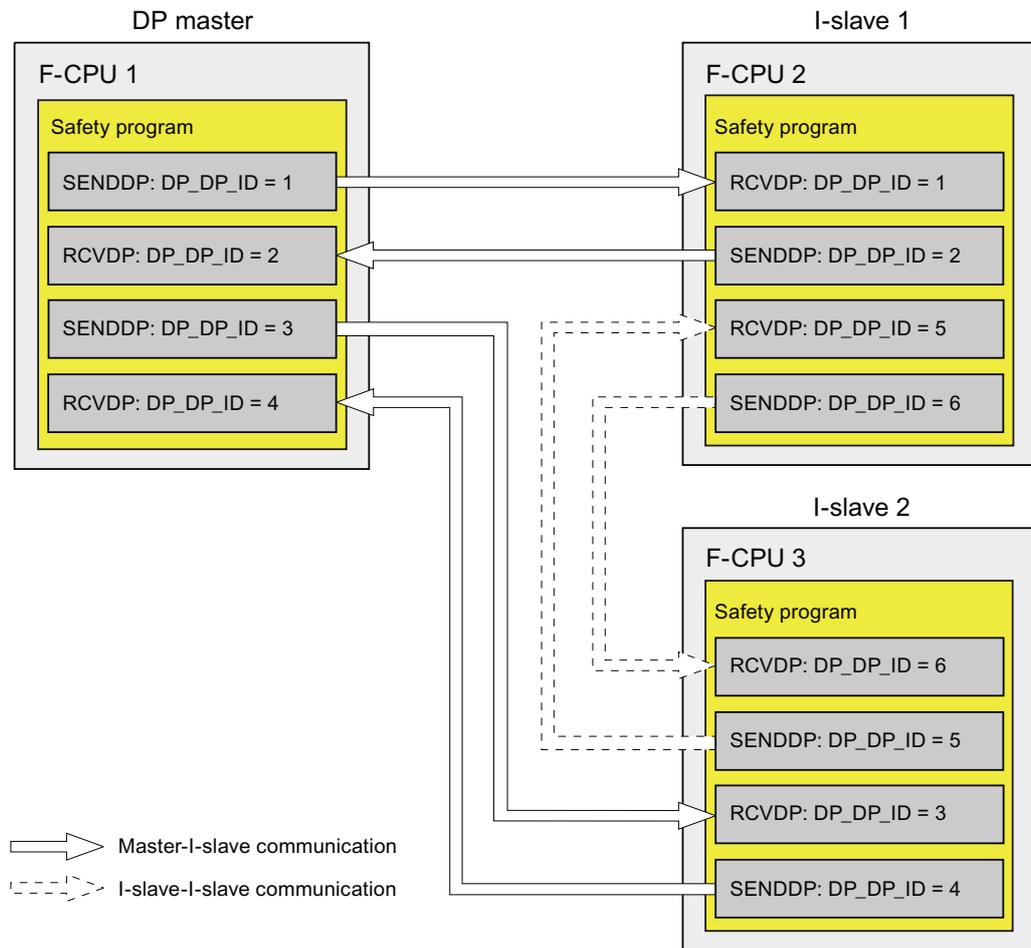
Programming procedure

The procedure for programming safety-related master-I-slave communication or I-slave-I-slave communication is the same as for programming safety-related master-master communication (see Program safety-related master-master communication (Page 126)).

The assignment of the start addresses of the transfer areas for the LADDR parameter of the SENDDP/RCVDP instructions can be obtained from the following table.

Instruction	Start address LADDR	
	From row	From column
SENDDP in the DP master	→	Master address
RCVDP in the DP master	←	Master address
SENDDP in the I-slave	←	Slave address
RCVDP in the I-slave	→	Slave address

The figure below contains an example of how to specify the address relationships at the inputs of SENDDP and RCVDP instructions for two safety-related master-I-slave and one I-slave-I-slave communications.



WARNING

The value for each address relationship (input parameter DP_DP_ID; data type: INT) is user-defined; however, it must be unique from all other safety-related communication connections in the network. The uniqueness must be checked in the print-out of the safety program during acceptance testing of the safety program. Additional information can be found in Correctness of the communication configuration (Page 208).

You must supply inputs DP_DP_ID and LADDR with constant values when calling the instruction. Direct read or write access to the associated instance DB is not permitted in the safety program! (S016)

WARNING

It can only then be ensured (from a fail-safe standpoint) that a signal level to be transferred will be captured on the sender side and transferred to the receiver if the signal level is pending for at least as long as the assigned monitoring time. (S018)

Information on calculating the monitoring times can be found in Monitoring and response times (Page 523).

9.5.4 Limits for data transfer of safety-related master-I-slave or I-slave-I-slave communication

Limits for data transfer

If the amount of data to be transferred is greater than the capacity of related SENDDP/RCVDP instructions, you can use additional SENDDP/RCVDP instructions. Configure additional transfer areas for this purpose. Note the maximum limit of 244 bytes of input data or 244 bytes of output data for transfer between an I-slave and a DP master.

The following table shows the amount of output and input data assigned in safety-related communication connections:

Safety-related communication	Communication connection	Assigned input and output data					
		DP master		I-slave 1		I-slave 2	
		Output data	Input data	Output data	Input data	Output data	Input data
Master-I-slave	Sending: I-slave 1 to DP master	6 bytes	12 bytes	12 bytes	6 bytes	—	—
	Receiving: I-slave 1 from DP master	12 bytes	6 bytes	6 bytes	12 bytes	—	—
I-slave-I-slave	Sending: I-slave 1 to I-slave 2	—	18 bytes	12 bytes	6 bytes	6 bytes	12 bytes
	Receiving: I-slave 1 from I-slave 2	—	18 bytes	6 bytes	12 bytes	12 bytes	6 bytes

Consider all additional configured safety-related and standard communication connections (transfer areas of type F-MS-, F-DX-, F-DX-Mod., MS-, DX- and DX-Mod) for the maximum limit of 244 bytes of input data or 244 bytes of output data for transfer between an I-Device and a DP master . If the maximum limit of 244 bytes of input data or 244 bytes of output data is exceeded, you will receive a corresponding error message.

9.6 Safety-related I-slave-I-slave communication

9.6.1 Configure safety-related I-slave-I-slave communication

Introduction

Safety-related communication between the safety program of the F-CPU's of I-slaves takes place using direct data exchange (F-DX) – same as in standard programs.

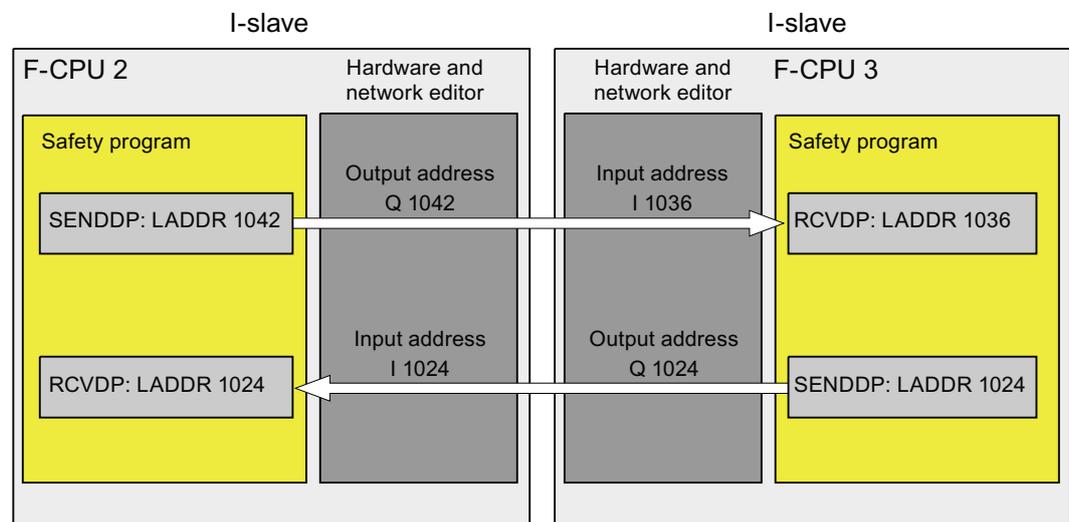
You do not need any additional hardware for I-slave-I-slave communication.

I-slave-I-slave communication is also possible:

- when the assigned DP master is a standard CPU, if the standard CPU supports direct data exchange
- when instead of a DP master, an IO Controller is networked with the I-slaves via an IE/PB link

Configuring transfer areas

For every safety-related communication connection between two I-slaves, you must configure transfer areas in the *hardware and network editor*. In the figure below, both of the I-slaves are to be able to send and receive data (bidirectional communication).



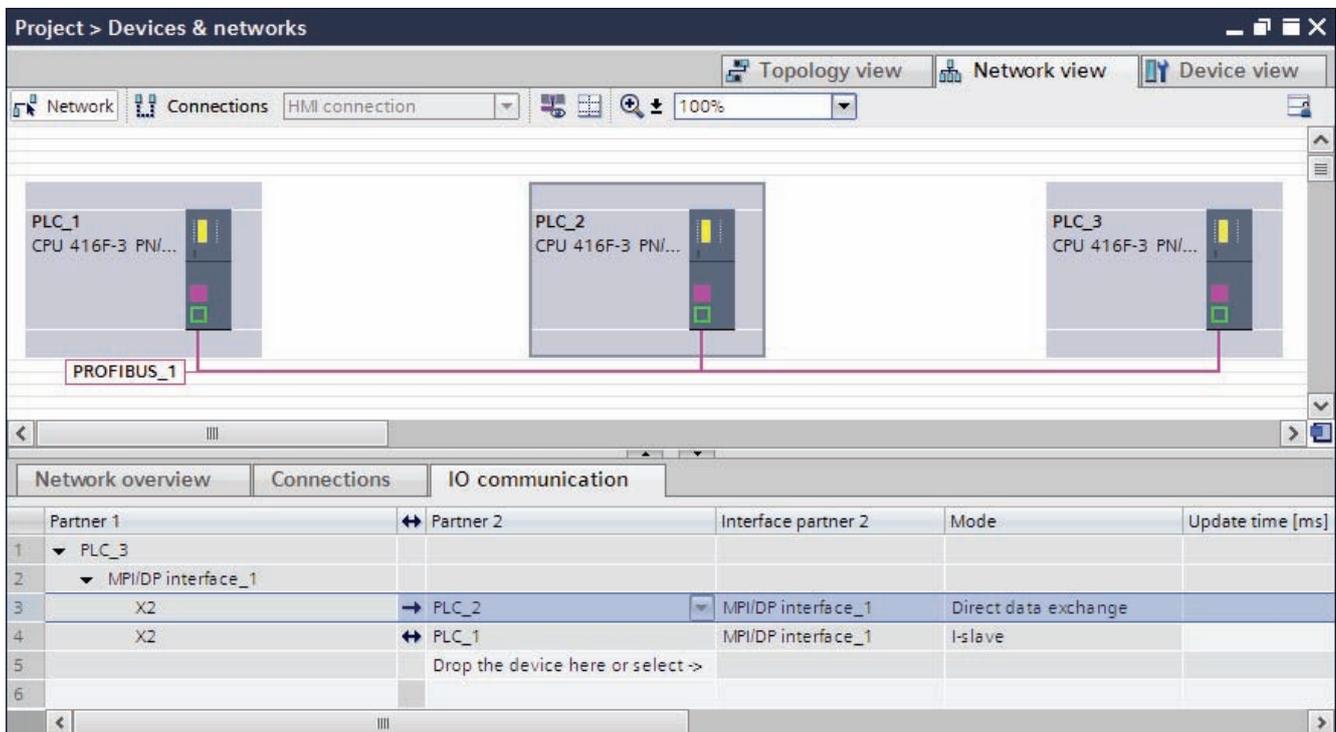
You assign the start addresses of the transfer areas to the LADDR parameter of the corresponding SENDDP and RCVDP instructions in the safety programs.

Procedure for configuration

The procedure for configuring safety-related I-slave-I-slave communication is identical to that in the standard system. Proceed as follows:

1. Insert three F-CPUS from the "Hardware catalog" task card in the project.
2. Activate "DP slaves" mode (I-slave) for F-CPU 2 and F-CPU 3 in the properties of their DP interfaces and assign these DP interfaces to a DP interface of F-CPU 1.
3. Select the DP interface of F-CPU 3 in the network view.
4. Select the "I/O communication" tab.
5. Use a drag-and-drop operation in the network view to move F-CPU 2 to the "Partner 2" column on the "I/O-communication" tab.

This creates a line with "Direct data exchange" mode for sending to the I-slave (F-CPU 2) (→).

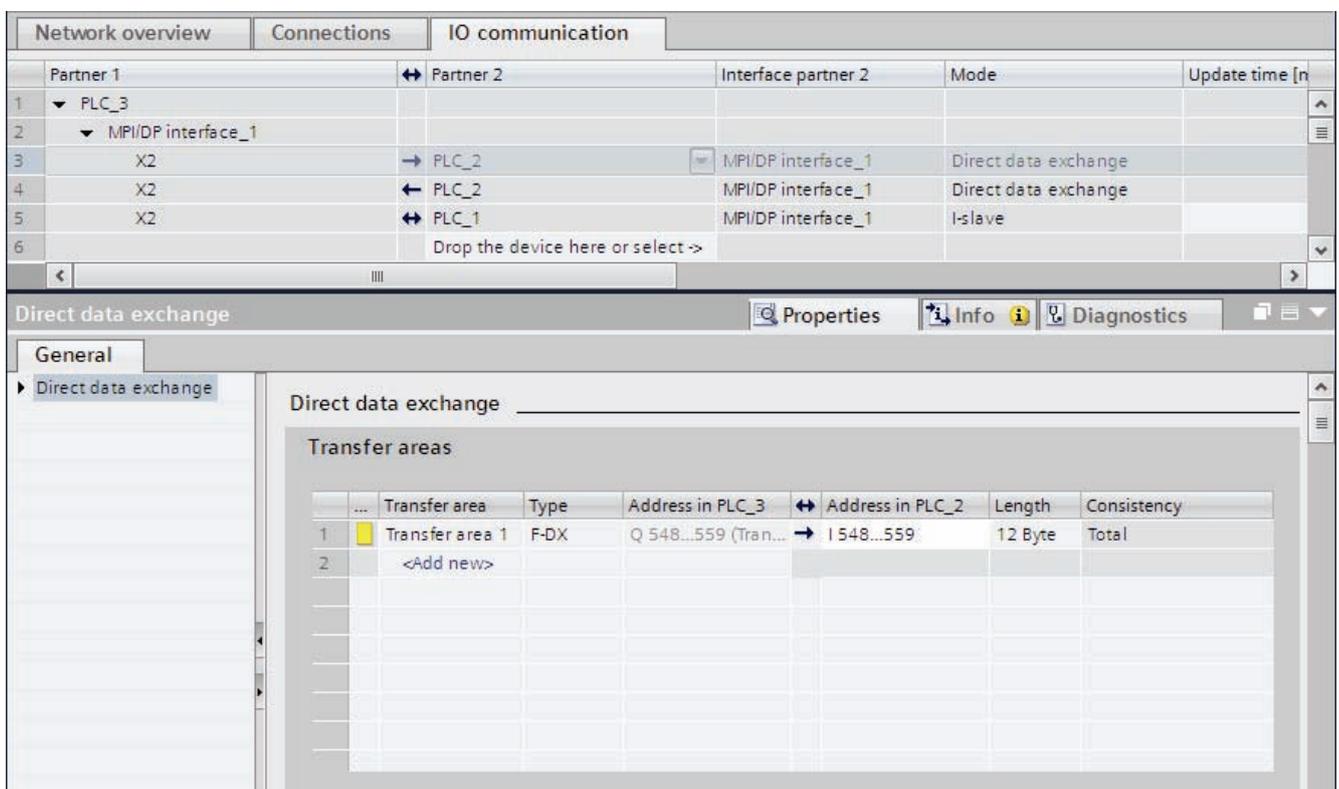


6. In "Transfer areas" ("Direct data exchange" table), create an F-DX connection (type "F-DX") for sending to the I-slave (F-CPU 2) (→). The F-DX connection is shown in yellow in the table and the address areas in the I-slaves assigned outside of the process image are displayed.

In addition, a line with "Direct data exchange" mode for receiving from the I-slave (F-CPU 2) (→) is created automatically in the "I/O communication" tab, and an acknowledgment connection (←, transfer area x_Ack) is created automatically in the associated "Direct data exchange" table.

In the "I-slave communication table" of both I-slaves, two transfer areas (type F-MS) for the master CPU are created (disabled in display).

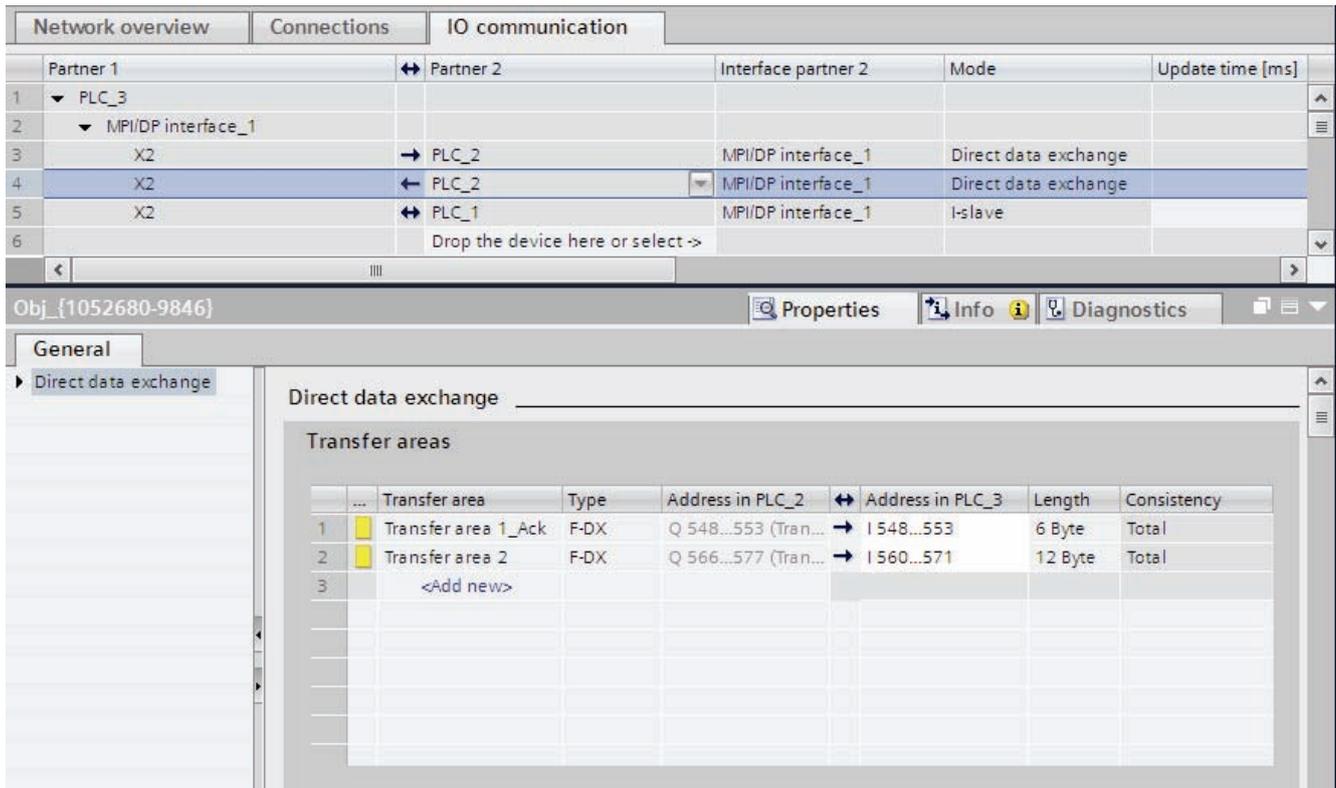
This completes the configuration for sending to F-CPU 2.



7. In the "I/O communication" tab, select the automatically created line with "Direct data exchange" mode for receiving from the I-slave (F-CPU 3) (←).
8. In "Transfer areas" ("Direct data exchange" table), create another F-DX connection for receiving from the I-slave (F-CPU 3).

In this case, as well, an acknowledgment connection (←, transfer area x_Ack) is created automatically in the "Data exchange table" and two transfer areas (type F-MS) for the master CPU (disabled in display) are created in the "I-slave communication" table of both I-slaves.

This completes the configuration for receiving from F-CPU 2.



Changing disabled local address areas of the transfer areas

In order to change the disabled local address area of "Transfer area x", you must change the address area of the corresponding acknowledgment connection "Transfer area x_Ack".

1. In "I/O communication", select the line with the arrow pointing in the same direction as the arrow of the "Transfer area x" in the "Direct data exchange" table.
2. In the "Direct data exchange" table, select the line with "Transfer area x_Ack".
3. Change the address area there.

9.6.2 Safety-related I-slave-I-slave communication via SENDDP and RCVDP

Reference

The description of the communication via SENDDP and RCVDP for safety-related I-slave-I-slave communication can be found in Safety-related master-I-slave or I-slave-I-slave communication via SENDDP and RCVDP (Page 139).

9.6.3 Programming safety-related I-slave-I-slave communication

Reference

The description of the programming of safety-related I-slave-I-slave communication can be found in Program the safety-related master-I-slave or I-slave-I-slave communication (Page 140).

The assignment of the start addresses of the transfer areas for the LADDR parameter of the SENDDP/RCVDP instructions can be obtained from the following table.

Instruction	Start address LADDR	
	From row	From column
SENDDP in the DP master	→	Master address
RCVDP in the DP master	←	Master address
SENDDP in the I-slave	←	Slave address
RCVDP in the I-slave	→	Slave address

9.6.4 Limits for data transfer of safety-related I-slave-I-slave communication

Limits for data transfer

The description of the limits for the data transfer of safety-related I-slave-I-slave communication can be found in Limits for data transfer of safety-related master-I-slave or I-slave-I-slave communication (Page 142).

9.7 Safety-Related I-Slave-Slave Communication

9.7.1 Configuring Safety-Related I-Slave-Slave Communication

Introduction

Safety-related communication between the safety program of the F-CPU of an I-slave and F-I/O in a DP slave takes place using direct data exchange (**F-DX-Mod**), same as in standard programs.

You do not need any additional hardware for I-slave-slave communication.

I-slave-slave communication is also possible:

- when the assigned DP master is a standard CPU, if the standard CPU supports direct data exchange
- when instead of a DP master, an IO Controller is networked with the I-slaves via an IE/PB link

An F-I/O DB is automatically generated for each F-I/O when it is configured in the *hardware and network editor*; this is required for the F-I/O access via safety-related I-slave-slave communication. The F-I/O DB is initially created in the safety program of the DP master, provided it is an F-CPU with F-activation. Only with the setup of the F-DX-Mod connection is the F-I/O DB created in the safety program of the I-slave.

The process input image is used to access the channels of the F-I/O in the safety program of the F-CPU of the I-slave (see description in Safety-Related I-Slave-Slave Communication - F-I/O Access (Page 153)).

Restrictions

Note

Safety-related I-slave-slave communication is possible for F-I/O in a DP slave that supports safety-related I-slave-slave communication, e.g., for all ET 200S F-modules with IM 151-x HIGH FEATURE and all S7-300 fail-safe signal modules with IM 153-2, as of order no. 6ES7153-2BA01-0XB0, firmware > V4.0.0.

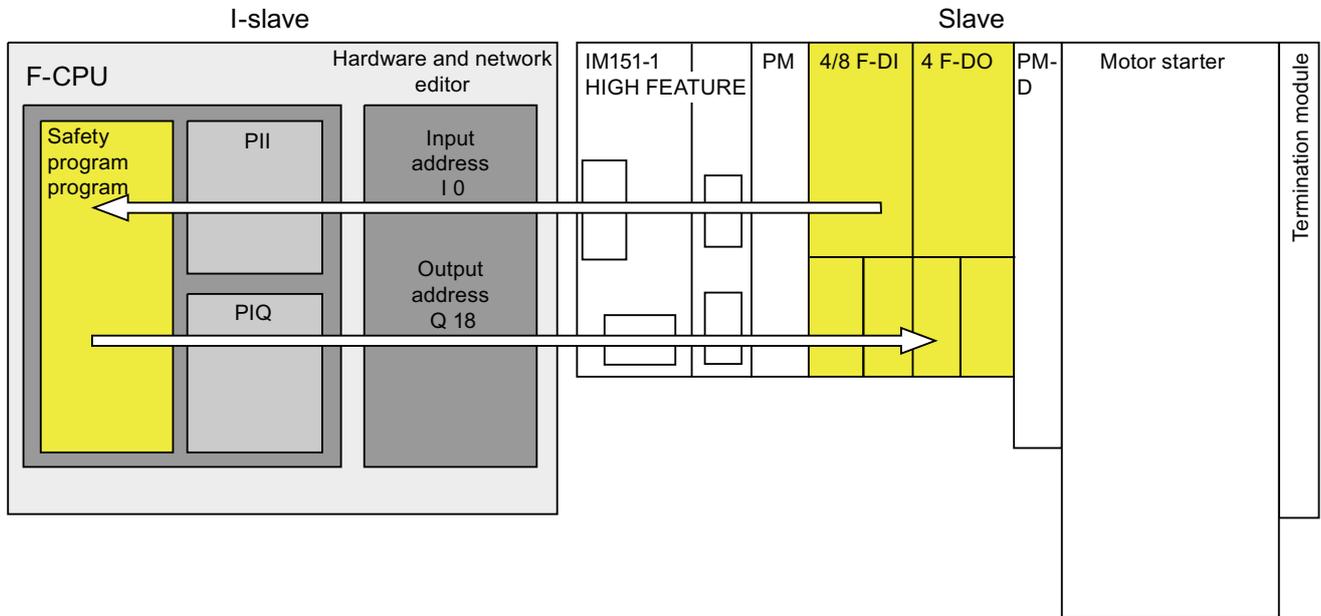
Note

With safety-related I-slave-slave communication, make sure that the CPU of the DP master is powered up before the F-CPU of the I-slave.

Otherwise, depending on the F-monitoring time specified for the F-I/O, the F-system can detect an error in safety-related communication (communication error) between the F-CPU and the F-I/O assigned to the I-slave. This means, after startup of the F-system, the F-I/O are not reintegrated automatically. Rather, they are only reintegrated after a user acknowledgment with a positive edge in the ACK_REI tag of the F-I/O DB (see also after communication errors (Page 92) and After startup of F-system (Page 90)).

Configuring transfer areas

For every safety-related communication connection between an I-slave and slave, you must configure transfer areas in the *hardware and network editor*.

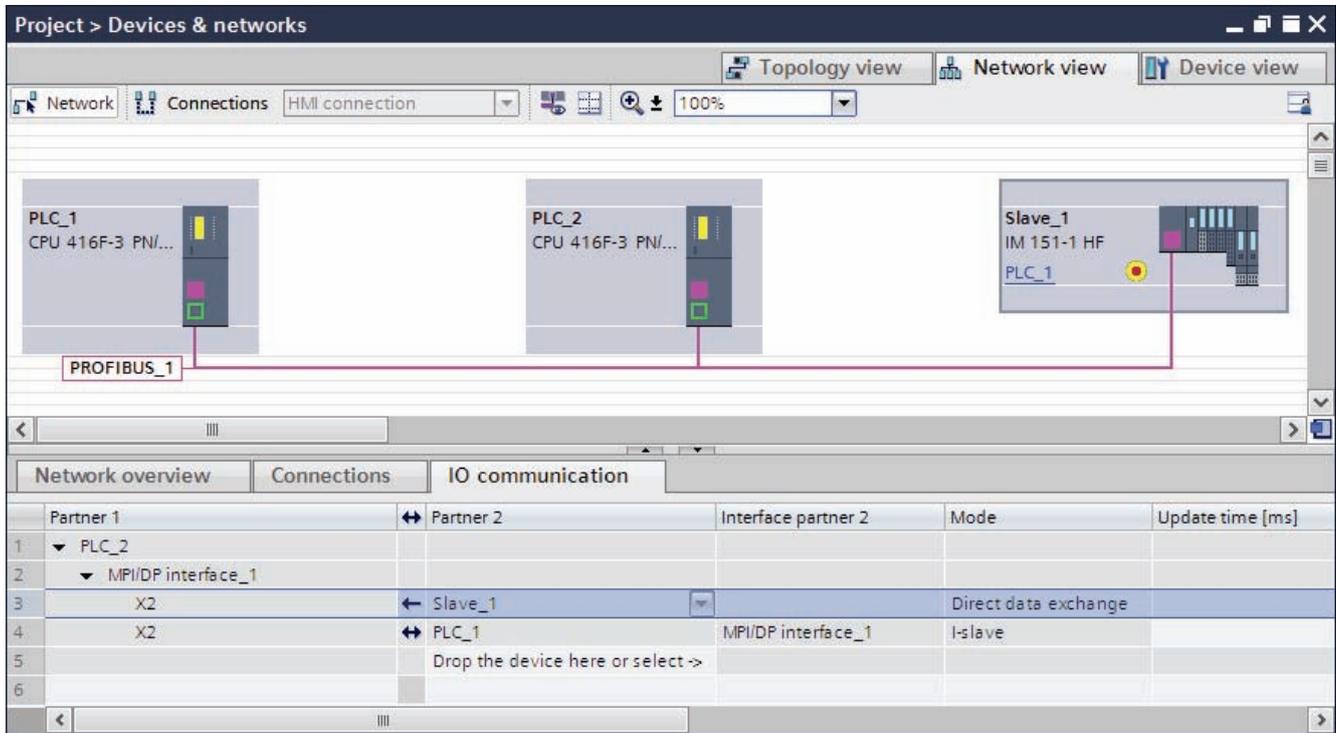


Procedure for configuring using the example of an ET 200S with fail-safe modules as slave

The procedure for configuring safety-related I-slave-slave communication is identical to that in the standard system. Proceed as follows:

1. Insert two F-CPUS from the "Hardware catalog" task card in the project.
2. Insert a suitable DP slave, e.g., IM 151-1 HF, order no. 6ES7151-1BA0... from the "Hardware catalog" task card into the network view of the *hardware and network editor*.
3. Assign a 4/8 F-DI module and a 4 F-DO-module in the device view of the ET 200S.
4. Activate the "DP slave" mode (I-slave) for F-CPU 2 in the properties of its DP interface and assign this to F-CPU 1.
5. Assign the DP interface of the IM 151-1 HF to the DP master (F-CPU 1).
6. Select the DP interface of F-CPU 2 (I-slave) in the network view.
7. Select the "I/O communication" tab.

- Use a drag-and-drop operation in the network view to move the ET 200S to the "Partner 2" column in the "I/O-communication" tab.



- In "Transfer areas", create an F-DX-Mod connection (type "F-DX-Mod"). The F-DX-Mod connection is marked in yellow in the table. The addresses for the "partner module" 4/8 F-DI in the I-Slave (PLC_2) are displayed. You can change the addresses directly in the table, if required.

This completes the configuration for the 4/8 F-DI module.

- In "Transfer areas", create another F-DX-Mod connection.

11. Change the partner module to the 4 F-DO module, either directly in the "Transfer areas" table or in the details of transfer area 2, if the 4 F-DO module was not already selected.

This completes the configuration for the 4/8 F-DO module.

The screenshot shows the SIMATIC Manager interface. The top part displays the 'IO communication' tab with a table of connections between Partner 1 and Partner 2. The bottom part shows the 'Direct data exchange' configuration window, specifically the 'General' tab, which contains a 'Transfer areas' table.

Partner 1	Partner 2	Interface partner 2	Mode	Update time [ms]
1 PLC_2				
2 MPI/DP interface_1				
3 X2	← Slave_1		Direct data exchange	
4 X2	↔ PLC_1	MPI/DP interface_1	I-slave	
5	Drop the device here or select ->			
6				

Transfer area	Type	Partner module	Address in PLC_2	Length	Consistency
1 Transfer area 1	F-DX-Mod	4/8 F-DI DC24V_1 (Slot...	I 0...5	6 Byte	Unit
2 Transfer area 2	F-DX-Mod	4 F-DO DC24V/2A_1 (S...	I 18...22	5 Byte	Unit
3 <Add new>					

In the "I-slave communication table" of the I-slave, a transfer area (type F-MS) for the master CPU (disabled in display) is created for each F-DX-Mod connection.

The screenshot displays the SIMATIC Manager interface. The top window shows the 'IO communication' tab with a table of connections:

Partner 1	Partner 2	Interface partner 2	Mode	Update time [ms]
1 PLC_2	↔ Partner 2			
2 MPI/DP interface_1				
3 X2	← Slave_1		Direct data exchange	
4 X2	↔ PLC_1	MPI/DP interface_1	I-slave	
5	Drop the device here or select ->			
6				

The bottom window shows the 'I-slave communication' configuration with the following table:

...	Transfer area	Type	Master address	↔ Slave address	Length	Consistency
1	Transfer area_1	F-MS	I 11...14	← Q 0...3	4 Byte	Total length
2	Transfer area_2	F-MS	I 15...19	← Q 18...22	5 Byte	Total length
3	<Add new>					

Change in configuration of I-slave-slave communication

⚠ WARNING

If you have reconfigured I-slave-slave communication for an F-I/O or have deleted an existing I-slave-slave communication, you must compile the hardware configuration of the DP master as well as the hardware configuration of the I-slave and download them to the DP master and I-slave, respectively.

The collective F-signature in the F-CPU of the I-slave and the collective F-signature in the F-CPU of the DP master (if a safety program exists there, too) are set to "0". You must then recompile the safety program(s). (S019)

9.7.2 Safety-Related I-Slave-Slave Communication - F-I/O Access

Access via the process image

In safety-related I-slave-slave communication, you use the process image (PII or PIQ) to access the F-I/O in the safety program of the F-CPU of the I-slave. This is the same as F-I/O access to F-I/O that are directly assigned to an I-slave or DP master. In the I-slave you access the F-I/O with the addresses that were assigned for the F-DX-Mod connection in "Transfer areas" ("Direct data exchange" table).

In this case, ignore the displayed operand area. Access F-I/O with inputs using the PII and F-I/O with outputs using PIQ.

Information on I/O access can be found in F-I/O access (Page 79).

9.7.3 Limits for data transfer of safety-related I-slave-I-slave communication

Limits for data transfer

Note the maximum limit of 244 bytes of input data or 244 bytes of output data for transfer between an I-slave and a DP master.

An example of the amount of output data and input data that are assigned for safety-related communication is shown in the table below for a 4/8 F-DI and a 4 F-DO ET 200S:

Safety-related communication	Communication connection	Assigned input and output data*	
		Between I-slave and DP master	
		Output Data in the I-slave	Input data in the I-slave
I-slave-slave	I-slave-slave communication with 4/8 F-DI	4 bytes	6 bytes
	I-slave-slave communication with 4 F-DO	5 bytes	5 bytes

* Example for 4/8 F-DI and 4 F-DO of ET 200S

Consider all additional configured safety-related and standard communication connections (F-MS, F-DX, F-DX-Mod., MS, DX und DX-Mod. connections) for the maximum limit of 244 bytes of input data or 244 bytes of output data for transfer between an I-slave and a DP master. If the maximum limit of 244 bytes of input data or 244 bytes of output data is exceeded, you will receive a corresponding error message.

9.8 Safety-related IO Controller-I-slave communication

Introduction

Safety-related communication between the safety program of the F-CPU of an IO-controller and the safety program(s) of the F-CPU(s) of one or more I-slaves takes place over master-I-slave connections (F-MS), as in standard systems.

IE/PB link

For the safety-related IO-controller-I-slave communication, the IE/PB link is mandatory. Each of the two F-CPU(s) is linked to the IE/PB link by means of its PROFIBUS DP or PROFINET-interface.

Note

If you are using an IE/PB link, you must take this into account when configuring the F-specific monitoring times and when calculating the maximum response time of your F-system (see also Monitoring and response times (Page 523)).

Note that the Excel file for calculating response times does not support all of the conceivable configurations.

Reference

In addition, the information on safety-related master-I-slave communication in Safety-related master-I-slave communication (Page 136) also applies analogously.

9.9 Safety-related communication via S7 connections

9.9.1 Configuring safety-related communication via S7 connections

Introduction

Safety-related communication between the safety programs of F-CPU's via S7 connections takes place by means of established S7 connections that you create in the network view of the *hardware and network editor* - same as in standard programs.

Restrictions

Note

In SIMATIC Safety, S7 connections are generally permitted only via Industrial Ethernet.

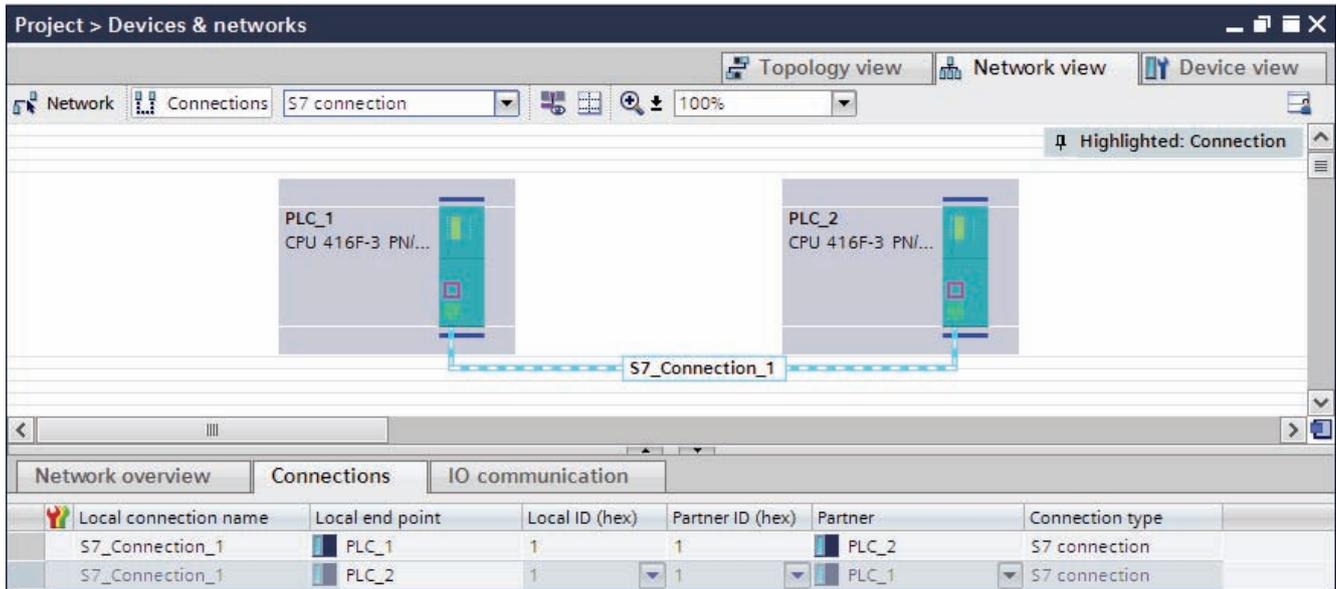
Safety-related communication via S7 connections is possible from and to the following CPUs:

- S7-300 F-CPU's via the integrated PROFINET interface
 - S7-400 F-CPU's via the integrated PROFINET interface or a CP 443-1 Advanced-IT
-

Creating S7 connections

For each connection between two F-CPU's, you must create an S7 connection in the network view of the *hardware and network editor*.

For every end-point of a connection, a local and a partner ID is automatically assigned from the perspective of the end-point (the F-CPU). If necessary, you can change both IDs in the "Connections" tab. You assign the local ID to the "ID" parameter of the SENDS7 and RCVS7 instructions in the safety programs.

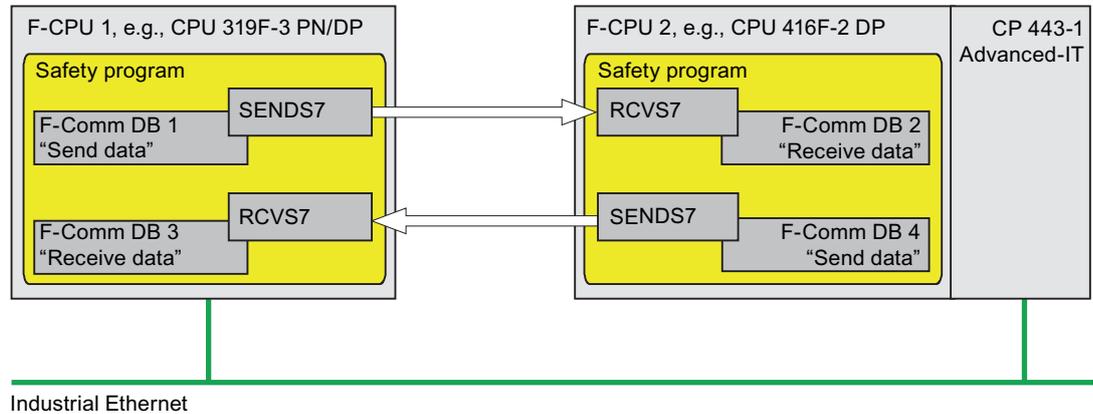


Procedure for configuring S7 connections

You configure the S7 connections for safety-related CPU-to-CPU communication in the same way as in *STEP 7 Professional* (see *help on STEP 7 Professional* "S7 connections").

9.9.2 Communication via SENDS7, RCVS7, and F-Communication DB

Communication via the SENDS7 and RCVS7 instructions



You use the **SENDS7** and **RCVS7** instructions for fail-safe sending and receiving of data via S7 connections.

These instructions can be used to transmit a specified amount of fail-safe data of data types BOOL, INT, WORD, DINT, DWORD, and TIME in a fail-safe manner. The fail-safe data are stored in F-DBs (F-communication DBs) that you have created.

You will find these instructions in the "Instructions" task card under "Communication". The **RCVS7** instruction **must** be called at the start of the main safety block. The **SENDS7** instruction **must** be called at the end of the main safety block.

Note that the send signals are sent only after calling the **SENDS7** instruction at the end of the relevant F-runtime group execution.

A detailed description of the **SENDS7** and **RCVS7** instructions is found in **SENDS7 and RCVS7: Communication via S7 connections (STEP 7 Safety Advanced V11)** (Page 517).

F-communication DB

For each connection, send data are stored in an F-DB (F-communication DBx) and receive data are stored in an F-DB (F-communication DBy).

You can assign the F-communication DB numbers in the **SENDS7** and **RCVS7** instructions.

9.9.3 Programming safety-related communication via S7 connections

Introduction

The programming of safety-related CPU-CPU communication via S7 connections is described below. You must set up the following in the safety programs of the relevant F-CPU:

- Create F-DBs (F-Communication-DBs) in which send/receive data for communication are stored.
- Call and assign parameters for instructions for communication from the "Instructions" Task Card in the safety program.

Requirement for Programming

The S7 connections between the relevant F-CPU must be configured in the network view in the "Connections" tab of the *hardware and network editor*.

Creating and Editing an F-Communication DB

F-communication DBs are F-DBs that you create and edit in the same way as other F-DBs in the project tree. You can assign the F-communication DB numbers in the SENDS7 and RCVS7 instructions.

Note

The length and structure of the F-communication DB on the receiver side must match the length and structure of the associated F-communication DB on the sender side.

If the F-communication DBs do not match, the F-CPU can go to STOP mode. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"
- "Safety program: internal CPU fault; internal error information: 404"

For this reason, we recommend that you use the following procedure:

1. Create an F-communication DB in the project tree in or below the "Program blocks" folder of the F-CPU at the sender side.
 2. Specify the appropriate structure of the F-communication DB, taking into account the data to be transferred.
 3. Copy this F-communication DB to the project tree in or below the "Program blocks" folder of the F-CPU of the receiver side, and change the name, if necessary.
-

Other requirements for F-communication DBs

F-communication DBs must also conform to the following properties:

- They are not permitted to be instance DBs.
- Their length is not permitted to exceed 100 bytes.
- In F-communication DBs, only the following data types may be declared: BOOL, INT, WORD, DINT, DWORD, and TIME.
- The data types must be arranged block-wise and in the following order: BOOL, data types with bit length of 16 bits (INT, WORD), and data types with bit length of 32 bits (DINT, DWORD, and TIME). Within the data blocks with lengths of 16 bits and 32 bits, the data types can be arranged in any order.
- No more than 128 data elements of data type BOOL are permitted to be declared.
- The amount of data of data type BOOL must always be an integer multiple of 16 (word limit). Reserve data must be added, if necessary.

If these criteria are not fulfilled, *STEP 7 Safety Advanced V11* outputs an error message.

Assignment of fail-safe values

Fail-safe values are made available from the receiver side:

- While the connection between the communication partners is being established the first time after startup of the F-systems
- Whenever a communication error occurs

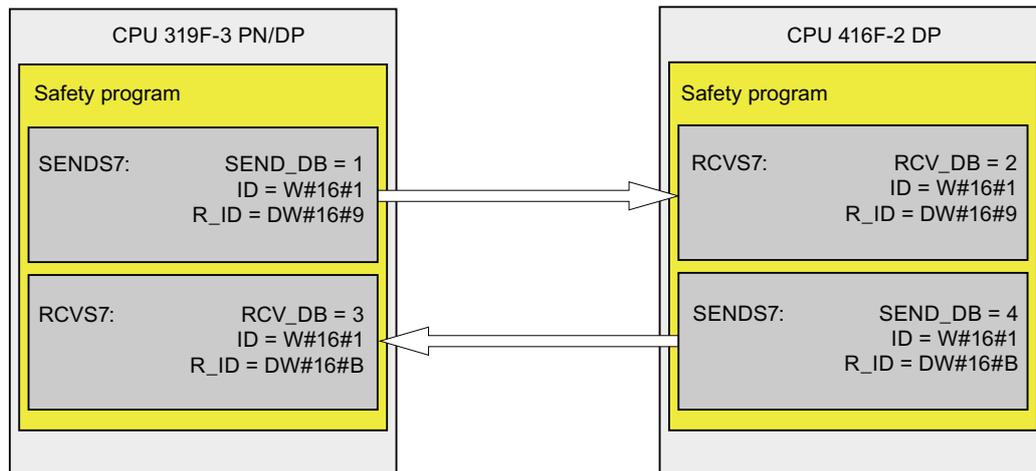
The values you specified as initial values in the F-communication DB on the receiver side are made available as initial values.

Programming procedure

You program safety-related communication via S7 connections as follows:

1. Supply the tags in the F-communication DB of the sender side with send signals using fully qualified access (e.g., "Name of F-communication DB"."tag name").
2. Read the tags in the F-communication DB of the receiver side (receive signals) that you want to process further in other sections of the program using fully qualified access (e.g., "Name of F-communication DB"."tag name").
3. In the safety program from which data are to be sent, call the SENDS7 instruction for sending at the end of the main safety block.
4. In the safety program from which data are to be received, call the RCVS7 instruction for receiving at the start of the main safety block.
5. Assign F-communication DB numbers to the SEND_DB input of SENDS7 and the RCV_DB input of RCVS7.
6. Assign the local ID of the S7 connection (data type:WORD) from the perspective of the F-CPU that was configured in the "Connections" tab of the network view to the ID input of SENDS7 .

7. Assign the local ID of the S7 connection (data type: WORD) that was configured in the "Connections" tab of the network view to the ID input of RCVS7 .
8. Assign an odd number (data type: DWORD) to the R_ID inputs of SENDS7 and RCVS7 . This serves to specify that a SENDS7 instruction belongs to an RCVS7 instruction. The associated instructions receive the same value for R_ID.



WARNING

The value for each address relationship (input parameter R_ID; data type: DWORD) is user-defined; however, it must be an odd number and be unique from all other safety-related communication connections in the network. The value R_ID + 1 is internally assigned and must not be used.

You must supply inputs ID and R_ID with constant values when calling the instruction. Direct read or write access to the associated instance DB is not permitted in the safety program! (S020)

9. Assign the TIMEOUT inputs of the SENDS7 and RCVS7 instructions with the required monitoring time.

WARNING

It can only then be ensured (from a fail-safe standpoint) that a signal level to be transferred will be captured on the sender side and transferred to the receiver if the signal level is pending for at least as long as the assigned monitoring time. (S018)

Information on calculating the monitoring times can be found in Monitoring and response times (Page 523).

10. To reduce the bus load, you can temporarily shut down communication between the F-CPU at the EN_SEND input of the SENDS7 instruction. To do so, specify the "F_GLOBDB".VKE0 tag (fully qualified) in the EN_SEND input (initial value = "TRUE"). In this case, send data are no longer sent to the F-communication DB of the associated RCVS7 instruction and the receiver RCVS7 provides fail-safe values for this period (initial values in its F-communication DB). If communication was already established between the partners, a communication error is detected.

11. Optional: evaluate the ACK_REQ output of RCVS7, for example, in the standard user program or on the operator control and monitoring system in order to query or to indicate whether user acknowledgment is required.
12. Supply the ACK_REI input of RCVS7 with the signal for the acknowledgment for reintegration.
13. Optional: evaluate the SUBS_ON output of RCVS7 or SENDS7 in order to query whether the RCVS7 instruction is outputting the fail-safe values you specified as initial values in the F-communication DB.
14. Optional: evaluate the ERROR output of RCVS7 or SENDS7, for example, in the standard user program or on the operator control and monitoring system in order to query or to indicate whether a communication error has occurred.
15. Optional: evaluate the SENDMODE output of RCVS7 in order to query whether the F-CPU with the associated SENDS7 instruction is in deactivated safety mode (Page 190).

Particularities for migrated projects

If you have migrated a project from *S7 Distributed Safety V5.4* to *STEP 7 Safety Advanced V11* in which safety-related communication via S7 connections is programmed, you must note the following:

- Do not delete migrated instance DBs for the SENDS7 and RCVS7 instructions in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks".

Otherwise, communication errors may occur in the relevant communication connections.

A migrated instance DB for the SENDS7 and RCVS7 instructions has been deleted if, after compiling the safety program, the "User-defined ID" in the newly generated is not identical to "FRCVS7CL" or "FSNDS7CL".

You will find the "User-defined ID" of a block in its properties in the "Information" area.

9.9.4 Safety-related communication via S7 connections - Limits of data transfer

Note

If the amount of data to be transmitted exceeds the permitted length for the F-communication DB (100 bytes), you can create another F-communication DB that you transfer to additional SENDS7/RCVS7 instructions with modified R_ID.

Note that USEND and URCV instructions are called internally at each SENDS7 or RCVS7 call and use connection resources in the F-CPU. This affects the maximum number of communication connections available (*see manuals for F-CPU*s).

Additional information on the data transfer limits for S7 connections of individual F-CPU is available on the Internet (<http://support.automation.siemens.com/WW/view/en/38549114>).

9.10 Safety-related communication with S7 F-systems

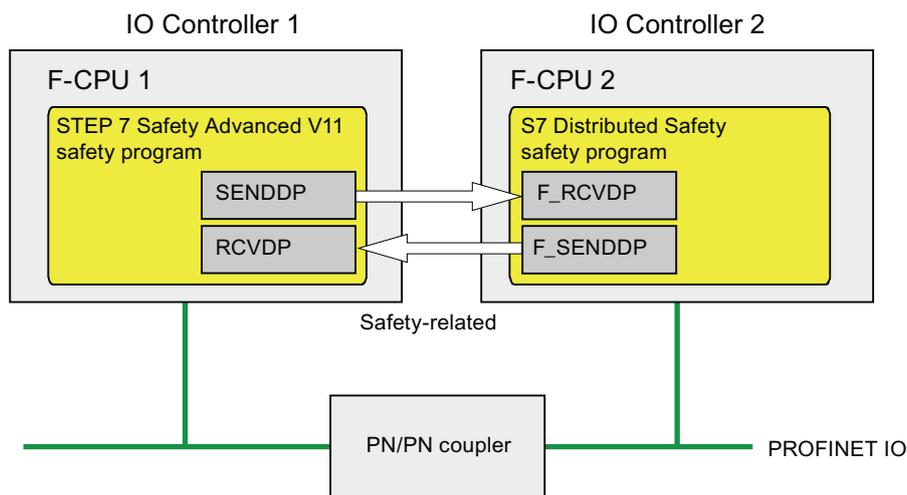
9.10.1 Introduction

Safety-related communication from F-CPU in SIMATIC Safety to F-CPU in S7 Distributed Safety F-systems is possible via a PN/PN coupler or DP/DP coupler that you use between the two F-CPU as IO Controller-IO Controller communication, master-master communication or communication via established S7 connections.

Safety-related communication from F-CPU in SIMATIC Safety to F-CPU in S7 F/FH Systems F-systems is possible via established S7 connections.

9.10.2 Communication with S7 Distributed Safety via PN/PN coupler (IO Controller-IO Controller communication)

Communication functions between SENDDP/RCVDP instructions on the *STEP 7 Safety V11* side and F_SENDDP/F_RCVDP F-application blocks on the *S7 Distributed Safety* side:



Procedure on the *S7 Distributed Safety* side

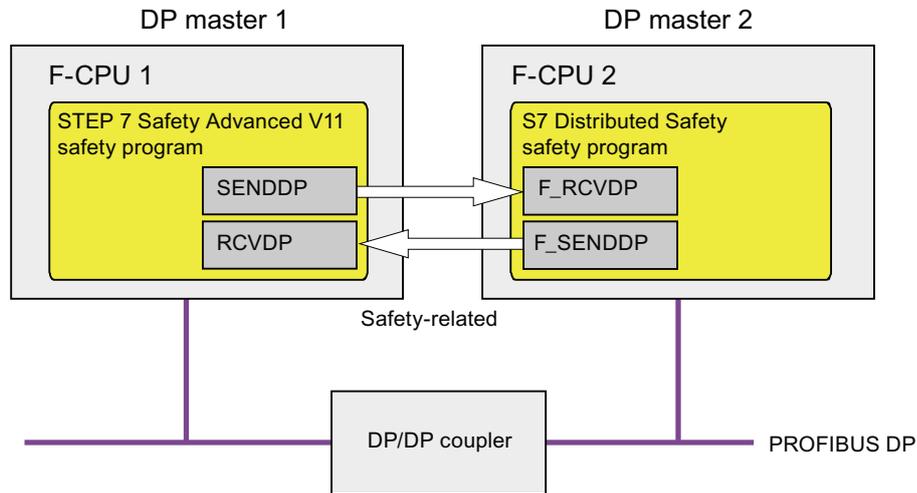
On the *S7 Distributed Safety* side, proceed as described in chapter "Safety-related IO Controller-IO Controller Communication" in the S7 Distributed Safety - Configuring and Programming (<http://support.automation.siemens.com/WW/view/en/22099875>) manual.

Procedure on the *STEP 7 Safety Advanced V11* side

On the *STEP 7 Safety Advanced V11* side, proceed as described in Safety-related IO controller-IO controller communication (Page 114).

9.10.3 Communication with S7 Distributed Safety via DP/DP coupler (master-master communication)

Communication functions between SENDDP/RCVDP instructions on the *STEP 7 Safety V11* side and F_SENDDP/F_RCVDP F-application blocks on the *S7 Distributed Safety* side:



Procedure on the *S7 Distributed Safety* side

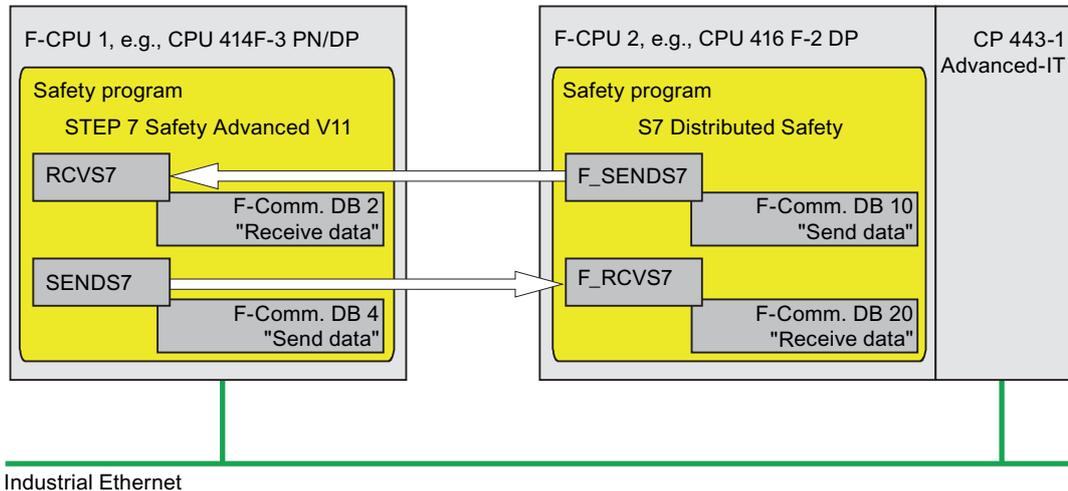
On the *S7 Distributed Safety* side, proceed as described in chapter "Safety-related master-master communication" in the *S7 Distributed Safety - Configuring and Programming* (<http://support.automation.siemens.com/WW/view/en/22099875>) manual.

Procedure on the *STEP 7 Safety Advanced V11* side

On the *STEP 7 Safety Advanced V11* side, proceed as described in *Safety-related master-master communication* (Page 122).

9.10.4 Communication with S7 Distributed Safety via S7 connections

Communication functions between SENDS7/RCVS7 instructions on the *STEP 7 Safety V11* side and F_SENDS7/F_RCVS7 F-application blocks on the *S7 Distributed Safety* side:



Procedure on the *S7 Distributed Safety* side

On the *S7 Distributed Safety* side, proceed as described in chapter "Safety-related communication via S7 communications" in the *S7 Distributed Safety - Configuring and Programming* (<http://support.automation.siemens.com/WW/view/en/22099875>) manual.

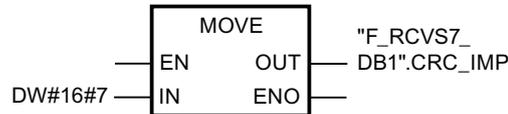
Because safety-related communication via S7 connections is not possible with unspecified partners in S7 Distributed Safety, you must first create a "virtual" SIMATIC station in *S7 Distributed Safety* in which you configure an F-CPU as a proxy for the F-CPU in *STEP 7 Safety Advanced V11* with its IP address.

You then insert an S7 connection to this F-CPU in the connection table. Both the local connection and partner connection resources (hex) are thereby fixed. You must then set these in the associated, unspecified S7 connection that you created in *STEP 7 Safety Advanced V11*.

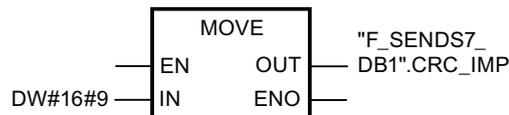
In addition, for all communication connections to this F-CPU, you must transfer the address relationship that you assigned in the R_ID input of the associated calls of the F_SENDS7 and F_RCVS7 F-application blocks additionally to the CRC_IMP data element in the instance DB of F_SENDS7 and F_RCVS7, respectively, in the standard user program immediately before calling the F-CALL.

Program example:

Network 1: Communication with STEP 7 Safety Advanced V11: R_ID -> CRC_IMP



Network 2: Communication with STEP 7 Safety Advanced V11: R_ID -> CRC_IMP



Procedure on the *STEP 7 Safety Advanced V11* side

On the *STEP 7 Safety Advanced V11* side, proceed as described in Safety-related communication via S7 connections (Page 155).

For the F-CPU in *S7 Distributed Safety*, you must create and specify an unspecified S7 connection (links to standard online help: [Creating an unspecified connection and Specifying and unspecified connection](#)).

For these you must set the local and partner connection resources (hex) that are fixed as a result of the associated S7 connection that you have created in *S7 Distributed Safety*.

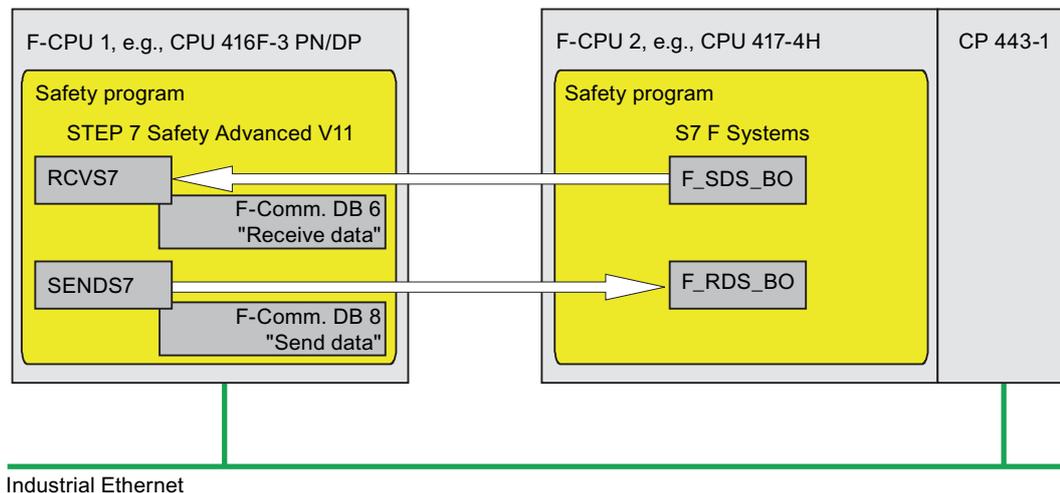
If the local connection resource (hex) is already occupied by an existing connection, you must change the connection resource (hex) for it.

If the instance DBs of the SENDS7 and RCVS7, instructions that you want to use for communication with *S7 Distributed Safety* were migrated from *S7 Distributed Safety*, you must delete them in the project tree in the "STEP 7 Safety" folder, under "Program blocks > System blocks" (contrary to the information in chapter Programming safety-related communication via S7 connections (Page 158), section "Particularities for migrated projects").

9.10.5 Communication with S7 F/FH Systems via S7 connections

Communication functions between SENDS7/RCVS7 instructions on the *STEP 7 Safety Advance V11* side and F_SDS_BO/F_RDS_BO F-blocks on the *S7 F Systems* side.

A maximum of 32 data elements of data type BOOL can be exchanged.



Procedure on the *S7 F Systems* side

On the *S7 F-systems* side, proceed as described in chapter "Safety-related communication between F-CPU's" in the *S7 F/FH Systems - Configuring and Programming* (<http://support.automation.siemens.com/WW/view/en/16537972>) manual.

Because safety-related communication via S7 connections is not possible with unspecified partners in *S7 F/FH Systems*, you must first create a "virtual" SIMATIC station in *S7 F/FH Systems* in which you configure an F-CPU as a proxy for the F-CPU in *STEP 7 Safety Advanced V11* with its IP address.

You then insert an S7 connection to this F-CPU in the connection table. Both the local connection and partner connection resources (hex) are thereby fixed. You must then set these in the associated, unspecified S7 connection that you created in *STEP 7 Safety Advanced V11*.

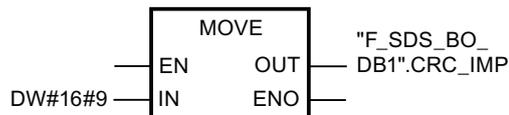
In addition, you must insert a function in your S7 program (in the area reserved in CFC for other applications), in which, for all communication connections for this F-CPU, you transfer the address relationship that you assigned in the R_ID input of the associated calls of the F_SDS_BO and F_RDS_BO F-blocks additionally to the CRC_IMP data element in the instance DB of the F_SDS_BO and F_RDS_BO, respectively. You obtain the number of the instance DB from the object properties of the block in CFC. Assign descriptive symbolic names for these instance DBs. If you perform a compress operation in CFC, you must check whether the numbers of these instance DBs have changed.

Program example:

Network 1: Communication with STEP 7 Safety Advanced V11: R_ID -> CRC_IMP



Network 2: Communication with STEP 7 Safety Advanced V11: R_ID -> CRC_IMP



You must then import the function in CFC as block type and insert your standard user program in a chart. In the run sequence, make sure that the associated standard runtime group is processed before the F-runtime group.

Procedure on the *STEP 7 Safety Advanced V11* side

On the *STEP 7 Safety V11* side, proceed as described in "Safety-related communication via S7 communications" (Page 155).

Particularity: In *STEP 7 Safety Advanced V11*, you must create the F-communication DB with exactly 32 data elements of data type BOOL.

For the F-CPU in *S7 F/FH Systems*, you must create and specify an unspecified S7 connection (see online help of *STEP 7 Professional* under "Creating an unspecified connection" and "Specifying an unspecified connection").

For these you must set the local and partner connection resources (hex) that are fixed as a result of the associated S7 connection that you have created in *S7 F Systems*.

If the local connection resource (hex) is already occupied by an existing connection, you must change the connection resource (hex) for it.

Compiling and commissioning a safety program

10.1 Compiling the safety program

Introduction

To compile a safety program, follow the same basic procedure as for compiling a standard user program. There are various points of entry to accomplish this in *STEP 7 Professional*. The basics for compiling user programs can be found in the *help on STEP 7 Professional*.

We will show you the options for compiling the safety program later.

Basics for compiling

1. Selection	2. Via menu command/symbol...	3. The following are compiled...	4. Result: Safety program is...
Select folder of the F-CPU in the project tree or Select F-CPU in network view or Select F-CPU in the device view or Select F-CPU in the topology view	"Compile" context menu:		
	• "All"	Hardware configuration, standard and safety programs	Consistent
	• "Hardware configuration"	Hardware configuration	—
	• "Software"	Changes in the standard and safety programs	Consistent
Select "Program blocks" folder in the project tree	Menu "Edit > Compile" or "Compile" button	Entire standard program and safety program	Consistent
	"Compile" context menu:		
	• "Software"	Changes in the standard and safety programs	Consistent
Select user-created folder in the project tree User-created folder containing all F-blocks	Menu "Edit > Compile" or "Compile" button	All standard and F-blocks contained in the folder	Inconsistent
	Menu "Edit > Compile" or "Compile" button	All standard blocks contained in the folder and complete safety program	Consistent

1. Selection	2. Via menu command/symbol...	3. The following are compiled...	4. Result: Safety program is...
(F-)blocks in "Program blocks" folder in the project tree	Menu "Edit > Compile" or "Compile" button	Selected standard and F-blocks	Inconsistent
Different (F-)blocks in "Program blocks" folder in the project tree			
Individual (F-)blocks in "Program blocks" folder in the project tree			
Downloading a safety program to a F-CPU	Automatic compiling	Entire safety program	Consistent

A consistency check is always performed, regardless of the selection. This consistency check extends across all selected blocks. If the consistency check does not detect any errors, the status of the compiled safety program is as specified in column 4 "Result: Safety program is..." of the table.

Result "Safety program is consistent"

Following a successful compile operation of the safety program, the block container always contains a consistent safety program, which is in the "Program blocks" folder and composed of all F-blocks with F-attribute.

Nevertheless, there can be F-blocks without F-attribute. This is the case when an F-block is not called in an F-runtime group. These F-blocks are displayed in the "F-blocks" area of the *Safety Administration Editor*, in which they are marked with "No" in the "Used and compiled" column .

Result "Safety program is inconsistent"

When the safety program is compiled with the result "Safety program is not consistent", only the selected F-blocks were compiled. Additionally required F-blocks and F-system blocks were not generated. The safety program in the "Program blocks" folder is not consistent and is, thus, not executable.

Use this procedure to test modified F-blocks.

Reporting compiling errors

You can recognize whether or not the compilation was successful based on the message in the inspector window under "Info > Compile", error messages and warnings are output.

For information on the procedure you must follow to eliminate compiling errors, see Help in *help on STEP 7 Professional* under "Eliminating compiling errors".

You must not insert F-system blocks from the "System blocks" folder to a main safety block/F-FB/F-FC.

See also

Safety Administration Editor (Page 39)

10.2 Downloading the Safety Program

Introduction

Once you have successfully compiled your safety program, you can download it to the F-CPU. Follow the same basic procedure for downloading a safety program as for downloading a standard user program using various points of entry in *STEP 7 Professional*.

- In the "Load preview" dialog, enter data (e.g., password for F-CPU) and set the requirements for downloading (e.g., that the F-CPU is switched to STOP mode before downloading).
- The "Load results" dialog shows the results after downloading.

We will show you the options for downloading the safety program later. For basic information on downloading, refer to the *help on STEP 7 Professional*.

Downloading a safety program to an F-CPU, if multiple F-CPU's are accessible

WARNING

If multiple **F-CPU's** are accessible over a network (e.g., Ind. Ethernet) by **one programming device or PC**, you must take the following actions to ensure that the safety program is downloaded to the correct F-CPU:

Use passwords specific to each F-CPU, e.g., a uniform password for the F-CPU's with the respective Ethernet address as an extension (max. 8 characters) "PW_8".

Note the following:

- A point-to-point connection must be used when assigning a password to an F-CPU for the first time (analogous to assigning an MPI address to an F-CPU for the first time).
- Before downloading a safety program to an F-CPU, you must first revoke existing access permission for any other F-CPU.
- After activation of the access protection and before the transition to productive mode, you must download the safety program again to the F-CPU. (*S021*)

Password prompt before downloading to an F-CPU

If you have assigned a protection level for the F-CPU (Page 54) (in the properties of the F-CPU in the "Protection" tab), the corresponding password is prompted in "Load preview" dialog. Without entry of password, only actions that are allowed without password are possible. As soon as the conditions for downloading are met, the "Load" button becomes active.

Note

If *STEP 7 Safety V11* detects an inconsistent safety program during startup of the F-CPU, the F-CPU cannot be started up, provided the F-CPU supports this detection function (see Product Information for the particular F-CPU). The following diagnostic event is then entered in the diagnostic buffer of the F-CPU:

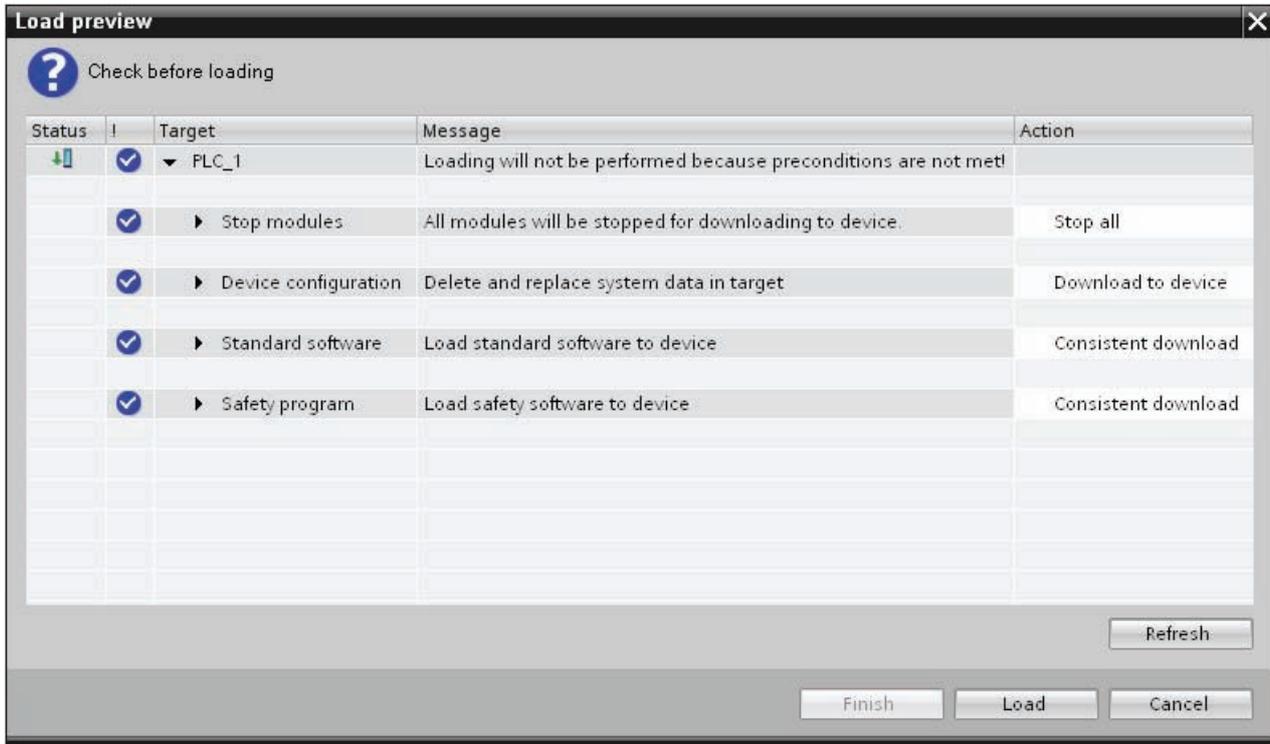
- "Inconsistent safety program"

If the F-CPU does not support this detection function, the F-CPU can go to STOP mode if an inconsistent safety program is executed in activated safety mode. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
 - "Data corruption in the safety program prior to output to partner F-CPU"
 - "Safety program: internal CPU fault; internal error information: 404"
-

"Load preview" dialog

For an F-CPU, the "Load preview" dialog also contains the "Safety program" section.

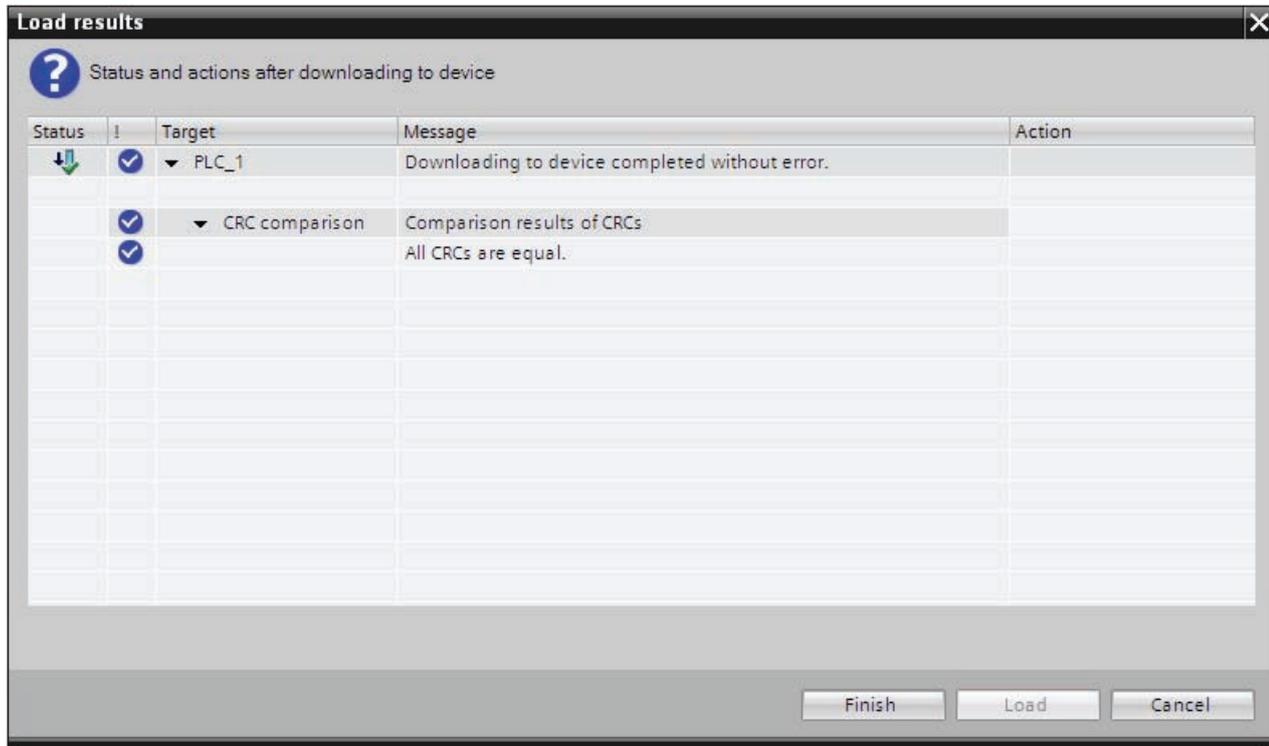


Make the following selection:

- In order to download a consistent safety program, select the "Consistent download" action under destination "Safety program".
- To download individual F-blocks selectively, select the "Download selection" action under destination "Safety program", and then select the required F-blocks. If necessary, you will be prompted to deactivate safety mode under "Deactivate safety mode". This setting is only suitable for the online test of individual F-blocks.
- In order to download the safety program only, select the "Consistent download" action under destination "Safety program" and the "Download selection" action under destination "Standard software", and then select only the standard blocks that call the main safety block.
- Users that do not know the password for the F-CPU must select "No action" for safety program.

"Load results" Dialog

After downloading into the F-CPU, the dialog "Load results" is opened. This dialog shows you the status and the necessary actions after downloading.



In this dialog, switch the F-CPU back to RUN mode, if necessary.

Result: If the F-CPU is online, the following status is displayed in the "General" work area of the *Safety Administration Editor*: "Safety mode is activated" will be shown.

If the F-CPU was in RUN mode during downloading, the dialog indicates that safety mode is deactivated. The "General" work area of the *Safety Administration Editor* indicates the status as "Safety mode is deactivated".

Verify that the "All CRCs are identical" message appears in this dialog. On the basis of the CRCs, the system checks after the download operation to determine whether all F-blocks were correctly transferred to the F-CPU.

Rules for downloading the safety program to an F-CPU

Note

You can perform the downloading of a consistent safety program only in STOP mode.

If you are downloading F-blocks only, the blocks in which the main safety blocks are called (e.g., cyclic interrupt OB 35) are not downloaded. To do so, select the "Select" option under "Standard software" in the preview dialog, and select the necessary blocks.

When downloading the safety program, ensure that the "Consistent download" action is set for the "Safety program" selection in the "Load preview" dialog.

Verify that the "All CRCs are identical" message appears in the "Load results" dialog. On the basis of the CRCs, the system checks after the download operation to determine whether all F-blocks were correctly transferred to the F-CPU. If not, repeat the download operation.

Inconsistent downloading is only possible in deactivated safety mode.

Loading of individual F-blocks

You can download F-blocks and standard blocks simultaneously to the F-CPU via the project tree, same as in *STEP 7 Professional*. However, as soon as F-blocks are to be downloaded, a check is carried out to determine whether or not the F-CPU is in STOP mode or deactivated safety mode. If not, you have the option of switching to deactivated safety mode or placing the F-CPU in STOP mode.

Be aware that the consistency of the safety program in the F-CPU cannot be guaranteed when individual F-blocks are downloaded. For a consistent safety program, always download the entire safety program to the F-CPU.

Note

If *STEP 7 Safety V11* detects an inconsistent safety program during startup of the F-CPU, the F-CPU cannot be started up, provided the F-CPU supports this detection function (see Product Information for the particular F-CPU). The following diagnostic event is then entered in the diagnostic buffer of the F-CPU:

- "Inconsistent safety program"

If the F-CPU does not support this detection function, the F-CPU can go to STOP mode if an inconsistent safety program is executed in activated safety mode. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
 - "Data corruption in the safety program prior to output to partner F-CPU"
 - "Safety program: internal CPU fault; internal error information: 404"
-

Rules for loading individual F-blocks

The following rules apply to downloading of individual F-blocks:

- Downloading is only possible in deactivated safety mode or when the F-CPU is in STOP mode.
- F-blocks can only be downloaded to an F-CPU to which a safety program has already been downloaded.
- Only an offline safety program is permitted to be used as a source program.

Consequently, the initial downloading of the safety program can be performed exactly the same way as downloading after a change in the safety program password.

Note

Downloading of individual F-blocks is only suitable for testing F-blocks. Prior to the transition to productive mode, you must download the safety program consistently to the F-CPU.

Downloading a migrated safety program to an F-CPU

You can download a safety program migrated (Page 22) from *S7 Distributed Safety V5.4 SP5* to an F-CPU.

Transferring a safety program with programming device/PC or memory card to an F-CPU

For information on transferring the safety program to the F-CPU with a PG/PC or a memory card, refer to Function Test of Safety Program and Protection through Program Identification (Page 178).

Uploading the safety program to a programming device or PC

Note

In principle, it is possible to upload a safety program from the F-CPU to a programming device or PC. Note, however, that any symbols used in the safety program are deleted and cannot be recreated, since no symbol information is saved in the F-CPU. Symbols are available only if you are using an offline project.

A safety program uploaded from the F-CPU to a programming device/PC must not be downloaded again to the F-CPU.

Procedure for uploading to a PG/PC

You upload a safety program from an F-CPU to a programming device/PC using menu command "Online> Upload device to PG/PC" or the "Load from device" button in the toolbar.

Downloading to an *S7-PLCSIM*

You can download the safety program with *S7-PLCSIM* (hardware simulation) similar to a real F-CPU. For this purpose, use the menu command "Online > Simulation > Start" to start *S7-PLCSIM*, same as in the standard system.

10.3 Work Memory Requirement for Safety Program

Estimation

You can estimate the work memory requirement for the safety program as follows:

Work memory required for the safety program

32 kbytes for F-system blocks

- + 4.4 kbytes for safety-related communication between F-runtime groups
- + 4.5 x work memory requirement for all F-FB/F-FCs/main safety blocks
- + 4.5 x work memory requirement of all utilized instructions, which are shown in the "Instructions" task card with the  block icon. (except SENDDP, RCVDP, SENDS7, and RCVS7)
- + Work memory requirement of the utilized SENDDP and RCVDP instructions (4.3 kbytes each)
- + Work memory requirement of the utilized SENDS7 and RCVS7 instructions (8.5 kbytes each)

Work memory required for data

5 x work memory requirement for all F-DBs (including F-communication DB, but excluding DB for F-runtime group communication) and I-DBs for main safety block/F-FB

- + 24 x work memory requirement for all DBs for F-runtime group communication
- + 2.3 x work memory requirement for all I-DBs of instructions (except SENDDP, RCVDP, SENDS7 and RCVS7)
- + Work memory requirement of all I-DBs of instructions SENDDP (0.2 kbytes), RCVDP(0.3 kbytes), SENDS7 (0.6 kbytes), and RCVS7 (1.0 kbyte)
- + 0.7 kbytes per F-FC
- + 0.7 kbytes per F-I/O (for F-I/O DBs, etc.)
- + 4.5 kbytes

Block size of automatically generated F-blocks

Do not utilize the entire maximum size of an F-block, because the automatically generated F-blocks are larger and as a result, the maximum possible size may be exceeded in the F-CPU. If the block size is exceeded, this triggers a corresponding error message with information on which F-blocks are too large. These must be divided up, if necessary.

10.4 Function Test of Safety Program and Protection through Program Identification

Transferring the safety program to the F-CPU with a programming device or PC

F-CPU with memory card inserted (Flash Card or SIMATIC Micro Memory Card)

The following warnings apply when the safety program is transferred from a programming device or PC to:

- F-CPU with flash card inserted (e.g., CPU 416F-2)
- F-CPU with SIMATIC Micro Memory Card
(e.g., CPU 317F-2 DP, CPU 315F-2 PN/DP or IM 151-7 F-CPU)

WARNING

If the function test of the safety program is not carried out in the target F-CPU, you must comply with the following procedure when transferring the safety program to the F-CPU with a **PG/PC** to ensure that the F-CPU does not contain an "old" safety program:

- Load the safety program to the F-CPU.
- Perform a program identification (that is, check to determine whether the collective F-signatures match online and offline).
- Perform a memory reset of the F-CPU using the mode selector or via the programming device/PC. Once the work memory has been deleted, the safety program is again transferred from the load memory (Memory Card, SIMATIC Micro Memory Card for F-CPU 3xxF, IM-CPU ET 200S und ET 200pro or Flash-Card for F-CPU 4xxF). (S022)

WARNING

If multiple **F-CPU**s are accessible over a network (e.g., Ind. Ethernet) by **one programming device or PC**, you must take the following actions to ensure that the safety program is downloaded to the correct F-CPU:

Use passwords specific to each F-CPU, e.g., a uniform password for the F-CPU with the respective Ethernet address as an extension (max. 8 characters) "PW_8".

Note the following:

- A point-to-point connection must be used when assigning a password to an F-CPU for the first time (analogous to assigning an MPI address to an F-CPU for the first time).
- Before downloading a safety program to an F-CPU, you must first revoke existing access permission for any other F-CPU.
- After activation of the access protection and before the transition to productive mode, you must download the safety program again to the F-CPU. (S021)

F-CPU without inserted Flash Card

The following warnings apply when the safety program is transferred from a programming device or PC to:

- F-CPU without an inserted Flash card (e.g., CPU 416F-2)

<p> WARNING</p> <p>If the function test of the safety program is not carried out in the target F-CPU, you must comply with the following procedure when transferring the safety program to the F-CPU with a PG/PC to ensure that the F-CPU does not contain an "old" safety program:</p> <ul style="list-style-type: none"> • Perform a memory reset of the F-CPU using the mode selector or via the programming device/PC. • Download the configuration and the safety program to the F-CPU. • Perform a program identification (that is, check to determine whether the collective F-signatures match online and offline). (<i>S023</i>)
--

<p> WARNING</p> <p>If multiple F-CPUs are accessible over a network (e.g., Ind. Ethernet) by one programming device or PC, you must take the following actions to ensure that the safety program is downloaded to the correct F-CPU:</p> <p>Use passwords specific to each F-CPU, e.g., a uniform password for the F-CPU with the respective Ethernet address as an extension (max. 8 characters) "PW_8".</p> <p>Note the following:</p> <ul style="list-style-type: none"> • A point-to-point connection must be used when assigning a password to an F-CPU for the first time (analogous to assigning an MPI address to an F-CPU for the first time). • Before downloading a safety program to an F-CPU, you must first revoke existing access permission for any other F-CPU. • After activation of the access protection and before the transition to productive mode, you must download the safety program again to the F-CPU. (<i>S021</i>)
--

Transferring the safety program to the WinAC RTX F

WARNING

In order to guarantee that no "old" safety program is in the F-controller, you must comply with the following procedure when transferring the safety program into the F-controller with a programming device/PC:

1. Perform a memory reset of WinAC RTX F (see Windows Automation Center RTX WinAC RTX (F) 2010 (<http://support.automation.siemens.com/WW/view/en/43715176>) manual).
2. Download the configuration to the WinAC RTX F (see Windows Automation Center RTX WinAC RTX (F) 2010 (<http://support.automation.siemens.com/WW/view/en/43715176>) manual).
3. Download the safety program (Page 171) to the WinAC RTX F.
If the function test of the safety program does not take place in the destination F-controller, you must also follow points 4 and 5:
4. Perform a program identification (that is, check to determine whether the collective F-signatures match online and offline).
5. Execute the F-system startup (Page 76).

Between the online program identification and the startup of the F-system, the WinAC RTX F may not be closed (for example, through NETWORK OFF/NETWORK ON or booting). (S024)

WARNING

If **multiple F-controllers** are accessible over a network (e.g., Ind. Ethernet) by a **programming device or PC**, you must take the following actions to ensure that the safety program is downloaded to the correct F-controller:

Use passwords specific to each F-controller, e.g., a uniform password for the F-controller Us with the respective Ethernet address as an extension (max. 8 characters) "PW_8".

Note the following:

- A point-to-point connection must be used when assigning a password to an F-controller for the first time (analogous to assigning an MPI address to an F-CPU for the first time).
- Before downloading a safety program to an F-controller, you must first revoke existing access permission for any other F-controller.
- After activation of the access protection and before the transition to productive mode, you must download the safety program again to the F-controller. (S021)

Rules for plugging removable media (for example, SIMATIC Micro Memory Card, Flash Card or hard disk) in the WinAC RTX F

 **WARNING**

You must limit the access for the WinAC RTX F through an access protection to persons, who are entitled for plugging removable media.

You must guarantee that the correct safety program is on the plugged removable media, either through online program identification or other measures (for example, unique ID of the removable medium). (S025)

Transferring the Safety Program to the F-CPU with a Memory Card

Use of SIMATIC Micro Memory Card or Flash Card

The following warning applies when the safety program is transferred using a:

- Flash Card (e.g., for CPU 416F-2)
- SIMATIC Micro Memory Card (e.g., for CPU 317F-2 DP, CPU 315F-2 PN/DP or IM 151-7 F-CPU)

WARNING

If the function test of the safety program is not carried out in the target F-CPU, you must comply with the following procedure when transferring the safety program to the F-CPU with a memory card (SIMATIC Micro Memory Card or Flash Card) to ensure that the F-CPU does not contain an "old" safety program:

- Turn off the power to the F-CPU. For F-CPU with battery backup (e.g., CPU 416F-2), remove the battery, if present. (To make sure that the F-CPU is de-energized, wait for the buffer time of the power supply you are using or, if this is unknown, remove the F-CPU.)
- Remove the Memory Card (SIMATIC Micro Memory Card or Flash Card) with the old safety program from the F-CPU.
- Plug the Memory Card (SIMATIC Micro Memory Card or Flash Card) with the new safety program into the F-CPU.
- Switch on the F-CPU again. For F-CPU with battery backup (e.g., CPU 416F-2), reinsert the battery, if one was removed.

You must make sure that the inserted memory card (SIMATIC Micro Memory Card or Flash Card) contains the correct safety program. You can do so through a program identification or other measures, such as a unique identifier on the memory card (SIMATIC Micro Memory Card or Flash Card).

When downloading a safety program to a memory card (**SIMATIC Micro Memory Card** or Flash Card), you must adhere to the following procedure:

- Download the safety program to the memory card (SIMATIC Micro Memory Card or flash card).
- Perform a program identification - in other words, check whether the collective F-signatures of all F-blocks with F-attribute in the offline block container and on the memory card (SIMATIC Micro Memory Card or Flash Card) match.
- Affix an appropriate label to the memory card (SIMATIC Micro Memory Card or Flash Card).

The procedure outlined must be ensured through organizational measures. (S026)

10.5 Comparing Safety Programs

Compare safety programs as in standard

You can use the *comparison editor* in *STEP 7 Professional* for offline-online or offline-offline comparison of safety programs. The procedure is the same as for standard user programs. For the comparison of safety programs, the contents of F-blocks are also compared. As a result, an offline-offline comparison can also be used for a change acceptance test (Page 210).

Note

During the offline-online comparison, the comparison statuses may occasionally differ between the *comparison editor* and status display in the project tree or *Safety Administration Editor*. The decisive status is the result of the comparison in the *comparison editor*, since this is the only comparison that takes into account the contents of the F-blocks.

Comparison output of a comparison of safety programs

The representation of the comparison result corresponds to *STEP 7 Professional*.

In addition, you also receive the following information for safety programs in the "Details" column:

- The signature of the F-blocks is displayed if it can be determined; if it cannot be determined, the reason is given.
- If the F-blocks are found not to be identical, you receive more detailed information on the differences.

During the offline-offline comparison, the collective signature is also displayed if the safety program is consistent. Otherwise, the status of the safety program is displayed.

Note

If you interrupt the connection to the F-CPU during the comparison, the comparison result will be incorrect.

The meaning of the symbols in the "Status" column is explained below:

Table 10- 1 Offline-online comparison: Status of the comparison and causes

Symbol	Meaning for safety program	Possible causes
	The offline and online versions of the object are identical.	All criteria named above match both offline and online.
	The offline and online versions of the object are different.	The differences are indicated in the "Details" column.
	Object only available offline	The block is only available offline.
	Object only available online	The block is only available online.

Table 10- 2 Offline-offline comparison: Status of the comparison and causes

Symbol	Meaning for safety program	Possible causes
	Both offline versions of the object are identical.	All criteria indicated above are met
	Both offline versions of the object are different	The differences are indicated in the "Details" column.
	Object available only in safety program 1	—
	Object available only in safety program 2	—

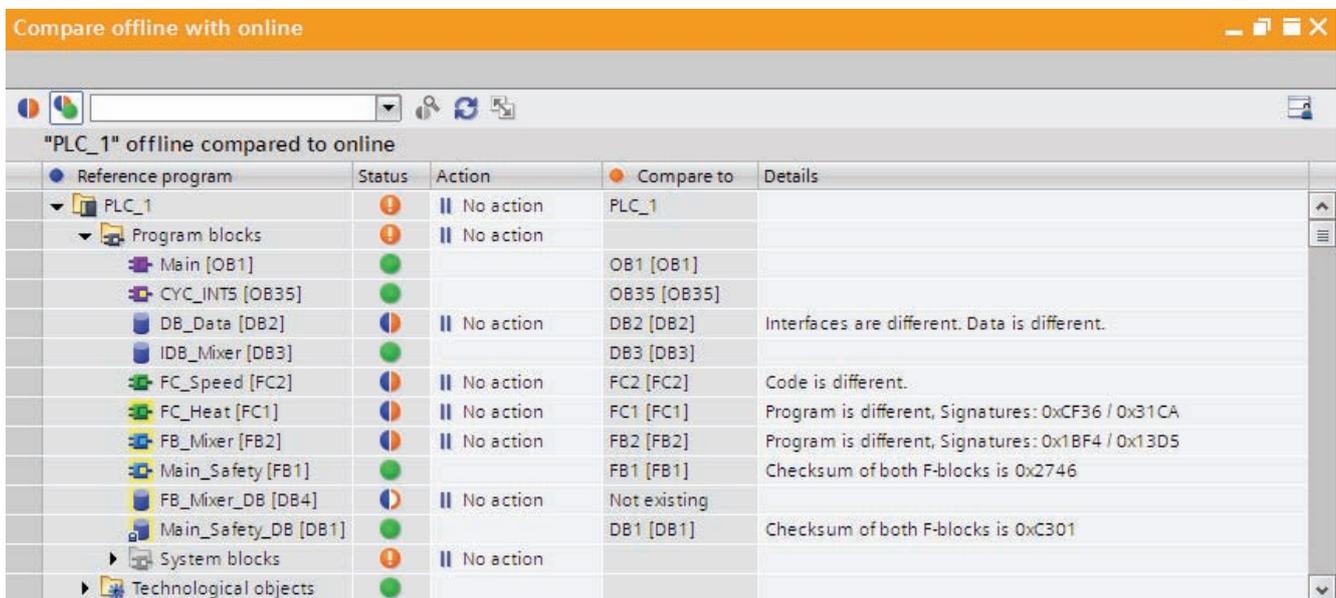


Figure 10-1 Comparison result of an offline-online comparison

Reference program	Status	Action	Compare to	Details
PLC_1	ⓘ	No action	PLC_1	
Program blocks	ⓘ	No action	Program blocks	Collective F-signatures of both safety programs: 0xAA1C8B86
Main [OB1]	●		Main [OB1]	
CYC_INT5 [OB35]	●		CYC_INT5 [OB35]	
DB_Data [DB2]	●		DB_Data [DB2]	
IDB_Mixer [DB3]	●		IDB_Mixer [DB3]	
FC_Speed [FC2]	●		FC_Speed [FC2]	
FC_Heat [FC1]	ⓘ	No action	FC_Heat [FC1]	Signatures: 0xCF36 / Not available
FB_Mixer [FB2]	ⓘ	No action	FB_Mixer [FB2]	Binary is different, Signatures: 0x1BF4 / 0x7288
Main_Safety [FB1]	ⓘ	No action	Main_Safety [FB1]	Signatures: 0x2746 / Not available
FB_Mixer_DB [DB4]	ⓘ	No action	FB_Mixer_DB [DB4]	Signatures: 0x26AE / Not available
Main_Safety_DB [DB1]	●		Main_Safety_DB [DB1]	Checksum of both F-blocks is 0xC301
System blocks	ⓘ	No action		
Technological objects	●			
PLC tags	●			
PLC data types	●			

Figure 10-2 Comparison result of an offline-offline comparison

Comparison filter options

You can use filters in the *comparison editor* to limit the comparison result to the following block groups:

- Compare only F-blocks
- Compare only F-blocks relevant for certification
- Compare all blocks
- Compare only standard blocks

In addition, the filter options "Show only Objects with differences" and "Show identical and different objects" of *STEP 7 Professional* are also available.

For comparison of safety programs, F-blocks in the "System blocks" folder are also relevant. Manually open the folder along with its subfolders in order to view the differences for all F-blocks.

Assignment of displayed changes

Regardless of whether you carried out an offline/online or offline/offline-comparison, the following changes may account for the indicated, changed objects:

- Change in the maximum cycle time of the F-runtime group
- Change in F-parameters of the F-CPU
- Modified version of F-system blocks
- Change in the F-runtime group communication, for example, change in the number of a DB for F-runtime group communication
- Change in main safety block, F-FB, F-FC, F-DB
- Change in the F-I/O addressed in the safety program
- Change in read access to data of the standard user program

Printing result of comparison

The comparison result can be printed via "Project > Print" in the menu bar or the print button in the toolbar. Choose "Objects/Area" "All" and "Properties" "All".

10.6 Printing project data

Printing

You can print all important project data of the hardware configuration of the F-I/O and safety program. As a result, you receive a "safety printout" that serves as a basis for the test for correctness of the individual components of the system. Correctness is a prerequisite for system acceptance.

The collective F-signature specifications in the footer of the printout ensure that the printout is explicitly associated with a safety program.

Procedure for creating the safety program printout

To create a safety printout, follow these steps:

1. In the project tree, select the *Safety Administrations Editor* of the F-CPU, whose safety printout you want to create.
2. Select "Print" in the context menu or "Project > Print" in the menu bar or the print button in the toolbar.

In the displayed dialog, you can make layout settings for the printout and specify the scope of the printout (all/subset), among other things.

3. Select the "All" option, if the source code of the F-blocks is to be shown in the printout. This is necessary, for example, to document the program code for the acceptance test (see Acceptance test of system (Page 201)). Select the "Compact" option to exclude the source code from the printout.
4. Click the "Print" button.

As a result, you receive the safety printout for the F-CPU.

Safety printout

The safety printout provides documentation of the safety program provides support for the acceptance test of the system.

Contents of the printout in overview

The topics that are considered in the printout are summarized in the following:

- General information on program identification, software versions, access protection, settings (from the "Settings" work area of the *Safety Administration Editor*)
- Utilized elements of the internal system libraries (from "Instructions" task card and F-system blocks) along with their versions and signatures
- Information about the F-runtime groups (F-monitoring time, F-blocks and their names)
- Listing of the F-blocks within the "Program blocks" folder (name, function, associated F-runtime group, signature)
- Data from the standard user program that are evaluated in the safety program
- Parameters of safety-related CPU-CPU communication (instructions, addresses, calling block and calling F-runtime group)
- Absolute addresses and names of the F-shared DB tags that can be accessed from the standard user program
- Information on hardware (used F-I/O, CPU version, addresses)
- Information on the printout (print date, number of pages)

Footer of the printouts

On the basis of the footer of the printout, you can find out:

- whether you printed out the "correct" project (specification of the project name and file path)
- whether the printout is consistent and belongs to the same safety program and the same version (the same collective F-signature in the footer of every page means that the printout belongs to the safety program with this collective F-signature).

The footer is added to the source code of the F-blocks only if the "All" option was selected for the safety program printout.

If F-blocks are printed by other means, the footer is omitted, and you can no longer easily identify whether the block printout belongs to the current safety program version.

Printing a migrated project

You can print a safety printout for a project migrated from *S7 Distributed Safety V5.4 SP5* only if it was compiled with *STEP 7 Safety Advanced V11* and, thus, the new program structure for safety programs (main safety block) was applied. Otherwise, the printout is not possible and you will receive a corresponding error message.

We recommend that you print out your project in *S7 Distributed Safety V5.4 SP5* before the migration.

10.7 Testing safety program

10.7.1 Overview of Testing the Safety Program

Complete function test or test of changes

After creating a safety program, you must carry out a complete function test in accordance with your automation task.

For changes made to a safety program that has already undergone a complete function test, only the changes need be tested.

Monitoring

In general, all read-only test functions (such as tag monitoring) are also available for safety programs and in safety mode.

Modifying

Modifying safety program data and write accesses to the safety program are possible only conditionally and in deactivated safety operation mode.

Simulation via *S7-PLCSIM*

You can test the safety program using *S7-PLCSIM*. You use *S7-PLCSIM* in the same way as for standard user programs.

You start the simulation with *S7-PLCSIM* using menu item "Online > Simulation > Start"

10.7.2 Deactivating Safety Mode

Introduction

The safety program generally runs in the F-CPU in safety mode. This means that all fault control measures are activated. The safety program cannot be modified during operation (in RUN mode) in safety mode. You must deactivate safety mode of the safety program to download changes to the safety program in RUN mode. Safety mode remains deactivated until the F-CPU is next switched from STOP to RUN mode.

Rules for deactivating safety mode

 **WARNING**

Because changes to the safety program can be made in RUN mode when safety mode is deactivated, you must take the following into account:

- Deactivation of safety mode is intended for test purposes, commissioning, etc. Whenever safety mode is deactivated, the safety of the system must be ensured by other organizational measures, such as monitored operation, manual safety shutdown, and access restrictions to certain areas.
- The deactivation of safety mode must be displayed.
In addition, the MODE tag in the F-shared-DB ("F_GLOBDB".MODE) is available, which you can evaluate for reading out the operating mode (1 = deactivated safety mode). Thus, not only is the deactivated safety mode displayed on the programming device or PC in the dialog box for deactivating safety mode, but it can also be indicated by means of an indicator light controlled by the standard user program or a message to an operator control and monitoring system generated by evaluating the "Deactivated Safety Mode" tag in the F-shared DB (Page 107).
- It must be possible to verify that safety mode has been deactivated. A log is required, if possible by recording messages to the operator control and monitoring system, but if necessary, through organizational measures. In addition, it is recommended that deactivation of safety mode be indicated on the operator control and monitoring system.
- Safety mode is deactivated across the F-CPU only. You must take the following into account for safety-related CPU-CPU communication: If the F-CPU with the SENDDP or SENDS7 instruction is in deactivated safety mode, you can no longer assume that the data sent by this F-CPU are generated safely. You must then implement organizational measures such as operation monitoring and manual safety shutdown to ensure safety in those portions of the system that are affected by the sent data. Alternatively, you must output fail-safe values instead of the received data in the F-CPU with RCVDP or RCVS7 (Page 510) or SENDMODE by evaluating SENDMODE. (S027)

Procedure for deactivating safety mode

To deactivate safety mode, follow these steps:

1. Open the *Safety Administration Editor* of the corresponding F-CPU.
2. Open the work area "General (Page 41)" in the area navigation.
3. Check to see whether the safety mode status is displayed as activated.

If so, continue with the next step; if not, stop the process, because safety mode is already deactivated or cannot be deactivated.

4. Click the "Disable safety mode" button.
5. Enter the password for the online safety program.

If you enter the correct password, another prompt will appear, which also contains the collective F-signature in the F-CPU. Check to see whether this is the collective F-signature you expected. If there is a match, acknowledge the dialog.

You can now download changes in the safety program to the F-CPU during operation (in RUN mode).

If the password is not valid, safety mode is not deactivated and remains active.

When individual F-blocks are downloaded, the condition "Disable safety mode" is listed automatically in the "Load preview" dialog. For this reason, it is not necessary to explicitly deactivate safety mode before every F-block download.

Note

If the collective F-signature or the passwords do not agree for the safety program online and offline, this means:

- The offline safety program was modified after the last downloading, or
 - An incorrect F-CPU was addressed. Check the latter based on the online collective F-signature.
-

Activating safety mode

Note

To activate safety mode, the F-CPU must be switched from STOP to RUN mode.

Switching the F-CPU from STOP to RUN mode always activates safety mode, even if the safety program has been modified or is not consistent. The MODE tag in the F-shared DB is set to "0". Keep this in mind when you evaluate the MODE tag to read out the operating mode.

If you have changed your safety program, but have not recompiled and downloaded it, the F-CPU can revert to STOP mode.

Evaluating safety mode/deactivated safety mode

If you wish to evaluate safety mode/deactivated safety mode in the safety program, you can evaluate the "MODE" tag in the F-shared DB (1 = deactivated safety mode). You access this tag with fully qualified access ("F_GLOBDB".MODE).

You can use this evaluation, for example, to passivate F-I/O when the safety program is in deactivated safety mode. To do so, assign the MODE tag in the F-shared DB to all PASS_ON tags in the F-I/O DBs of the F-I/O that you wish to passivate.

 **WARNING**

When the safety program is in deactivated safety mode, the "MODE" tag in the F-shared DB is also evaluated in deactivated safety mode.

Even if the F-I/O are passivated in deactivated safety mode as a result of evaluation of the "MODE" tag, system safety must be ensured in deactivated safety mode through other organizational measures, such as operation monitoring and manual safety shutdown.
(S028)

10.7.3 Testing the Safety Program

Introduction

In deactivated safety mode, certain fault control measures of the safety program are deactivated to enable online changes to be made to the safety program in RUN mode. In this way, safety program data can be changed using standard tools of *STEP 7 Professional* (watch table, monitoring in *program editor*).

Changing the safety program data by modifying tags

In addition to data in the standard user program, which can always be modified, you can modify the following data of a safety program in deactivated safety mode:

- Process image of F-I/O
- F-DBs (except DB for F-runtime group communication), instance DBs of F-FBs
- F-I/O DBs (for permitted signals, see F-I/O DB (Page 82))

Note

F-I/O can only be modified in RUN mode of the F-CPU. You must allocate a separate row in the watch table for each channel to be modified; this means, for example, that digital channels of data type BOOL cannot be modified on a byte-by-byte or word-by-word basis.

You can modify a maximum of 5 inputs/outputs from a watch table. You can use more than one monitoring table.

You cannot modify configured F-I/O in which no single channel or tag from the associated F-I/O DB has been used. Therefore, always use at least one tag from the associated F-I/O DB or at least one channel of the F-I/O to be modified in your safety program.

As a trigger point, you must set "Begin scan cycle" or "End scan cycle", either "permanently" or "once". However, note that regardless of the trigger point setting, requests to modify inputs (PII) of F-I/O always become effective before the main safety block is executed and requests to modify outputs (PIQ) always become effective after execution of the main safety block.

For inputs (PII), modify requests take priority over fail-safe value output, while for outputs (PIQ), fail-safe value output takes priority over modify requests. For outputs (channels) that are not activated in the properties for the F-I/O, modify requests affect the PIQ only, and not the F-I/O.

 **WARNING**

Constant modify requests for F-I/O can continue to remain active in the following cases:

- The connection between the programming device or PC and F-CPU is interrupted (e.g., due to unplugging of bus cable)
- if the watch table no longer responds

To delete constant modify requests in such cases, you must

- Switch the relevant F-CPU from STOP to RUN when the connection to the programming device or PC is interrupted,
or
- Perform a memory reset on the relevant F-CPU (*S029*)

Wiring test using tag table

You can carry out a wiring test for an input by changing an input signal and verifying whether or not the new value arrives in the PII.

You can carry out a wiring test for an output by changing the output with the Modify function and verifying whether the required actuator responds.

For the wiring test (both for an input and output), note that a safety program must be running on the F-CPU in which at least one channel of the F-I/O to be modified or one tag from the associated F-I/O DB has been used.

For F-I/O that can also be operated as standard I/O (e.g., S7-300 fail-safe signal modules), you can also carry out the wiring test for outputs using the Modify function in STOP mode by operating the F-I/O as standard I/O rather than in safety mode. When doing so, you must comply with the other rules for testing.

Note

A Modify function controlled by the F-system is only possible with the *STEP 7 Safety Advanced V11* optional package installed. If tags are modified by means of an operator control and monitoring system or modification occurs without an installed *STEP 7 Safety Advanced V11* optional package, the modification request can cause the F-CPU to go to STOP mode.

The test functions are selected using the standard tools of *STEP 7 Professional* (*program editor*, watch table). An attempt to modify a safety program in safety mode is rejected with a corresponding error message, or a dialog box for deactivating safety mode is provided. In certain circumstances, a modify request can cause the F-CPU to go to STOP mode.

Opening and changing F-blocks

Opening of an F-block online in the F-CPU is only possible with write protection in the *program editor*; that is, you cannot change an F-block directly in the F-CPU, even if safety mode is deactivated. After a successful password prompt, the F-block switches automatically to offline mode while opening (block icon is labeled accordingly in the project tree) and can be changed by you.

You must then compile the F-block and download it to the F-CPU. To do so, follow the procedure as outlined in "Downloading the Safety Program (Page 171)".

Modifying values in F-DBs

Values in F-DBs can only be modified online in the F-CPU. If the value is also to be changed offline, you must do this by editing the actual value offline and compiling the safety program.

Additional rules for testing

- Forcing is not possible for F-I/O.
- Setting breakpoints in the standard user program will cause the following errors in the safety program:
 - Expiration of F-cycle time monitoring
 - Error during communication with the F-I/O
 - Error during safety-related CPU-CPU communication
 - Internal CPU faults

If you nevertheless want to use breakpoints for testing, you must first deactivate safety mode (Page 190). This will result in the following errors:

- Error during communication with the F-I/O
- Error during safety-related CPU-CPU communication
- Changes in the configuration of F-I/O or safety-related CPU-CPU communication can only be tested after the hardware configuration has been saved and downloaded, and after the safety program has been compiled and downloaded to the F-CPU.

Note

If you use the watch table to test a safety program, this test does not detect all changes you make in parallel using other editors in the F-CPU.

For example, if the collective F-signature is changed through a revision/modification while safety mode is deactivated, the change may not be detected and an old collective F-signature may continue to be displayed.

In such cases, close the watch table and restart it in order to work with updated data.

Procedure for monitoring and modifying the safety program

Monitor or modify the required F-data and/or F-I/O from an open watch table or from the *Program editor* (program status) (for procedure, see *help on STEP 7 Professional*, "Testing user program").

1. For modifying, deactivate the safety mode (Page 190) in the automatically shown dialog.
2. Terminate existing modify requests after testing is complete before activating safety mode.

If the safety program does not behave as you wish during testing, you have the option of modifying the safety program in RUN mode (Page 197) and immediately continuing testing until the safety program behaves according to your requirements.

Testing the safety program with *S7-PLCSIM*

You can monitor and modify tags of your safety program in *S7-PLCSIM* and perform other write access operations in your safety program.

To use *S7-PLCSIM* (Page 171), you only have to download your consistent safety program to an *S7-PLCSIM*.

Note

- If you would like to modify tags in *S7-PLCSIM*, you must deactivate safety mode (Page 190) beforehand. Otherwise, *S7-PLCSIM* can go to STOP mode.
 - You can use the SENDDP, RCVDP, SENDS7, and RCVS7 instructions together with *S7-PLCSIM*. Note, however, that the instructions constantly signal "communication errors" when they are run in the simulation CPU.
-

10.7.4 Modifying the safety program in RUN mode

Introduction

Changes to the safety program during operation (in RUN mode) can only be made in deactivated safety mode (Page 190). You make changes to F-blocks offline in the *program editor* in the same way as for a standard program. F-blocks cannot be changed online.

Note

If you do **not** want to make changes to the safety program during operation, see Creating F-modules in FBD/LAD (Page 73).

Procedure for changing the safety program in RUN mode

To change the safety program, follow these steps:

1. Change the main safety block or F-FB and its associated instance DB, F-FC, or F-DB in the *Program editor*.
2. Download the changed F-block(s) to the F-CPU (for procedure, see Downloading the Safety Program (Page 171)). The entire program is then automatically compiled.
3. If safety mode is active, the "Load preview" dialog will prompt you to deactivate it and to enter the password for the safety program.

Note

When downloading in deactivated safety mode, you can only download the fail-safe blocks created by you (main safety blocks, F-FB, F-FC, or F-DB), F-application blocks, or standard blocks and their associated instance DBs. If you download automatically added F-blocks (F-SBs or automatically generated F-blocks and associated instance DBs, F-shared DB), the F-CPU can go to STOP mode or safety mode can be activated.

Therefore, always select individual blocks only when downloading in deactivated safety mode.

Sequence for downloading changes

Changes in the safety program in RUN mode when safety mode is deactivated can cause e.g., the status of an actuator to change as a result of program changes.

After changes, start by downloading the safety program and then the function of the standard user program monitored by the safety program.

Restrictions on safety-related CPU-CPU communication

During operation (in RUN mode), you cannot establish new safety-related CPU-CPU communication by means of new SENDDP/RCVDP or SENDS7/RCVS7 instructions.

To establish new safety-related CPU-CPU communication you must always download the relevant safety program consistently to the F-CPU while in STOP mode after inserting a new SENDDP, SENDS7, RCVDP, or RCVS7 instruction.

Restrictions on F-runtime group communication

You cannot make any changes to the safety-related communication between F-runtime groups in RUN mode. This means that you cannot assign, delete, or change any DBs for F-runtime group communication of a F-runtime group.

Following changes in the F-runtime group communication, you must always download the safety program consistently to the F-CPU while in STOP mode.

Restrictions on F-I/O access

If during operation (in RUN mode), you insert an F-I/O access to an F-I/O of which no single channel or tag from the associated F-I/O DB has yet been used in the safety program, the F-I/O access only becomes effective when the safety program is downloaded consistently to the F-CPU.

Changes in the standard user program

You can download changes in the standard user program when the F-CPU is in RUN mode, regardless of whether safety mode is activated or deactivated.

WARNING

In safety mode, access by means of the CPU password must not be authorized when making changes to the standard user program, since changes to the safety program can also be made. To rule out this possibility, you must configure **protection level "Write protection for fail-safe blocks"** and configure a password for the F-CPU. If only **one person** is authorized to change the standard user program **and** the safety program, then the protection level "Write protection" or "Read/write protection" should be configured so that other persons have only limited access or no access at all to the entire user program (standard and safety programs). (See also Access protection (Page 49)) (*S001*)

Procedure for applying changes to the safety program

If you download individual F-blocks to the F-CPU during operation (in RUN mode), the F-system blocks (F-SBs) and the automatically generated F-blocks are neither updated nor downloaded, resulting in an inconsistent safety program in the F-CPU. Use the following procedure to apply changes to the safety program:

1. Download the safety program consistently to the F-CPU, and activate safety mode by switching the F-CPU from STOP to RUN mode (for procedure, see Downloading the Safety Program (Page 171)).
2. Follow the steps described in Acceptance Test of Changes (Page 210).

10.7.5 Deleting the Safety Program

Deleting individual F-blocks

To delete an F-block, follow the same procedure as in *STEP 7 Professional*.

Remove all calls that you have used to call the safety program (Main_Safety).

Deleting an F-runtime group

See Deleting an F-runtime group (Page 72)

Remove all calls that you have used to call the safety program (Main_Safety).

Deleting the entire safety program for F-CPU's *with* plugged memory card (SIMATIC Micro Memory Card or Flash Card)

To delete an entire safety program, proceed as follows:

1. Delete all F-blocks (shown with yellow symbol) in the project tree.
2. Select the F-CPU in the *hardware and network editor*, and deactivate the "F-capability activated" option in the properties of the F-CPU.
3. Compile the hardware configuration.
The offline project no longer contains a safety program.
4. To delete a safety program on a Memory Card (SIMATIC Micro Memory Card or Flash Card), insert the Memory Card (SIMATIC Micro Memory Card or Flash Card) in the programming device or PC or in a SIMATIC USB prommer.
5. Select the menu command "Project> SIMATIC Card Reader> Show SIMATIC Card Reader" in the menu bar.
6. Open the "SIMATIC Card Reader" folder. You can now access the memory Card and can delete the safety program.

You can then download the offline standard user program to the F-CPU.

Deleting the entire safety program for F-CPU's *without* plugged Flash Card

To delete an entire safety program, proceed as follows:

1. Delete all F-blocks (shown with yellow symbol) in the project tree.
2. Select the F-CPU in the *hardware and network editor* and deactivate the "F-capability activated" option in the properties of the F-CPU.
3. Compile the hardware configuration.

The offline project no longer contains a safety program.

By performing a memory reset on the F-CPU (in the "Online tools" task card of the F-CPU), you can delete the safety program.

System Acceptance Test

11.1 Overview of System Acceptance Test

Introduction

When a system undergoes an acceptance test, all standards relevant to the specific application must be met. This also applies to systems that are not "subject to acceptance testing". For the acceptance test, you must consider the documents in the Certification Report (<http://support.automation.siemens.com/WW/view/en/49368678/134200>).

As a general rule, the acceptance test of an F-System is performed by an independent expert.

We recommend the following procedure for the preparation of the acceptance test.

Proof of correctness of components

In order for a system acceptance to be granted, you must recognize and document the correctness and of the individual components. For the documentation of the characteristics of the components, it is highly recommended to create a safety printout.

The following characteristics must be covered:

- Correctness of the safety program including hardware configuration
- Completeness of the safety printout
- Compliance of the used instructions with the TÜV certificate
- Correctness of the hardware configuration
- Correctness of the communication configuration
- Other characteristics such as software version, use of data from the standard user program

After the acceptance test, you should archive all relevant documents and also the project data so as to make the accepted project available as a reference for a subsequent change acceptance test.

Safety printout

The safety printout is the documentation of the project that provides supports for the acceptance test of the system.

11.2 Correctness of the safety program including hardware configuration

Verification/function test

Already during the creation, you will test (Page 189) your safety program and the associated hardware configuration. You must carry out tests with regard to the specification of your safety functions and document them before you seek acceptance for the plant.

In order to perform a code review of your safety program and document the accepted program code, the source code of all F-blocks is printed as a part of the safety printout, provided you have selected the "All" property for the printout.

The correct function of the safety program must be guaranteed by complete function tests before it may be used productively. You should archive the test reports along with the safety printout and the acceptance documents.

Times, for example, monitoring times (Page 523) and delay times can be verified through function tests (Page 178) only with very poor efficiency. You should check these times selectively to determine whether they are dimensioned correctly, for example, on the basis of the safety printout.

Some of these times are itemized specially in the safety printout, for example, the F-monitoring time (for communication between F-CPU and F-I/O) and the monitoring time of the safety-related CPU-CPU communication (parameter TIMEOUT). For the approximate determination of these monitoring times, the Excel file for response time calculation is available on the Internet

(<http://support.automation.siemens.com/WW/view/en/49368678/133100>).

Consistency of the safety program

Check in the "General information" section of the safety printout to determine whether the safety program was recognized as "consistent".

This is the case if the following signatures are identical:

- Collective F-signature ("General information" section, "Collective F-signature")
- "Signature of F-blocks with F-attribute" ("General information" section, "Current compilation")

The consistency of the safety program is required for the acceptance test. If the signatures are not identical, you should recompile the safety program and reprint the safety printout.

Password protection

Verify that a password was assigned for both the safety program and the F-CPU, if other organizational measures for access protection of the safety program were not taken.

For information about this, refer to section "General information" under "Access protection".

Additional notes

If required, review the "Notes" in section "General information", which provides additional notes on the safety program that must be observed or addressed.

11.3 Completeness of the safety printout

Introduction

If your safety program status including hardware configuration is ready for acceptance, you can carry out and document additional tests on the basis of the safety printout. The safety printout must be complete and belong to the safety program undergoing acceptance testing.

Procedure for creation of safety printout

To create the safety printout, follow the procedure described in Printing project data (Page 187).

In so doing, use the "All" property in order to include the source code of your F-blocks in the printout.

Checking printout for completeness

If you want to use an existing printout, whose completeness is not exactly known, you must check to determine whether the same collective F-signature is contained in the footer on all pages of the printout. This allows you to prove that all printed sheets belong to the same project.

In section "Supplementary information", you will find the number of pages in the safety printout, among other things. With this, you can prove that all pages of the safety printout are printout.

When you create the safety printout with the "All" option, the source code of all F-blocks will also be printed. The printout of this source code also contains the footer to enable you to easily assign the source code to a particular safety printout.

Association with the safety program

Check the "General information" section of the safety printout to determine whether the collective F-signature matches the collective F-signature of the safety program undergoing acceptance testing found in the work area of the Safety Administration Editor under "*General*". If there is no agreement, then the printout and safety program do not match.

11.4 Compliance of the used instructions with the TÜV certificate

Introduction

The *STEP 7 Safety Advanced V11* optional package contains LAD/FBD instructions that have been created and tested by SIEMENS and certified by TÜV for use in programming your safety program.

To check whether the instructions used match the TÜV certificate, these instructions are listed in the safety printout in section "System library elements used in safety program" along with their signatures and initial value signatures.

The same applies if you use shared libraries with F-blocks of other manufacturers.

Procedure

To check whether or not the utilized instructions match the TÜV certificate, proceed as follows:

1. Download the current Annex 1 of the report for the TÜV certificate "SIMATIC Safety" from the Internet (<http://support.automation.siemens.com/WW/view/en/49368678/134200>).

The versions of the instructions and F-blocks indicated in Annex 1 must match the version numbers in the safety printout. Otherwise, you did not receive the appropriate version of Annex 1 or a non-certified version of *STEP 7 Safety Advanced V11* is present.

2. Compare the signature and initial value signature with that specified in Annex 1 for each element itemized in the safety printout under "System library elements used in the safety program".
3. In case of discrepancies, recheck whether you have the correct versions.

If you have the correct versions, but the signature of the instructions are different, you should reinstall *STEP 7 Safety Advanced V11* and compile and download the project again, as the instructions may be corrupt.

11.5 Correctness of the hardware configuration

Introduction

The hardware configuration is an essential component of the project to be accepted. With the configuration of the hardware, you have undertaken settings that may influence the safety of signals. You can use the safety printout to document these settings in detail, in order to prove that you fulfill the safety requirements of your application.

The section "Hardware configuration of F-I/O" is available in the safety printout for this. This section consists of several tables:

- a table with information about the F-CPU and the range of F-destination addresses used
- an overview table with the F-I/O used
- a table for each F-I/O with detailed specifications, for example, the configured parameter values

Note

Note that will find F-I/O that you address via safety-related I-slave-slave communication in the safety printout of the I-slave and not in the printout of the assigned DP master.

Procedure for checking the correctness of the hardware configuration

To check the hardware configuration for correctness, proceed as follows:

1. Check in the "General information" section under "Protection" to determine whether the setting for access protection is permitted. Note the following warning.

Otherwise, the project must not be accepted, because the safety program in the F-CPU is not protected against unauthorized accesses.

 WARNING
--

<p>In safety mode, access by means of the CPU password must not be authorized when making changes to the standard user program, since changes to the safety program can also be made. To rule out this possibility, you must configure protection level "Write protection for fail-safe blocks" and configure a password for the F-CPU. If only one person is authorized to change the standard user program and the safety program, then the protection level "Write protection" or "Read/write protection" should be configured so that other persons have only limited access or no access at all to the entire user program (standard and safety programs). (See also Access protection (Page 49).) (S001)</p>

2. Check in the "F-hardware configuration" section to verify the uniqueness of the PROFIsafe destination addresses.

The first table outputs the address area that is used by the F-CPU.

Note

Recommendation for the assignment of address ranges

For each F-CPU that you use in your system, assign a separate address area within the network in such a way that none of them overlap. As a result, you only have to check for the absence of overlapping in the respective safety printouts of the F-CPU. If the areas of several F-CPU overlap one another, you must check the respective intersection area individually for each PROFIsafe destination address for uniqueness.

 WARNING
<p>The following applies to PROFIBUS only subnets:</p> <p>The PROFIsafe destination address and, thus, the switch setting on the address switch of the F-I/O must be unique network-wide* and station-wide** (system-wide). For S7-300 F-SMs and ET 200S, ET 200eco, ET 200pro, and ET 200iSP F-modules, you can assign a maximum of 1022 different PROFIsafe destination addresses.</p> <p>Exception: The F-I/O in different I-slaves may be assigned the same PROFIsafe destination address, as they are only addressed within the station, that is, by the F-CPU in the I-slave.</p> <p>The following applies to Ethernet subnets and hybrid configurations of PROFIBUS and Ethernet subnets:</p> <p>The PROFIsafe destination address and, thus, the address switch setting on the F-I/O have to be unique only*** within the Ethernet subnet, including all lower-level PROFIBUS subnets, and station-wide** (system-wide). For S7-300 F-SMs and ET 200S, ET 200eco, ET 200pro, and ET 200iSP F-modules, you can assign a maximum of 1022 different PROFIsafe destination addresses.</p> <p>Exception: The F-I/O in different I-slaves/I-Devices may be assigned the same PROFIsafe destination address, as they are only addressed within the station, that is, by the F-CPU in the I-slave/I-Device.</p> <p>The networked nodes of an Ethernet subnet are characterized by having IP addresses with the same subnet address, i.e., the IP addresses match in the digits that have the value "1" in the subnet mask.</p> <p>Example:</p> <p>IP address: 140.80.0.2.</p> <p>Subnet mask: 255.255.0.0 = 11111111.11111111.00000000.00000000</p> <p>Meaning: Bytes 1 and 2 of the IP address define the subnet; subnet address = 140.80. (S002)</p>

* A network consists of one or more subnets. "Network-wide" means beyond the boundaries of the subnet.

** "Station-wide" means for one station in the *hardware and network editor* (for example, a SIMATIC S7-300 or also an I-slave)

*** Across Ethernet subnets, excluding cyclic PROFINET IO communication

3. Check the safety-related parameters of all configured F-I/O.

You will find these parameters in the "Hardware configuration of F-I/O" section in the detailed tables for the F-I/O.

The table consists of two parts:

- Left part is for parameters that relate to the F-I/O themselves and their access function in the safety program ("Module data")
- Right part is for the parameters of the individual channels ("Channel parameters")

These parameters must be set as prescribed by the safety requirements of your application.

When using fail-safe DP standard slaves/standard I/O devices, note the relevant documents for the possible additional safety-related (technical) parameters.

Note

F-I/O that are to be assigned the same safety-related parameters (except for PROFIsafe addresses) can be copied during configuration. Except for the PROFIsafe addresses, you no longer have to check the safety-related parameters individually. It is sufficient to compare the "Signature of F-parameters (without addresses)" in the "Hardware configuration of the F-I/O" section in the overview table. This also applies to fail-safe DP standard slaves/standard I/O devices.

11.6 Correctness of the communication configuration

Introduction

Safety-related communication is based on the mechanisms of the standard communication of *STEP 7 Professional*. To detect errors in the standard communication configuration, safety-related communication connections between F-CPU must be protected additionally. For the acceptance test, you must document the restrictions (uniqueness) resulting from the protection.

For this purpose, the "Parameters for safety-related CPU-CPU-communication" section is available in the safety printout. In this section, there are up to two tables (for communication via PROFIBUS DP or PROFINET IO and for communication via S7 connections).

Procedure for testing for correctness of the communication configuration

To check the communication configuration for correctness, proceed as follows:

- In the "Safety-related CPU-CPU communication via PROFIBUS DP or PROFINET IO" table, check to determine whether you have assigned a unique DP_DP_ID parameter throughout the network for all safety-related communication connections (e.g., master-master, master-I-slave and IO Controller-IO Controller communication).

This means that sender and receiver of the relevant communication connection must have the same values for DP_DP_ID. All other communication connections in the entire network must not have these values.

- In the "Safety-related CPU-CPU communication via S7 connections" table, check to determine whether you have assigned a unique R_ID parameter throughout the network for all safety-related communication via S7 connections.

11.7 Other characteristics

Introduction

In addition, you must check a few more characteristics that are also relevant for the acceptance test of the project.

Validity check for data transfer from the standard program to the safety program

Check to determine whether a validity check was programmed for all data transferred from the standard user program to the safety program. For this purpose, the "Data from the standard user program" section lists all signals that you are using in the safety program.

Checking the program version

Check whether the version of *STEP 7 Safety Advanced V11* used to create the printout (in footer of printout) is greater than or equal to the version used to compile the safety program. The latter version can be found in the "General information" section of the safety printout under "Versions used".

See also

Data Transfer from Standard User Program to Safety Program (Page 108)

11.8 Acceptance Test of Changes

Introduction

In general, you can adopt the same approach for the acceptance test of changes as the initial acceptance test (see Overview of System Acceptance Test (Page 201)).

To avoid undergoing a new acceptance test of the entire system in the event of minor changes, *STEP 7 Safety V11* helps you to identify those parts of your safety program that have changed.

For an acceptance test of changes, it is sufficient to test the following:

- the changed or newly added F-blocks
- the changed or newly added instructions and F-system blocks
- the safety-related parameters of the changed or newly added F-I/O

Then perform a function test of the changes.

Detection of changes in the safety program

To detect changes in the safety program, proceed as follows:

1. Perform an offline-offline comparison between the changed safety program and the accepted safety program (see Comparing Safety Programs (Page 183)). Use filter setting "Compare only F-blocks relevant for certification". This limits the output of the comparison to exactly those F-blocks that must be considered for the acceptance test of changes.

The status of the comparison enables you to identify which F-blocks were changed.

2. Determine the changes in the F-blocks you created (main safety blocks/F-FBs/F-FCs/F-DBs).

Detection of safety-related changes in the configuration of the F-I/O

There are two possible ways of detecting safety-relevant changes in the configured F-I/O:

- Comparison in the comparison editor
- Comparison based on two safety printouts

Comparison in the comparison editor

If you have saved the accepted project, you can also perform an offline-offline comparison of the changed project and the accepted project to detect changes in the configuration of the F-I/O (see Comparing Safety Programs (Page 183)).

1. Navigate in the comparison result to the "System blocks > STEP 7 Safety > F-I/O DBs" folder. All data blocks listed in this folder are F-I/O-DBs and are each assigned to a F-I/O.
2. The names of the F-I/O-DBs (Page 88) are automatically assigned. If you have changed the names of the F-I/O-DBs, you can find their numbers in the module data of the safety printout.

If the F-I/O-DBs in the comparison result are identical, this means that the configuration of the assigned F-I/O was not changed (assuming that your current safety program is consistent and compiled). In this way, you can quickly identify which F-I/O have changed.

3. If you have found changed F-I/O, you can check the changed parameters in the safety printout as described above. In so doing, always check the DB number to ensure that the DB name did not lead you to the wrong F-I/O.

Comparison based on two safety printouts

In the overview table of the utilized F-I/O in section "Hardware configuration of F-I/O" of the safety printout, compare the parameter CRCs of all F-I/O/DP standard slaves/standard I/O devices with those in the safety printout of the accepted project.

If the "F-parameter signatures (w/o addresses)" is different for an F-I/O, this indicates the existence of a safety-related change in the configuration of this F-I/O, e.g., including PROFIsafe addresses.

If this information is identical, only the PROFIsafe addresses were changed. In this case, you do not have to check the other safety-related parameters of the F-I/O individually. Pay attention that the uniqueness of the PROFIsafe destination addresses of all configured F-I/O will continue to be ensured.

Detecting changes of the start addresses of F-I/O

You can detect changes in the start addresses of F-I/O in the output of the program comparison, as well. Note that the names of the F-I/O DBs contain the address in this case, provided you have not changed the names manually. In order to detect changes, it is necessary to distinguish between the following two cases:

- You have not changed the name of the F-I/O DB; the name represents the same F-I/O in both program versions.

If you change the start address of an F-I/O, this also changes the name of the F-I/O DB. If you then compare the program with its previous version, the newly named F-I/O does not exist in the old program version and the formerly named F-I/O does not exist in the new program version. The comparison log contains two lines in which the names are different. Both I-DBs represent the module whose start address has changed.
- You have changed the name of the F-I/O DB.

In this case, the F-I/O DB appears in the comparison output as a changed F-I/O, just the same as if other module parameters have changed.

Operation and Maintenance

12.1 Notes on Safety Mode of the Safety Program

Introduction

Pay attention to the following important notes on safety mode of the safety program.

Use of simulation devices/simulation programs

 WARNING
<p>If you operate simulation devices or simulation programs that generate safety message frames, e.g., based on PROFIsafe, and make them available to the SIMATIC Safety F-system via the bus system (such as PROFIBUS DP or PROFINET IO), you must ensure the safety of the F-system using organizational measures, such as operational monitoring and manual safety shutdown.</p> <p>If you use the S7-PLCSIM (Page 189) to simulate safety programs, these measures are not necessary because S7-PLCSIM cannot establish an online connection to a real S7 component.</p> <p>Note, for example, that a protocol analyzer is not permitted to perform any function that reproduces recorded message frame sequences with correct time behavior. (S030)</p>

STOP by means of programming device or PC, mode selector, or communication function

 WARNING
<p>Switching from STOP to RUN mode using a programming device or PC interface, mode selector, or communication function is not locked. For example, only one keystroke is necessary to switch from STOP to RUN mode on a programming device or PC interface. For this reason, a STOP that you have set by means of a programming device or PC, mode selector, or communication function must not be regarded as a safety condition.</p> <p>Therefore, you must program startup protection (see Programming startup protection (Page 76)). (S031)</p>

Switch F-CPU to STOP mode with the "STP" Instruction

 **WARNING**

A STOP state initiated by the "STP" instruction can be canceled very easily (and unintentionally) from the programming device or PC. For this reason, a STOP initiated by the STP instruction is not a safety-related STOP.

Therefore, you must program startup protection (see Programming startup protection (Page 76)). (S032)

12.2 Replacing Software and Hardware Components

Replacement of software components

When software components are replaced on your programming device or PC, e.g., in case of a new version of *STEP 7 Professional*, you must observe the notes regarding upward and downward compatibility in the documentation and readme files for these products.

Replacement of hardware components

Hardware components for *SIMATIC Safety* (F-CPU, F-I/O, batteries, etc.) are replaced in the same way as in standard automation systems.

Removing and inserting F-I/O during operation

It is possible to remove and insert F-I/O during operation, as with standard F-I/O. However, be aware that replacing an F-I/O module during operation can cause a communication error in the F-CPU.

You must acknowledge the communication error in your safety program at the ACK_REI tag of the F-I/O DB (Page 82). Without an acknowledgment, the F-I/O will remain passivated.

CPU firmware update

Check of the CPU operating system for F-approval: When using a new CPU operating system (firmware update), you must check to see if the CPU operating system you are using is approved for use in an F-system.

The minimum CPU operating system versions with guaranteed F-capability are specified in the appendix of the Certification Report. This information and any notes on the new CPU operating system must be taken into account.

Firmware update for interface module

When using a new operating system for an interface module, e.g., IM 151-1 HIGH FEATURE ET 200S (Firmware update), you must observe the following:

If you have selected the "Activate firmware after update" option for the firmware update (see *help on STEP 7 Professional*, "Online & Diagnostics"), the IM will be automatically reset following a successful downloading operation and will then run on the new operating system. Note that the firmware update for interface modules during operation generates a communication error in the F-CPU.

You must acknowledge the communication error in your safety program in the ACK_REI tag of the F-I/O DB (Page 82) or, alternatively, by using the "ACK_GL (Page 295)" instruction. Without an acknowledgment, the F-I/O will remain passivated.

PFD and PFH values and preventive maintenance (proof test)

You will find a list of the failure probability values (PFD and PFH values) and the proof test intervals for components that can be used in *SIMATIC Safety* on the Internet (<http://support.automation.siemens.com/WW/view/en/49368678/133300>).

See also

Installation/uninstallation of the STEP 7 Safety Advanced V11 optional package (Page 21)

12.3 Guide to Diagnostics

Introduction

Here you find a compilation of diagnostic capabilities that can be evaluated for your system when an error occurs. Most of the diagnostic capabilities are the same as those in standard automation systems. The sequence of steps represents one recommendation.

Steps for evaluating diagnostic capabilities

Step	Procedure	Reference
1	<p>Evaluate LEDs on the hardware (F-CPU, F-I/O):</p> <ul style="list-style-type: none"> • BUSF LED on the F-CPU: flashes when a communication error occurs on PROFIBUS DP/PROFINET IO; illuminates when OB 85 and OB 121 are programmed, and a programming error occurs (e.g., instance DB is not loaded) • STP LED on the F-CPU: illuminates when the F-CPU is in STOP mode • Fault LEDs on the F-I/O: SF-LED (group error LED) illuminates if any fault occurs in the individual F-I/O 	<p><i>Manuals for F-CPU and F-I/O</i></p>
2	<p>Evaluate diagnostic buffer of the modules:</p> <p>You read the diagnostic buffer of a module (F-CPU, F-I/O, CP) in its online and diagnostic view in the "Diagnostic buffer" group under the "Online & Diagnostics" folder.</p>	<p><i>Help on STEP 7 Professional and manuals for the F-CPU and F-I/O</i></p>
3	<p>Evaluate stacks of the F-CPU:</p> <p>when the F-CPU is in STOP mode, read the following successively:</p> <ul style="list-style-type: none"> • Block stack: Check whether STOP mode of the F-CPU was triggered by an F-block of the safety program • Interruption stack • Local data stack 	<p><i>Help on STEP 7 Professional</i></p>
4	<p>Evaluate diagnostic tag of the F-I/O DB using testing and commissioning functions, by means of an operator control and monitoring system, or in the standard user program:</p> <p>Evaluate the DIAG tag in the F-I/O DB</p>	<p>F-I/O access (Page 79)</p>

Step	Procedure	Reference
5	<p>Evaluate diagnostic outputs of the instance DBs of instructions using testing and commissioning functions, by means of an operator control and monitoring system, or in the standard user program:</p> <ul style="list-style-type: none"> • For MUTING, EV1oo2DI, TWO_H_EN, MUT_P, ESTOP1, FDBBACK, SFDOOR, evaluate in the assigned instance: <ul style="list-style-type: none"> – Output DIAG • Evaluate the following for SENDDP or RCVDP in the assigned instance DB: <ul style="list-style-type: none"> – Output RET_DPRD/RET_DPWR – Output DIAG • Evaluate the following for SENDS7 or RCVS7 in the assigned instance DB: <ul style="list-style-type: none"> – Output STAT_RCV – Output STAT_SND – Output DIAG 	Instructions (Page 369)

Tip on RET_DPRD/RET_DPWR

The diagnostic information of the RET_DPRD/RET_DPWR parameters correspond to the diagnostic information of the RETVAL parameter of the "DPRD_DAT" and "DPWR_DAT" instructions. You will find the description in the help on *STEP 7 Professional* for the "DPRD_DAT" and "DPWR_DAT" instructions.

Tip: STAT_RCV and STAT_SND

The diagnostic information of the parameter STAT_RCV corresponds to the diagnostic information of the parameter STATUS of the instruction "URCV". The diagnostic information of the parameter STAT_SND corresponds to the diagnostic information of the parameter STATUS of the instruction "USEND". You will find the description in the Help *STEP 7 Professional* for the instruction "UCRV" or "USEND" .

STEP 7 Safety Advanced V11 Instructions

13.1 Overview of instructions

Overview of instructions for the safety program

When programming an F-block, all instructions that you can use for programming an F-block in LAD or FBD are contained in the "Instructions" task card.

In addition to the instructions that are familiar to you from programming a standard block, there are also special safety functions, e.g., for two-hand monitoring, discrepancy analysis, muting, emergency STOP, safety door monitoring, and feedback monitoring.

All instructions for LAD and FBD are explained in the following.

Note the following

Note

Preconnection of enable input EN or evaluation of enable output ENO is not possible.

13.2 Instructions - LAD

13.2.1 General

13.2.1.1 New network (STEP 7 Safety Advanced V11)

Requirement

An F-block is open.

Procedure

To insert a new network, follow these steps:

1. Select the network after which you want to insert a new network.
2. Select the "Insert network" command in the shortcut menu.

Note

If you insert an element into the last empty network of the F-block in an LAD program, a new empty network is automatically inserted below it.

Result

A new empty network is inserted into the F-block.

13.2.1.2 Empty box (STEP 7 Safety Advanced V11)

Requirement

A network is available.

Procedure

To insert an LAD instruction into a network using an empty box, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "Basic instructions > General > Empty box".
3. Use a drag-and-drop operation to move the "Empty box" element to the desired place in the network.
4. Hover the cursor over the yellow triangle in the top right corner of the empty box.
A drop-down list is displayed.
5. Select the required instruction from the drop-down list.

If the instruction acts as a function block (FB) within the system, the "Call options" dialog opens. In this dialog, you can create an instance data block for the function block, either as a single instance or multi-instance, in which data of the inserted instruction are stored. After its creation, the new instance data block can be found in the "Program resources" folder in the project tree under "Program blocks > System blocks". If you have selected "multi-instance", you will find it in the block interface in the "Static" section.

Result

The empty box is changed to the appropriate instruction. Placeholders are inserted for the parameters.

13.2.1.3 Open branch (STEP 7 Safety Advanced V11)

Description

Use branches to program parallel connections with the Ladder Logic (LAD) programming language. Branches are inserted into the main current path. You can insert several contacts into the branch, thereby creating a parallel connection from series connections. You can program complex ladder diagrams in this way.

Requirement

- A network is available.
- The network contains elements.

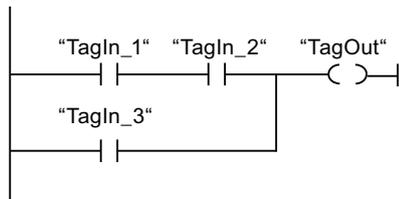
Procedure

To insert a new branch in a network, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "Basic instructions > General > Open branch".
3. Use a drag-and-drop operation to move the element to the desired place in the network.
4. If you want to connect the new branch directly to the power rail, drag the element to the power rail.

Example

The following figure provides an example of how to use branches:



13.2.1.4 Close branch (STEP 7 Safety Advanced V11)

Description

Branches must be closed again at suitable places. When a branch is closed, any necessary empty elements are added. If necessary, branches will be arranged so that they do not cross each other.

Requirement

A branch is available.

Procedure

To close an open branch, follow these steps:

1. Select the open branch.
2. Press and hold down the left mouse button.
3. A dashed line will appear as soon as the cursor is moved.
4. Drag the dashed line to a suitable place on the network. Permissible connections are indicated by green lines.
5. Release the left mouse button.

13.2.2 Bit logic operations

13.2.2.1 --| |--: Normally open contact (STEP 7 Safety Advanced V11)

Description

The activation of the normally open contact depends on the signal state of the associated operand. If the operand has signal state "1," the normally open contact is closed. Power flows from the left power rail through the normally open contact into the right power rail and the signal state at the output of the instruction is set to "1".

If the operand has signal state "0," the normally open contact is not activated. The power flow to the right power rail is interrupted and the signal state at the output of the instruction is reset to "0".

Two or more normally open contacts are linked bit-by-bit by AND when connected in series. With a series connection, power flows when all contacts are closed.

The normally open contacts are linked by OR when connected in parallel. With a parallel connection, power flows when one of the contacts is closed.

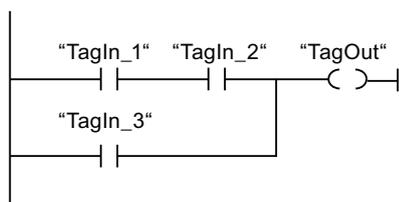
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand>	Input	BOOL	Operand whose signal state is queried.

Example

The following example shows how the instruction works:



Operand "TagOut" is set when one of the following conditions is fulfilled:

- Operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state at operand "TagIn_3" is "1".

13.2.2.2 ---| / |---: Normally closed contact (STEP 7 Safety Advanced V11)

Description

The activation of the normally closed contact depends on the signal state of the associated operand. If the operand has signal state "1", the normally closed contact is opened and the signal state at the output of the instruction is reset to "0".

If the operand has signal state "0", the normally closed contact is not activated and the signal state at the output of the instruction is set to "1".

Two or more normally closed contacts are linked bit-by-bit by AND when connected in series. With a series connection, power flows when all contacts are closed.

The normally closed contacts are linked by OR when connected in parallel. With a parallel connection, power flows when one of the contacts is closed.

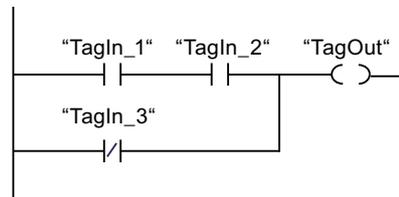
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand>	Input	BOOL	Operand whose signal state is queried.

Example

The following example shows how the instruction works:



Operand "TagOut" is set when one of the following conditions is fulfilled:

- Operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state at operand "TagIn_3" is "0".

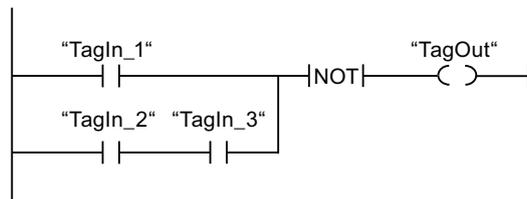
13.2.2.3 --|NOT|--: Invert RLO (STEP 7 Safety Advanced V11)

Description

You can use the "Invert RLO" instruction to invert the signal state of the result of logic operation (RLO). When the signal state is "1" at the input of the instruction, the output of the instruction has the signal state "0". When the signal state is "0" at the input of the instruction, the output has the signal state "1".

Example

The following example shows how the instruction works:



Operand "TagOut" is reset when one of the following conditions is fulfilled:

- Operand "TagIn_1" has signal state "1".
- Operands "TagIn_2" and "TagIn_3" have signal state "1".

13.2.2.4 ---()---: Assignment (STEP 7 Safety Advanced V11)

Description

You can use the "Assignment" instruction to set the bit of a specified operand. When the result of logic operation (RLO) at the input of the coil is "1," the specified operand is set to signal state "1". When the signal state is "0" at the input of the coil, the bit of the specified operand is reset to "0".

The instruction does not influence the RLO. The RLO at the input of the coil is sent immediately to the output.

The "Assignment" instruction can be placed at any position in the network.

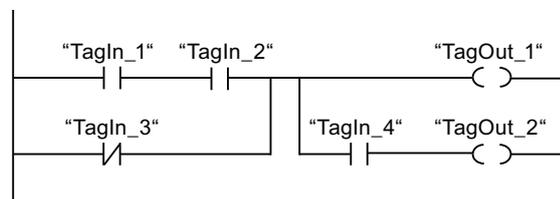
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand>	Output	BOOL	Operand to which the RLO is assigned.

Example

The following example shows how the instruction works:



Operand "TagOut_1" is set when one of the following conditions is fulfilled:

- Operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state at operand "TagIn_3" is "0".

Operand "TagOut_2" is set when one of the following conditions is fulfilled:

- Operands "TagIn_1" and "TagIn_2" as well as operand "TagIn_4" have signal state "1".
- The signal state of operand "TagIn_3" is "0" and the signal state of operand "TagIn_4" is "1".

13.2.2.5 ---(R)---: Reset output (STEP 7 Safety Advanced V11)

Description

You can use the "Reset output" instruction to reset the signal state of a specified operand to "0".

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If power flows to the coil (RLO is "1"), the specified operand is set to "0". If the result of logic operation at the input of the coil is "0" (no signal flow to the coil), the signal state of the specified operand remains unchanged.

The instruction does not influence the RLO. The RLO at the input of the coil is sent immediately to the output of the coil.

Note

If you want to use a formal parameter of an F-FB/F-FC for the operands of the instruction then this must be declared as an in/out parameter.

The F-CPU can go to STOP if this caution is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
 - "Data corruption in the safety program prior to output to partner F-CPU"
 - "Safety program: internal CPU fault; internal error information: 404"
-

Note

If the operand area "local data" is used for the operand of the instruction then the local data bit used must be initialized beforehand.

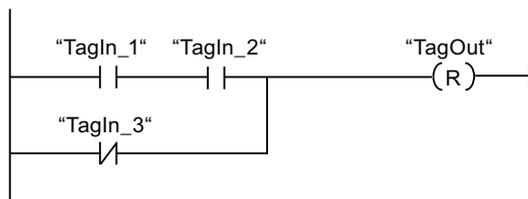
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand>	Output	BOOL	Operand that is reset when RLO = "1".

Example

The following example shows how the instruction works:



Operand "TagOut" is reset when one of the following conditions is fulfilled:

- Operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of operand "TagIn_3" is "0".

13.2.2.6 ---(S)---: Set output (STEP 7 Safety Advanced V11)

Description

You can use the "Set output" instruction to set the signal state of a specified operand to "1".

The instruction is only executed if the result of logic operation (RLO) at the input of the coil is "1". If power flows to the coil (RLO is "1"), the specified operand is set to "1". If the result of logic operation at the input of the coil is "0" (no signal flow to the coil), the signal state of the specified operand remains unchanged.

The instruction does not influence the RLO. The RLO at the input of the coil is sent directly to the output of the coil.

Note

The instruction is not executed if it is applied to an output of an F-I/O that is passivated (e.g., during startup of the F-system). Therefore, it is preferable to access outputs of the F-I/O using only the "Assignment" instruction.

You can evaluate whether an F-I/O or channels of an F-I/O are passivated in the associated F-I/O DB.

Note

If you want to use a formal parameter of an F-FB/F-FC for the operand of the instruction then this must be declared as an in/out parameter.

The F-CPU can go to STOP if this caution is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
 - "Data corruption in the safety program prior to output to partner F-CPU"
 - "Safety program: internal CPU fault; internal error information: 404"
-

Note

If the operand area "local data" is used for the operand of the instruction then the local data bit used must be initialized beforehand.

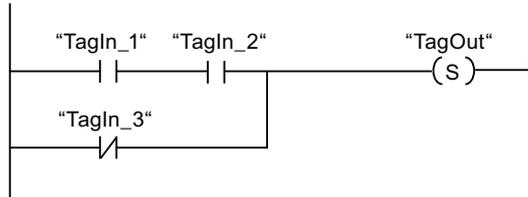
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand>	Output	BOOL	Operand that is set when RLO = "1".

Example

The following example shows how the instruction works:



Operand "TagOut" is set when one of the following conditions is fulfilled:

- Operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of operand "TagIn_3" is "0".

13.2.2.7 SR: Set/reset flip-flop (STEP 7 Safety Advanced V11)**Description**

You can use the "Set/reset flip-flop" instruction to set or reset the bit of the specified operand based on the signal state of inputs S and R1. If the signal state at input S is "1" and the signal state at input R1 is "0", the specified operand is set to "1". If the signal state at input S is "0" and the signal state at input R1 is "1", the specified operand is reset to "0".

Input R1 takes priority over input S. If the signal state is "1" at the two inputs S and R1, the signal state of the specified operand is reset to "0".

The instruction is not executed if the signal state at the two inputs S and R1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

Note

If you want to use a formal parameter of an F-FB/F-FC for the operand of the instruction then this must be declared as an in/out parameter.

The F-CPU can go to STOP if this caution is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"
- "Safety program: internal CPU fault; internal error information: 404"

Note

You cannot use the "process image input", "process image output", or "bit memory" operand areas for the operands of the instruction.

If the operand area "local data" is used for the operand of the instruction then the local data bit used must be initialized beforehand.

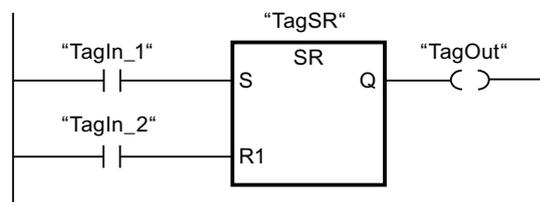
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
S	Input	BOOL	Enable setting
R1	Input	BOOL	Enable resetting
<Operand>	Output	BOOL	Operand that is set or reset.
Q	Output	BOOL	Signal state of the operand

Example

The following example shows how the instruction works:



Operands "TagSR" and "TagOut" are set when the following conditions are fulfilled:

- Operand "TagIn_1" has signal state "1".
- Operand "TagIn_2" has signal state "0".

Operands "TagSR" and "TagOut" are reset when the following conditions are fulfilled:

- Operand "TagIn_1" has signal state "0" and operand "TagIn_2" has signal state "1".
- Both operands "TagIn_1" and "TagIn_2" have signal state "1".

13.2.2.8 RS: Reset/set flip-flop (STEP 7 Safety Advanced V11)

Description

You can use the "Reset/set flip-flop" instruction to reset or set the bit of the specified operand based on the signal state of inputs R and S1. When the signal state is "1" at input R and "0" at input S1, the specified operand is reset to "0". When the signal state is "0" at input R and "1" at input S1, the specified operand is set to "1".

Input S1 takes priority over input R. If the signal state is "1" at the two inputs R and S1, the signal state of the specified operand is set to "1".

The instruction is not executed if the signal state at the two inputs R and S1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

Note

If you want to use a formal parameter of an F-FB/F-FC for the operand of the instruction then this must be declared as an in/out parameter.

The F-CPU can go to STOP if this caution is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
 - "Data corruption in the safety program prior to output to partner F-CPU"
 - "Safety program: internal CPU fault; internal error information: 404"
-

Note

You cannot use the "process image input", "process image output", or "bit memory" operand areas for the operands of the instruction.

If the operand area "local data" is used for the operand of the instruction then the local data bit used must be initialized beforehand.

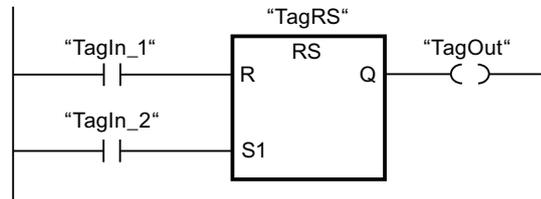
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
R	Input	BOOL	Enable resetting
S1	Input	BOOL	Enable setting
<Operand>	Output	BOOL	Operand that is reset or set.
Q	Output	BOOL	Signal state of the operand

Example

The following example shows how the instruction works:



Operands "TagRS" and "TagOut" are reset when the following conditions are fulfilled:

- Operand "TagIn_1" has signal state "1".
- Operand "TagIn_2" has signal state "0".

Operands "TagRS" and "TagOut" are set when the following conditions are fulfilled:

- Operand "TagIn_1" has signal state "0" and operand "TagIn_2" has signal state "1".
- Operands "TagIn_1" and "TagIn_2" have signal state "1".

13.2.2.9 --|P|--: Scan operand for positive signal edge (STEP 7 Safety Advanced V11)

Description

You can use the "Scan operand for positive signal edge" instruction to determine if there is a change from "0" to "1" in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous query saved in <Operand2>. If the instruction detects a change in the result of logic operation from "0" to "1", there is a positive, rising edge.

If a rising edge is detected, the output of the instruction has signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Enter the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Enter the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the edge memory bit would be overwritten. This would influence edge evaluation and the result would no longer be unequivocal.

Note

If you want to use a formal parameter of an F-FB/F-FC for the edge memory bit <Operand2> of the instruction then this must be declared as an in/out parameter.

The F-CPU can go to STOP if this caution is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
 - "Data corruption in the safety program prior to output to partner F-CPU"
 - "Safety program: internal CPU fault; internal error information: 404"
-

Note

You cannot use the "process image input", "process image output", or "bit memory" operand areas for the edge memory bit <Operand2> of the instruction.

If operand area "local data" is used for the edge memory bit <Operand2> of the instruction then the local data bit used must be initialized beforehand.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand1>	Input	BOOL	Signal to be queried
<Operand2>	InOut	BOOL	Edge memory bit in which the signal state of the previous query is saved.

13.2.2.10 --|N|--: Scan operand for negative signal edge (STEP 7 Safety Advanced V11)

Description

You can use the "Scan operand for negative signal edge" instruction to determine if there is a change from "1" to "0" in the signal state of a specified operand. The instruction compares the current signal state of <Operand1> with the signal state of the previous query saved in <Operand2>. If the instruction detects a change in the result of logic operation from "1" to "0", there is a negative, falling edge.

If a falling edge is detected, the output of the instruction has signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Enter the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Enter the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the edge memory bit would be overwritten. This would influence edge evaluation and the result would no longer be unequivocal.

Note

If you want to use a formal parameter of an F-FB/F-FC for the edge memory bit <Operand2> of the instruction then this must be declared as an in/out parameter.

The F-CPU can go to STOP if this caution is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"
- "Safety program: internal CPU fault; internal error information: 404"

Note

You cannot use the "process image input", "process image output", or "bit memory" operand areas for the edge memory bit <Operand2> of the instruction.

If operand area "local data" is used for the edge memory bit <Operand2> of the instruction then the local data bit used must be initialized beforehand.

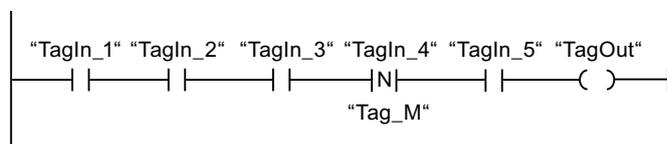
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand1>	Input	BOOL	Signal to be queried
<Operand2>	InOut	BOOL	Edge memory bit in which the signal state of the previous query is saved.

Example

The following example shows how the instruction works:



Operand "TagOut" is set when the following conditions are fulfilled:

- Operands "TagIn_1", "TagIn_2", and "TagIn_3" have signal state "1".
- There is a falling edge at operand "TagIn_4". The signal state of the previous query is saved at edge memory bit "Tag_M".
- The signal state of operand "TagIn_5" is "1".

13.2.2.11 P_TRIG: Scan RLO for positive signal edge (STEP 7 Safety Advanced V11)

Description

You can use the "Scan RLO for positive signal edge" instruction to query a change in the signal state of the result of logic operation from "0" to "1". The instruction compares the current signal state of the result of logic operation (RLO) with the signal state of the previous query, which is saved in the edge bit memory (<Operand>). If the instruction detects a change in the RLO from "0" to "1", there is a positive, rising edge.

If a rising edge is detected, the output of the instruction has signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the edge memory bit would be overwritten. This would influence edge evaluation and the result would no longer be unequivocal.

Note

If you want to use a formal parameter of an F-FB/F-FC for the edge memory bit <Operand> of the instruction then this must be declared as an in/out parameter.

The F-CPU can go to STOP if this caution is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
 - "Data corruption in the safety program prior to output to partner F-CPU"
 - "Safety program: internal CPU fault; internal error information: 404"
-

Note

You cannot use the "process image input", "process image output", or "bit memory" operand areas for the edge memory bit <Operand> of the instruction.

If operand area "local data" is used for the edge memory bit <Operand> of the instruction then the local data bit used must be initialized beforehand.

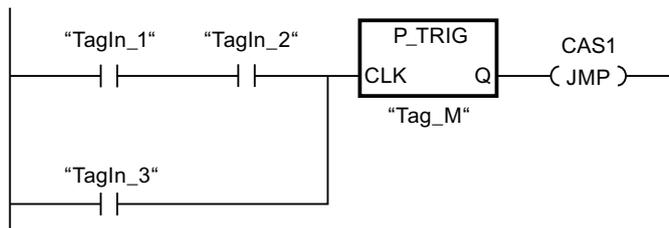
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
CLK	Input	BOOL	Current RLO
<Operand>	InOut	BOOL	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO from the previous bit logic operation is saved in edge memory bit "Tag_M". If a change in the RLO signal state from "0" to "1" is detected, the program jumps to jump label CAS1.

13.2.2.12 N_TRIG: Scan RLO for negative signal edge (STEP 7 Safety Advanced V11)

Description

You can use the "Scan RLO for negative signal edge" instruction to query a change in the signal state of the result of logic operation from "1" to "0". The instruction compares the current signal state of the result of logic operation with the signal state from the previous query, which is saved in the edge memory bit (<Operand>). If the instruction detects a change in the RLO from "1" to "0", there is a negative, falling edge.

If a falling edge is detected, the output of the instruction has signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the edge memory bit would be overwritten. This would influence edge evaluation and the result would no longer be unequivocal.

Note

If you want to use a formal parameter of an F-FB/F-FC for the edge memory bit <Operand> of the instruction then this must be declared as an in/out parameter.

The F-CPU can go to STOP if this caution is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
 - "Data corruption in the safety program prior to output to partner F-CPU"
 - "Safety program: internal CPU fault; internal error information: 404"
-

Note

You cannot use the "process image input", "process image output", or "bit memory" operand areas for the edge memory bit <Operand> of the instruction.

If operand area "local data" is used for the edge memory bit <Operand> of the instruction then the local data bit used must be initialized beforehand.

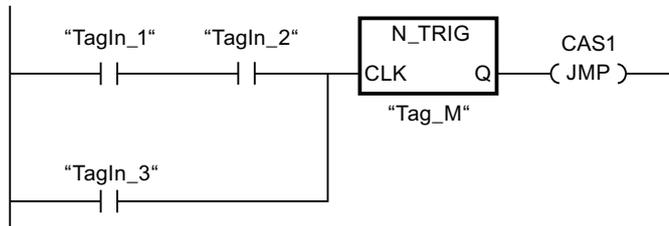
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
CLK	Input	BOOL	Current RLO
<Operand>	InOut	BOOL	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the previous bit logic operation is saved in edge bit memory "Tag_M". If a change in the RLO signal state from "1" to "0" is detected, the program jumps to jump label CAS1.

13.2.3 Safety functions**13.2.3.1 ESTOP1: Emergency Stop up to Stop Category 1 (STEP 7 Safety Advanced V11)****Description**

This instruction implements an emergency STOP shutdown with acknowledgment for Stop Categories 0 and 1.

Enable signal Q is reset to 0, as soon as input E_STOP takes a signal state of 0 (Stop category 0). Enable signal Q_DELAY is reset to 0 after the time delay set at input TIME_DEL (Stop Category 1).

Enable signal Q is reset to 1 not before input E_STOP takes a signal state of 1 and an acknowledgment occurs. The acknowledgment for the enable takes place according to the parameter assignment at input ACK_NEC:

- If ACK_NEC = 0, the acknowledgment is automatic.
- If ACK_NEC = 1, you must use a rising edge at input ACK for acknowledging the enable.

Output ACK_REQ is used to signal that a user acknowledgment is required at input ACK for the acknowledgment. The instruction sets output ACK_REQ to 1, as soon as input E_STOP = 1.

Following an acknowledgment, the instruction resets ACK_REQ to 0.

Every call of the "Emergency STOP up to Stop Category 1" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., ESTOP1_DB_1) or a multi-instance (e.g., ESTOP1_Instance_1) for the "Emergency STOP up to Stop Category 1" instruction.

Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

 **WARNING**

Tag ACK_NEC must not be assigned a value of 0 unless an automatic restart of the affected process is otherwise excluded. (S033)

 **WARNING**

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction")
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision. (S034)

Note: Only one emergency STOP signal (E_STOP) can be evaluated for the instruction. With suitable configuration (type of sensor interconnection: 2-channel equivalent), Discrepancy monitoring of the two NC contacts (when two channels are involved) in accordance with Categories 3 and 4 as defined in EN 954-1 is performed directly by the F-I/O with inputs. 2-channel equivalent) directly through the F-I/O with inputs. In order to keep the discrepancy time from influencing the response time, during the configuration of discrepancy behavior, you must parameterize "provide 0 value".

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
E_STOP	Input	BOOL	Emergency STOP
ACK_NEC	Input	BOOL	1=Acknowledgment necessary
ACK	Input	BOOL	1=Acknowledgment
TIME_DEL	Input	TIME	Time delay
Q	Output	BOOL	1=Enable
Q_DELAY	Output	BOOL	Enable is OFF delayed
ACK_REQ	Output	BOOL	1=Acknowledgment necessary
DIAG	Output	BYTE	Service information

Instruction versions

Two versions are available for this instruction:

- Version 1.0

When projects that were created with *S7 Distributed Safety V5.4 SP5* are migrated, Version 1.0 of the instruction is used automatically.

Version 1.0 requires that the F_TOF block with the number FB 186 is available in the project tree in the "Program blocks/System blocks/STEP 7 Safety" folder.

If you want to compile a migrated safety program with *STEP 7 Safety Advanced V11* for the first time, we recommend that you update the version of the ESTOP1 instruction to Version 1.1 beforehand. You will then avoid number conflicts.

- Version 1.1

When a new project is created with *STEP 7 Safety Advanced V11*, Version V1.1 is preset automatically. This version is functionally identical to Version V1.0, but does not require the F_TOF block to have a particular number.

For more information on the use of instruction versions, refer to the help on *STEP 7 Professional* under "Using instruction versions".

Startup characteristics

After an F-system startup, when ACK_NEC = 1, you must acknowledge the instruction using a rising edge at input ACK.

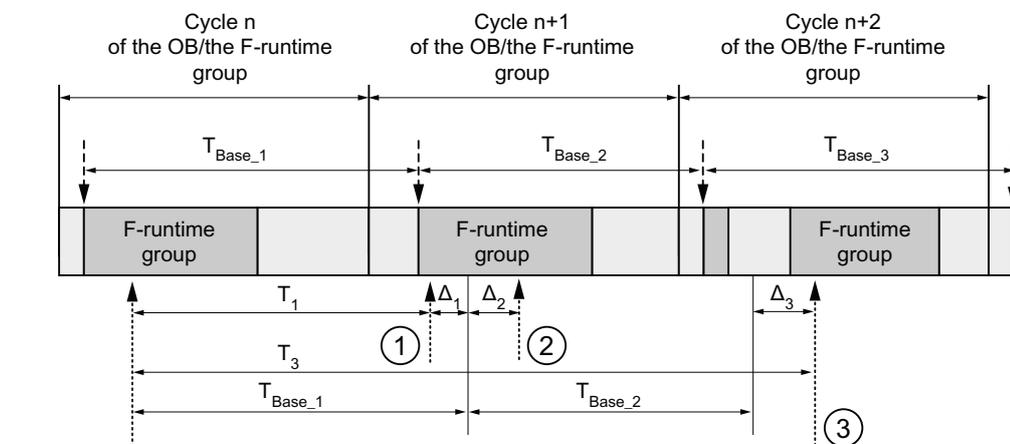
Output DIAG

The DIAG output provides non-fail-safe information on errors for service purposes. You can read out this information by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program. DIAG bits 1 to 5 are saved until acknowledgment at input ACK.

Structure of DIAG

Bit no.	Assignment	Possible error causes	Remedies
Bit 0	Incorrect TIM_DEL setting	Time delay setting < 0	Set time delay > 0
Bit 1	Reserved	—	—
Bit 2	Reserved	—	—
Bit 3	Reserved	—	—
Bit 4	Acknowledgment not possible because emergency STOP is still active	Emergency STOP switch is locked	Release interlocking of emergency STOP switch
		F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON of F-I/O of emergency STOP switch	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
		Emergency STOP switch is defective	Check emergency STOP switch
		Wiring fault	Check wiring of emergency STOP switch
Bit 5	If enable is missing: input ACK has a permanent signal state of 1	Acknowledgment button defective	Check acknowledgment button
		Wiring fault	Check wiring of acknowledgment button
Bit 6	Acknowledgement required (= state of ACK_REQ)	—	—
Bit 7	State of output Q	—	—

Timing imprecision resulting from the update time of the time base used in the instruction:



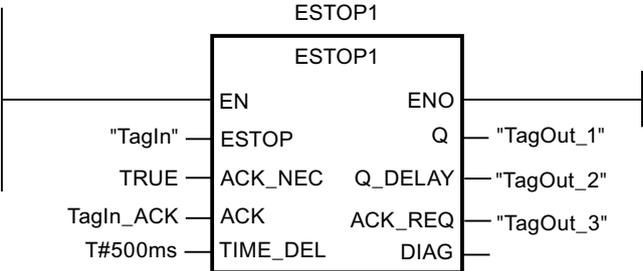
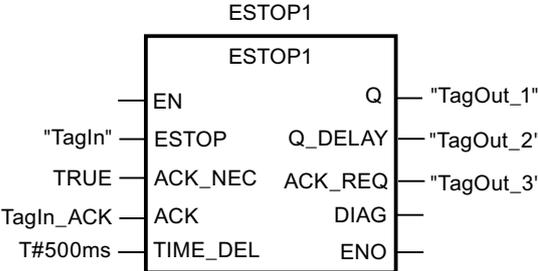
-----> = Time base update

.....> = Call time of an instruction with time processing

- ① For the first call in cycle n+1, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle n+1 are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle n+1. This does not involve another time update (by Δ_2).
- ③ For the call in cycle n+2, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle n+2. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle n+1.

Example

The following example shows how the instruction works:



13.2.3.2 TWO_HAND: Two-hand monitoring (STEP 7 Safety Advanced V11)

Description

This instruction implements two-hand monitoring.

If pushbuttons IN1 and IN2 are activated within the permitted discrepancy time DISCTIME \leq 500 ms (IN1/IN2 = 1) (synchronous activation), output signal Q is set to 1. If the time difference between activation of pushbutton IN1 and pushbutton IN2 is greater than DISCTIME, then the pushbuttons must be released and reactivated.

Q is reset to 0 as soon as one of the pushbuttons is released (IN1/IN2 = 0). Enable signal Q can be reset to 1 only if the other pushbutton has been released, and if both pushbuttons are then reactivated within the discrepancy time. Enable signal Q can never be set to 1 if the discrepancy time is set to values less than 0 or greater than 500 ms.

Every call of the "Two-hand monitoring" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., TWO_HAND_DB_1) or a multi-instance (e.g., TWO_HAND_Instance_1) for the "Two-hand monitoring" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

The instruction supports the requirements in accordance with EN 574:1996 + A1:2008.

Note: Only one signal per pushbutton can be evaluated in the instruction. Discrepancy monitoring of the NC and NO contacts of pushbuttons IN1 and IN2 is performed directly during suitable configuration (type of sensor interconnection: 2-channel equivalent) directly through the F-I/O with inputs. The normally open contact must be wired in such a way that it supplies the useful signal (see manual for the F-I/O you are using). In order to keep the discrepancy time from influencing the response time, you must assign "Provide value 0" for the behavior of discrepancy during configuration. If a discrepancy is detected, a fail-safe value of 0 is entered in the process image input (PII) for the pushbutton and QBAD or QBAD_I_xx = 1 is set in the relevant F-I/O DB. (See also F-I/O access (Page 79))

WARNING

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction")
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

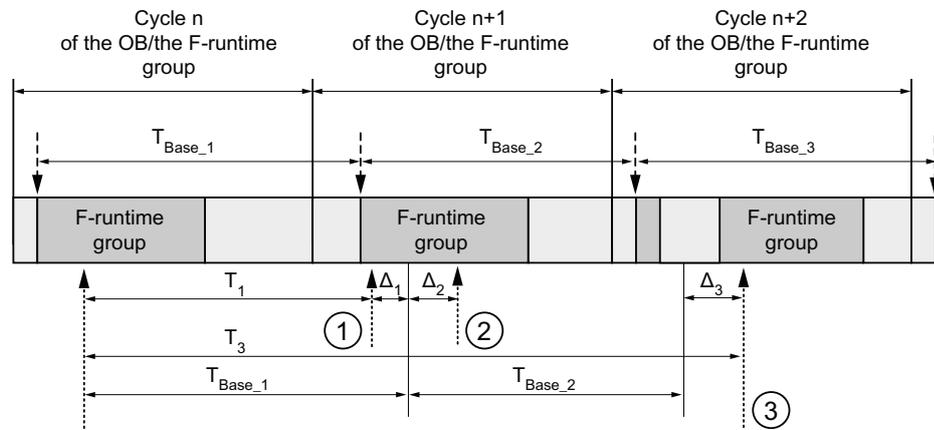
You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision. (S034)

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN1	Input	BOOL	Pushbutton 1
IN2	Input	BOOL	Pushbutton 2
DISCTIME	Input	TIME	Discrepancy time (0 to 500 ms)
Q	Output	BOOL	1=Enable

Timing imprecision resulting from the update time of the time base used in the instruction:

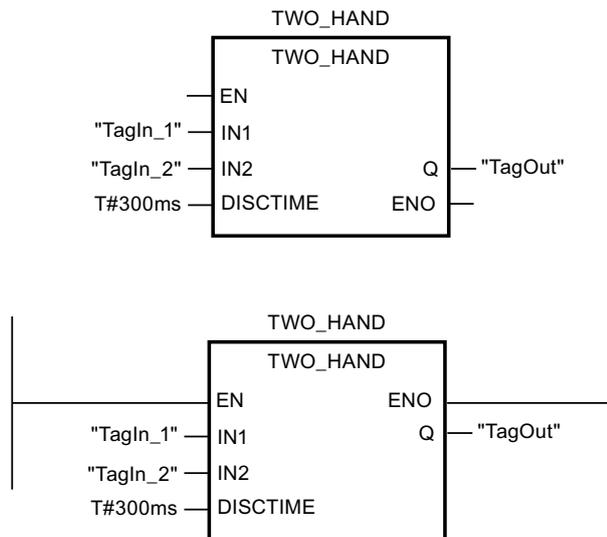


-----> = Time base update
> = Call time of an instruction with time processing

- ① For the first call in cycle n+1, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle n+1 are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle n+1. This does not involve another time update (by Δ_2).
- ③ For the call in cycle n+2, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle n+2. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle n+1.

Example

The following example shows how the instruction works:



13.2.3.3 TWO_H_EN: Two-hand monitoring with enable (STEP 7 Safety Advanced V11)

Description

This instruction implements two-hand monitoring with enable.

If pushbuttons IN1 and IN2 are activated within the permitted discrepancy time $DISCTIME \leq 500$ ms ($IN1/IN2 = 1$) (synchronous activation), output signal Q is set to 1 when existing $ENABLE = 1$. If the time difference between activation of pushbutton IN1 and pushbutton IN2 is greater than $DISCTIME$, then the pushbuttons must be released and reactivated.

Q is reset to 0 as soon as one of the pushbuttons is released ($IN1/IN2 = 0$) or $ENABLE = 0$. Enable signal Q can be reset to 1 only if the other pushbutton has been released, and if both pushbuttons are then reactivated within the discrepancy time when existing $ENABLE = 1$.

Every call of the "Two-hand monitoring with enable" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., `TWO_H_EN_DB_1`) or a multi-instance (e.g., `TWO_H_EN_Instance_1`) for the "Two-hand monitoring with enable" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

The instruction supports the requirements in accordance with EN 574:1996 + A1:2008.

Note: Only one signal per pushbutton can be evaluated in the instruction. Discrepancy monitoring of the NC and NO contacts of pushbuttons IN1 and IN2 is performed directly during suitable configuration (type of sensor interconnection: 2-channel equivalent) directly through the F-I/O with inputs. The normally open contact must be wired in such a way that it supplies the useful signal (see manual for the F-I/O you are using). In order to keep the discrepancy time from influencing the response time, during the configuration of discrepancy behavior, you must parameterize "provide 0 value".

If a discrepancy is detected, a fail-safe value of 0 is entered in the process image input (PII) for the pushbutton and QBAD or QBAD_I_xx = 1 is set in the relevant F-I/O DB.

WARNING

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction")
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision. (S034)

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN1	Input	BOOL	Pushbutton 1
IN2	Input	BOOL	Pushbutton 2
ENABLE	Input	BOOL	Enable input
DISCTIME	Input	TIME	Discrepancy time (0 to 500 ms)
Q	Output	BOOL	1=Enable
DIAG	Output	BYTE	Service information

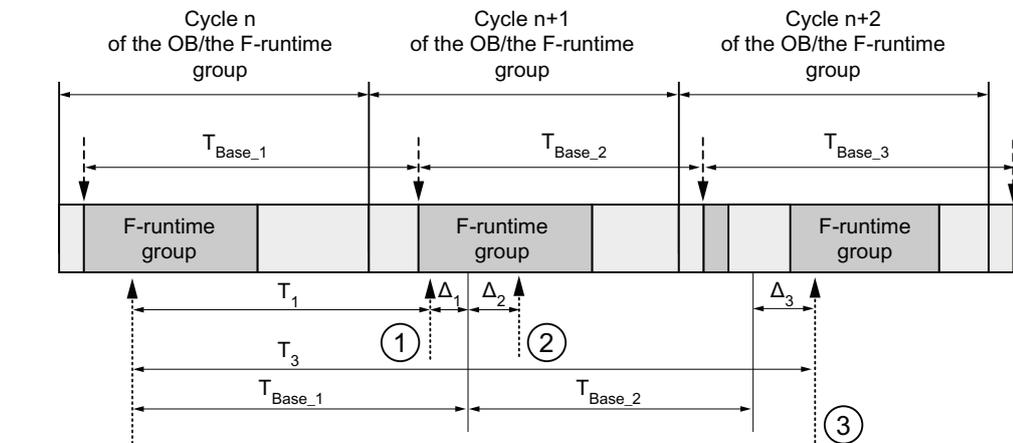
Output DIAG

The DIAG output provides non-fail-safe information on errors for service purposes. You can read out this information by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program. DIAG bits 0 to 5 are saved until the cause of the error has been eliminated.

Structure of DIAG

Bit no.	Assignment	Possible error causes	Remedies
Bit 0	Incorrect discrepancy time DISCTIME setting	Discrepancy time setting is <0 or > 500 ms	Set discrepancy time in range of 0 to 500 ms
Bit 1	Discrepancy time elapsed	Discrepancy time setting is too low	If necessary, set a higher discrepancy time
		Pushbuttons were not activated within the discrepancy time	Release pushbuttons and activate them within the discrepancy time
		Wiring fault	Check wiring of pushbuttons
		Pushbuttons defective	Check pushbuttons
		Pushbuttons are wired to different F-I/O, and F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON on an F-I/O	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
Bit 2	Reserved	—	—
Bit 3	Reserved	—	—
Bit 4	Incorrect activation sequence	One pushbutton was not released	Release pushbuttons and activate them within the discrepancy time
		Pushbuttons defective	Check pushbuttons
Bit 5	ENABLE does not exist	ENABLE = 0	Set ENABLE = 1, release pushbutton and activate it within the discrepancy time
Bit 6	Reserved	—	—
Bit 7	State of output Q	—	—

Timing imprecision resulting from the update time of the time base used in the instruction:



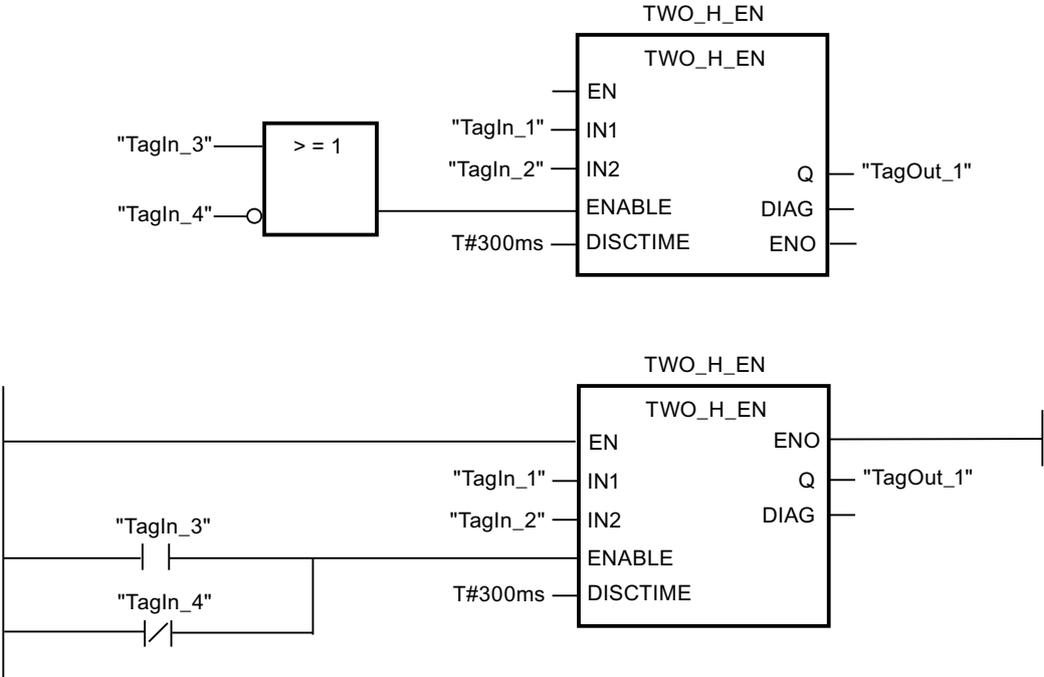
-----▶ = Time base update

.....▶ = Call time of an instruction with time processing

- ① For the first call in cycle n+1, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle n+1 are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle n+1. This does not involve another time update (by Δ_2).
- ③ For the call in cycle n+2, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle n+2. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle n+1.

Example

The following example shows how the instruction works:



13.2.3.4 MUTING: Muting (STEP 7 Safety Advanced V11)

Description

This instruction performs parallel muting with two or four muting sensors.

Muting is a defined suppression of the protective function of light curtains. Light curtain muting can be used to introduce goods or objects into the danger area monitored by the light curtain without causing the machine to stop.

To utilize the muting function, at least two independently wired muting sensors must be present. The use of two or four muting sensors and correct integration into the production sequence must ensure that no persons enter the danger area while the light curtain is muted.

Every call of the "Muting" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., MUTING_DB_1) or a multi-instance (e.g., MUTING_Instance_1) for the "Muting" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

WARNING

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction")
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

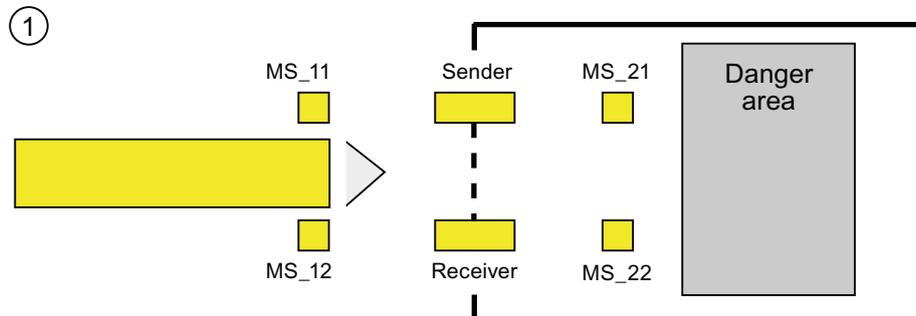
You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision. (S034)

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
MS_11	Input	BOOL	Muting sensor 1 of sensor pair 1
MS_12	Input	BOOL	Muting sensor 2 of sensor pair 1
MS_21	Input	BOOL	Muting sensor 1 of sensor pair 2
MS_22	Input	BOOL	Muting sensor 2 of sensor pair 2
STOP	Input	BOOL	1=Conveyor system stopped
FREE	Input	BOOL	1=Light curtain uninterrupted
QBAD_MUT	Input	BOOL	QBAD or QBAD_O_xx signal of F-I/O/channel of muting lamp (F-I/O DB)
DISCTIM1	Input	TIME	Discrepancy time of sensor pair 1 (0 to 3 s)
DISCTIM2	Input	TIME	Discrepancy time of sensor pair 2 (0 to 3 s)
TIME_MAX	Input	TIME	Maximum muting time (0 to 10 min)
ACK	Input	BOOL	Acknowledgment of restart inhibit
Q	Output	BOOL	1= Enable, not off
MUTING	Output	BOOL	Display of muting is active
ACK_REQ	Output	BOOL	Acknowledgment necessary
FAULT	Output	BOOL	Group error
DIAG	Output	BYTE	Service information

Schematic sequence of error-free muting procedure with 4 muting sensors (MS_11, MS_12, MS_21, MS_22)

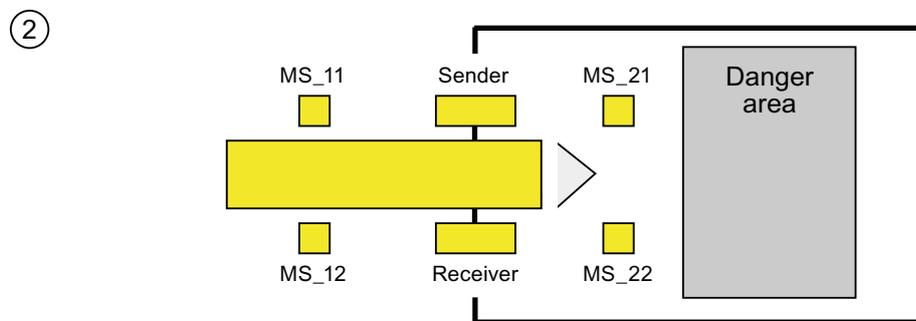


- If both muting sensors MS_11 and MS_12 are activated by the product within DISCTIM1 (apply signal state = 1), the instruction starts the MUTING function. Enable signal Q remains 1, even when input FREE = 0 (light curtain interrupted by product). The MUTING output for setting the muting lamp switches to 1.

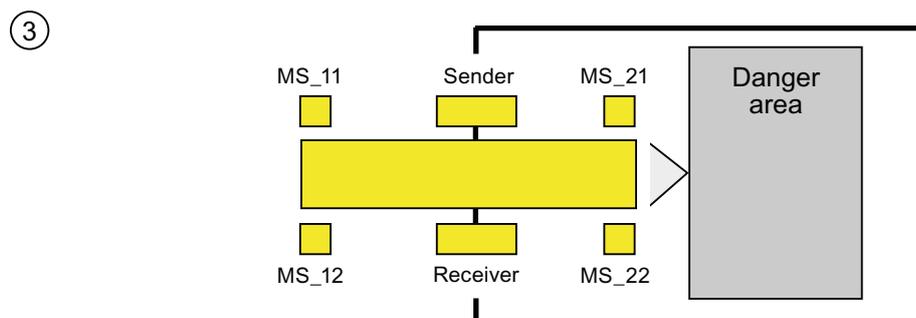
Note

The muting lamp can be monitored using the QBAD_MUT input. To do this, you must wire the muting lamp to an output with wire break monitoring of an F-I/O and supply the QBAD_MUT input with the QBAD or QBAD_O_xx signal of the associated F-I/O or channel. If QBAD_MUT = 1, muting is terminated by the instruction. If monitoring of the muting lamp is not necessary, you do not have to supply input QBAD_MUT.

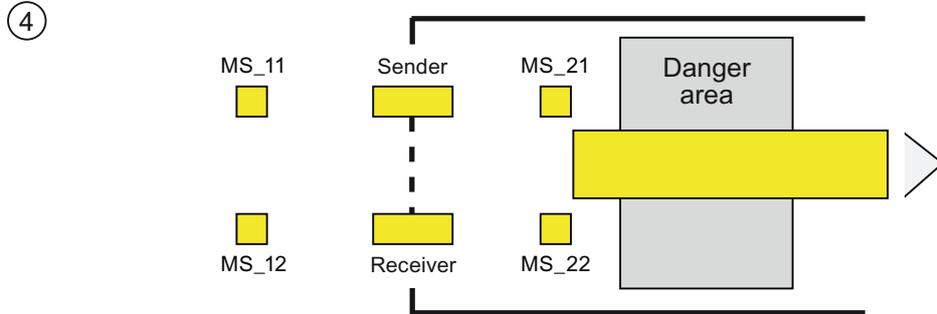
F-I/O that can promptly detect a wire break after activation of the muting operation must be used (*see manual for specific F-I/O*).



- As long as both muting sensors MS_11 and MS_12 continue to be activated, the MUTING function of the instruction causes Q to remain 1 and MUTING to remain 1 (so that the product can pass through the light curtain without causing the machine to stop).



- The two muting sensors MS_21 and MS_22 must be activated (within DISCTIM2) before muting sensors MS_11 and MS_12 are switched to inactive (apply signal state 0). In this way, the instruction retains the MUTING function. (Q = 1, MUTING = 1).

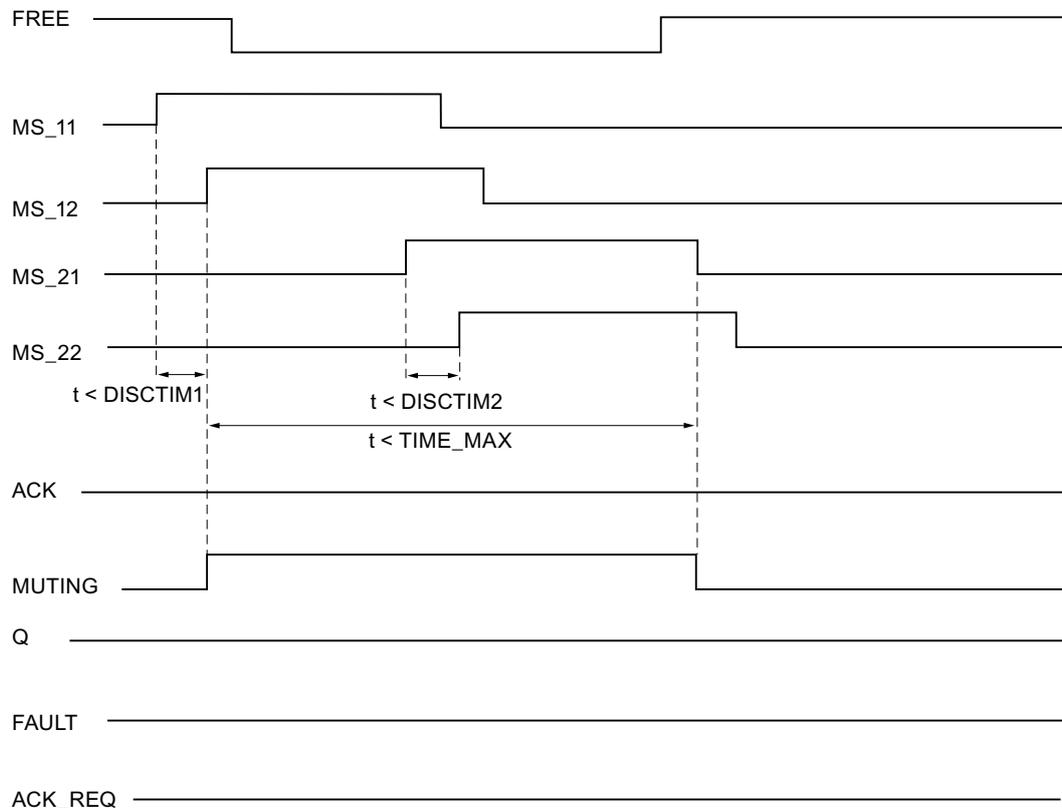


- Only if one of the two muting sensors MS_21 and MS_22 is switched to inactive (product enables sensors) is the MUTING function terminated (Q = 1, MUTING = 0). The maximum activation time for the MUTING function is the time set at input TIME_MAX.

Note

The MUTING function is also started if the product passes the light curtain in the reverse direction and the muting sensors are thus activated by the product in reverse order.

Timing diagrams for error-free muting procedure with 4 muting sensors

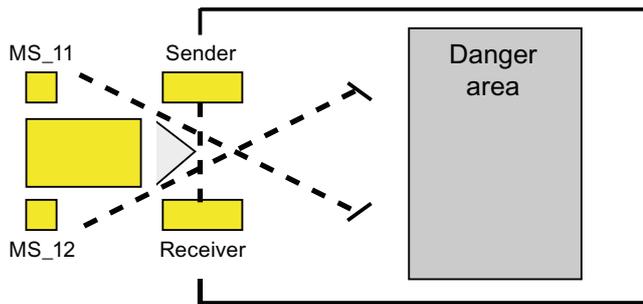


Schematic sequence of muting procedure with reflection light barriers

If reflection light barriers are used as muting sensors, they are generally arranged diagonally.

In general, this arrangement of reflection light barriers as muting sensors requires only two light barriers, and only MS_11 and MS_12 are interconnected.

The sequence is similar to that of the muting procedure with 4 muting sensors. Step 3 is omitted. In step 4, replace MS_21 and MS_22 with MS_11 and MS_12, respectively.



Restart inhibit upon interruption of light curtain (if MUTING is not active), when errors occur, and during F-system startup

Enable signal Q cannot be set to 1 or becomes 0, if:

- Light curtain is interrupted (e.g., by a person or material transport) while the MUTING function is not active
- The muting lamp monitoring function responds at input QBAD_MUT
- Sensor pair 1 (MS_11 and MS_12) or sensor pair 2 (MS_21 and MS_22) is not activated or deactivated during discrepancy time DISCTIM1 or DISCTIM2, respectively
- The MUTING function is active longer than the maximum muting time TIME_MAX
- Discrepancy times DISCTIM1 and DISCTIM2 have been set to values < 0 or > 3 s
- Maximum muting time TIME_MAX has been set to a value < 0 or > 10 min

In the identified cases, output FAULT (group error) is set to 1 (restart inhibit). If the MUTING function is started, it will be terminated and the Muting output becomes 0.

WARNING

When a valid combination of muting sensors is immediately detected at startup of the F-system (for example, because the muting sensors are interconnected to inputs of a standard I/O that immediately provide process values during the F-system startup), the MUTING function is immediately started and the MUTING output and enable signal Q are set to 1. The FAULT output (group error) is not set to 1 (no restart inhibit!). (S035)

Acknowledgment of restart inhibit

Enable signal Q becomes 1 again, if:

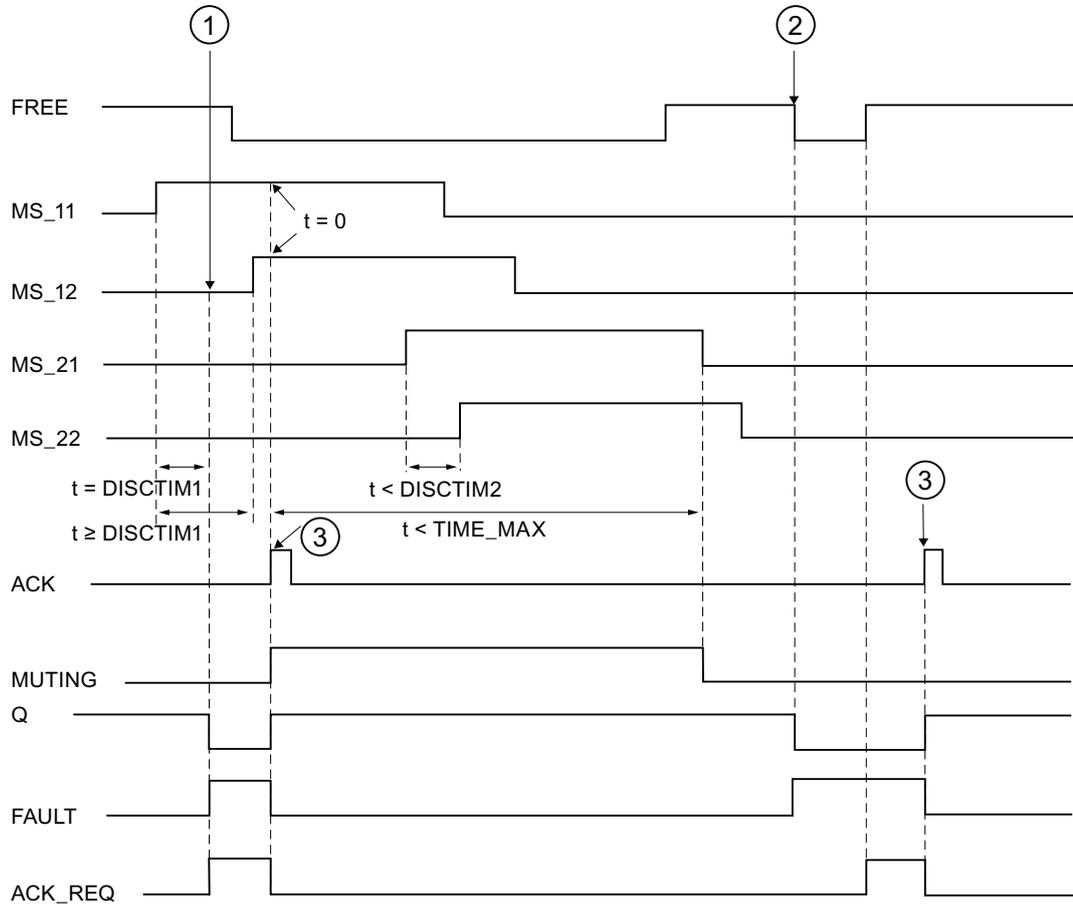
- The light curtain is no longer interrupted
 - Errors, if present, are eliminated (see output DIAG)
- and
- A user acknowledgement with positive edge is occurs at input ACK (see also Implementation of user acknowledgment (Page 99)).

The FAULT output is set to 0. Output ACK_REQ = 1 signals that user acknowledgment at input ACK is required to eliminate the restart inhibit. The instruction sets ACK-REQ = 1 as soon as the light curtain is no longer interrupted or errors have been eliminated. Once acknowledgment has occurred, the instruction resets ACK_REQ to 0.

Note

Following discrepancy errors and once the maximum muting time has been exceeded, ACK_REQ is immediately set to 1. As soon as a user acknowledgment has taken place at input ACK, discrepancy times DISCTIM1 and DISCTIM2 and maximum muting time TIME_MAX are reset.

Timing diagrams for discrepancy errors at sensor pair 1 or interruption of the light curtain (if MUTING is not active)



- ① Sensor pair 1 (MS_11 and MS_12) is not activated within discrepancy time DISCTIM1.
- ② The light curtain is interrupted even though the MUTING function is not active.
- ③ Acknowledgment

Behavior with stopped conveyor equipment

If, while the conveyor equipment has stopped, the monitoring for one of the following reasons is to be deactivated:

- To comply with discrepancy time DISCTIM1 or DISCTIM2
- To comply with maximum muting time TIME_MAX

you must supply input STOP with a "1" signal for as long as the conveyor equipment is stopped. As soon as the conveyor equipment is running again (STOP = 0), discrepancy times DISCTIM1 and DISCTIM2 and maximum muting time TIME_MAX are reset.

WARNING

When STOP = 1, the discrepancy monitoring is shut down. During this time, if inputs MSx1/MSx2 of a sensor pair both take a signal state of 1 due to an undetected error, e.g., because both muting sensors fail to 1, the error is not detected and the MUTING function can be started unintentionally. (S036)

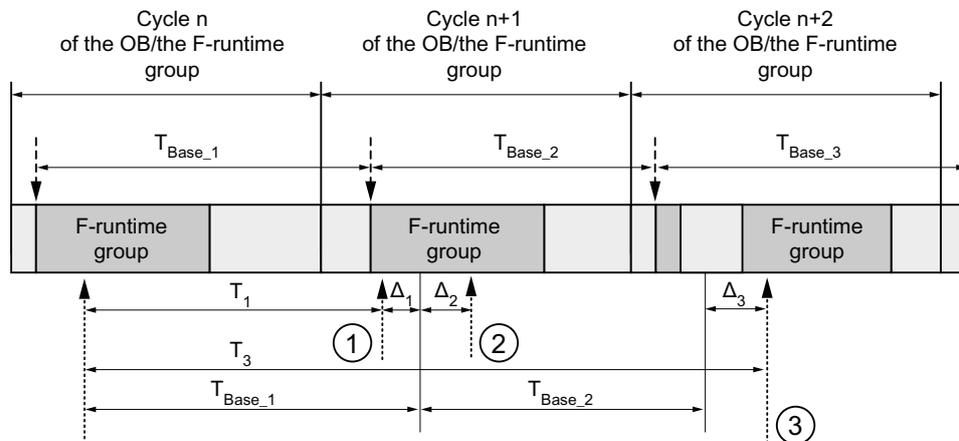
Output DIAG

The DIAG output provides non-fail-safe information on errors for service purposes. You can read out this information by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program. DIAG bits are saved until acknowledgment at input ACK.

Structure of DIAG

Bit no.	Assignment	Possible error causes	Remedies
Bit 0	Discrepancy error or incorrect discrepancy time DISCTIM 1 setting for sensor pair 1	Malfunction in production sequence	Malfunction in production sequence eliminated
		Sensor defective	Check sensors
		Wiring fault	Check wiring of sensors
		Sensors are wired to different F-I/O, and F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON on an F-I/O	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
		Discrepancy time setting is too low	If necessary, set a higher discrepancy time
		Discrepancy time setting is < 0 s or > 3 s	Set discrepancy time in range between 0 s and 3 s
Bit 1	Discrepancy error or incorrect discrepancy time DISCTIM 2 setting for sensor pair 2	Same as Bit 0	Same as Bit 0
Bit 2	Maximum muting time exceeded or incorrect muting time TIME_MAX setting	Malfunction in production sequence	Malfunction in production sequence eliminated
		Maximum muting time setting is too low	If necessary, set a higher maximum muting time
		Muting time setting is < 0 s or > 10 min	Set muting time in range from 0 s to 10 min
Bit 3	Light curtain interrupted and muting not active	Light curtain is defective	Check light curtain
		Wiring fault	Check wiring of light curtain (FREE input)
		F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON of F-I/O of light curtain (FREE input)	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
		See other DIAG bits	
Bit 4	Muting lamp is defective or cannot be set	Muting lamp is defective	Replace muting lamp
		Wiring fault	Check wiring of muting lamp
		F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON of F-I/O of muting lamp	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
Bit 5	Reserved	—	—
Bit 6	Reserved	—	—
Bit 7	Reserved	—	—

Timing imprecision resulting from the update time of the time base used in the instruction:



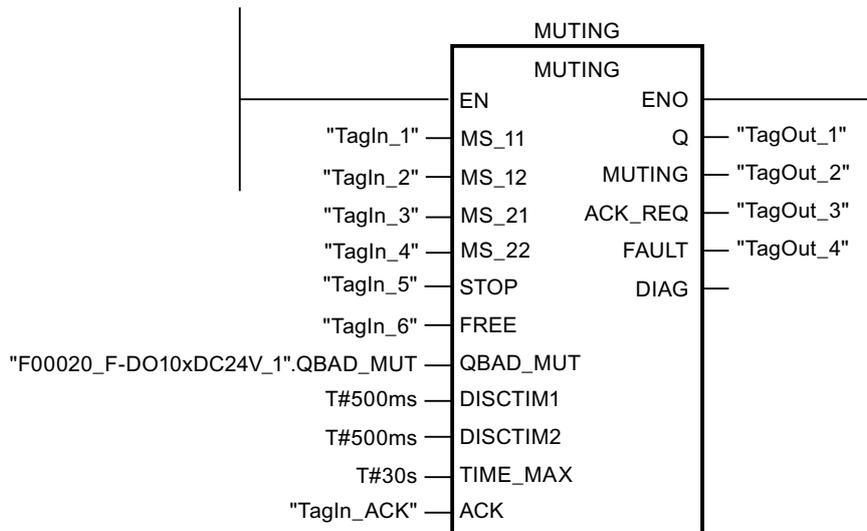
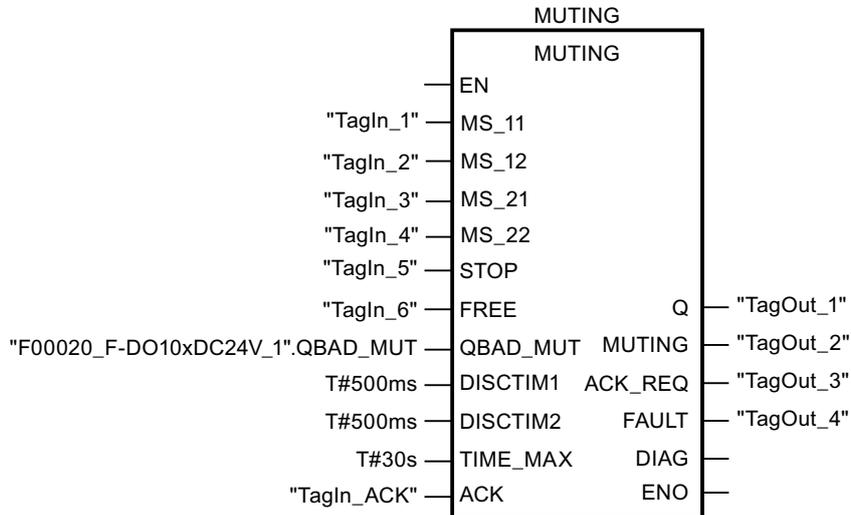
-----> = Time base update

.....> = Call time of an instruction with time processing

- ① For the first call in cycle n+1, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle n+1 are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle n+1. This does not involve another time update (by Δ_2).
- ③ For the call in cycle n+2, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle n+2. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle n+1.

Example

The following example shows how the instruction works:



13.2.3.5 MUT_P: Parallel muting (STEP 7 Safety Advanced V11)

Description

This instruction performs parallel muting with two or four muting sensors.

Muting is a defined suppression of the protective function of light curtains. Light curtain muting can be used to introduce goods or objects into the danger area monitored by the light curtain without causing the machine to stop.

To utilize the muting function, at least two independently wired muting sensors must be present. The use of two or four muting sensors and correct integration into the production sequence must ensure that no persons enter the danger area while the light curtain is muted.

Every call of the "Parallel muting" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., MUT_P_DB_1) or a multi-instance (e.g., MUT_P_Instance_1) for the "Parallel muting" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

WARNING

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction")
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

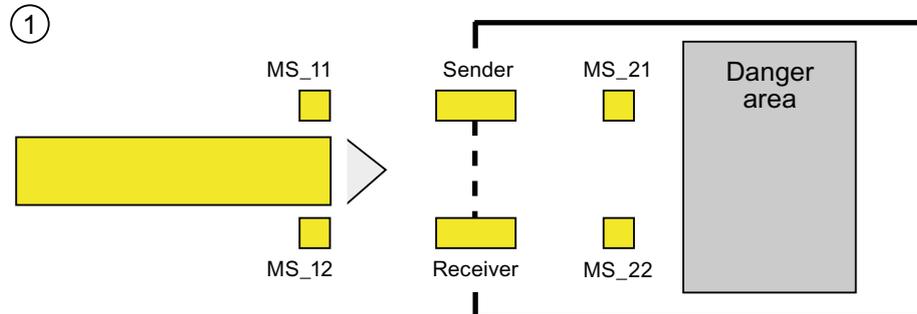
You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision. (S034)

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
MS_11	Input	BOOL	Muting sensor 11
MS_12	Input	BOOL	Muting sensor 12
MS_21	Input	BOOL	Muting sensor 21
MS_22	Input	BOOL	Muting sensor 22
STOP	Input	BOOL	1=Conveyor system stopped
FREE	Input	BOOL	1=Light curtain uninterrupted
ENABLE	Input	BOOL	1=Enable MUTING
QBAD_MUT	Input	BOOL	QBAD or QBAD_O_xx signal of F-I/O/channel of muting lamp (F-I/O DB)
ACK	Input	BOOL	Acknowledgment of restart inhibit
DISCTIM1	Input	TIME	Discrepancy time of sensor pair 1 (0 to 3 s)
DISCTIM2	Input	TIME	Discrepancy time of sensor pair 2 (0 to 3 s)
TIME_MAX	Input	TIME	Maximum muting time (0 to 10 min)
Q	Output	BOOL	1= Enable, not off
MUTING	Output	BOOL	Display of muting is active
ACK_REQ	Output	BOOL	Acknowledgment necessary
FAULT	Output	BOOL	Group error
DIAG	Output	WORD	Service information

Schematic sequence of error-free muting procedure with 4 muting sensors (MS_11, MS_12, MS_21, MS_22)

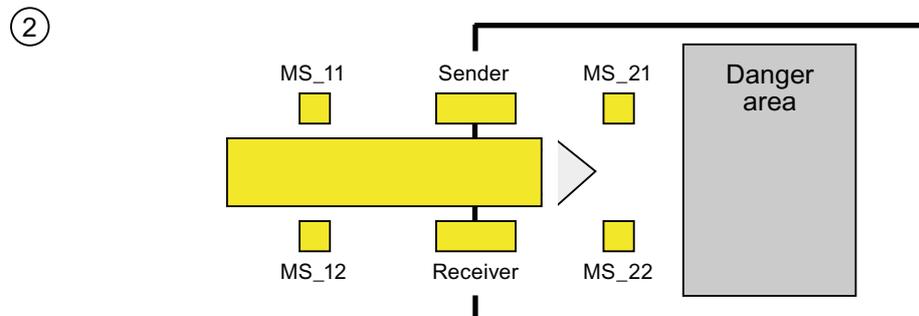


- If muting sensors MS_11 and MS_12 are both activated by the product within DISCTIM1 (apply signal state = 1) and MUTING is enabled by setting the ENABLE input to 1, the instruction starts the MUTING function. Enable signal Q remains 1, even when input FREE = 0 (light curtain interrupted by product). The MUTING output for setting the muting lamp switches to 1.

Note

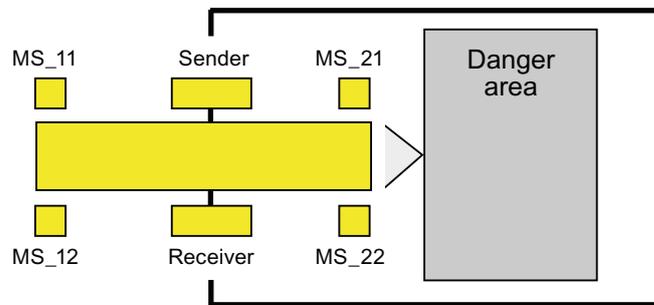
The muting lamp can be monitored using the QBAD_MUT input. To do this, you must wire the muting lamp to an output with wire break monitoring of an F-I/O and supply the QBAD_MUT input with the QBAD or QBAD_O_xx signal of the associated F-I/O or channel. If QBAD_MUT = 1, muting is terminated by the instruction. If monitoring of the muting lamp is not necessary, you do not have to supply input QBAD_MUT.

F-I/O that can promptly detect a wire break after activation of the muting operation must be used (*see manual for specific F-I/O*).



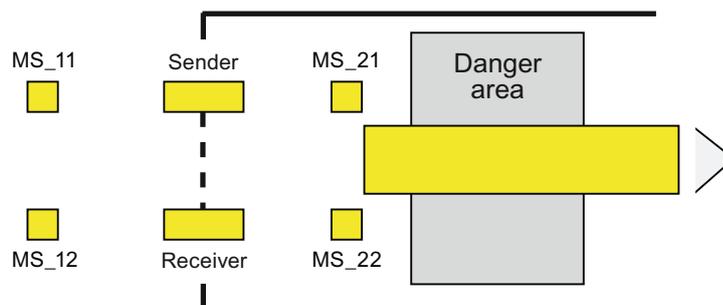
- As long as both muting sensors MS_11 and MS_12 continue to be activated, the MUTING function of the instruction causes Q to remain 1 and MUTING to remain 1 (so that the product can pass through the light curtain without causing the machine to stop). Each of the two muting sensors MS_11 and MS_12 may be switched to inactive ($t < DISCTIM1$) for a short time (apply signal state 0).

③



- Muting sensors MS_21 and MS_22 must both be activated (within DISCTIM2) before muting sensors MS_11 and MS_12 are switched to inactive (apply signal state 0). In this way, the instruction retains the MUTING function. ($Q = 1$, $MUTING = 1$).

④

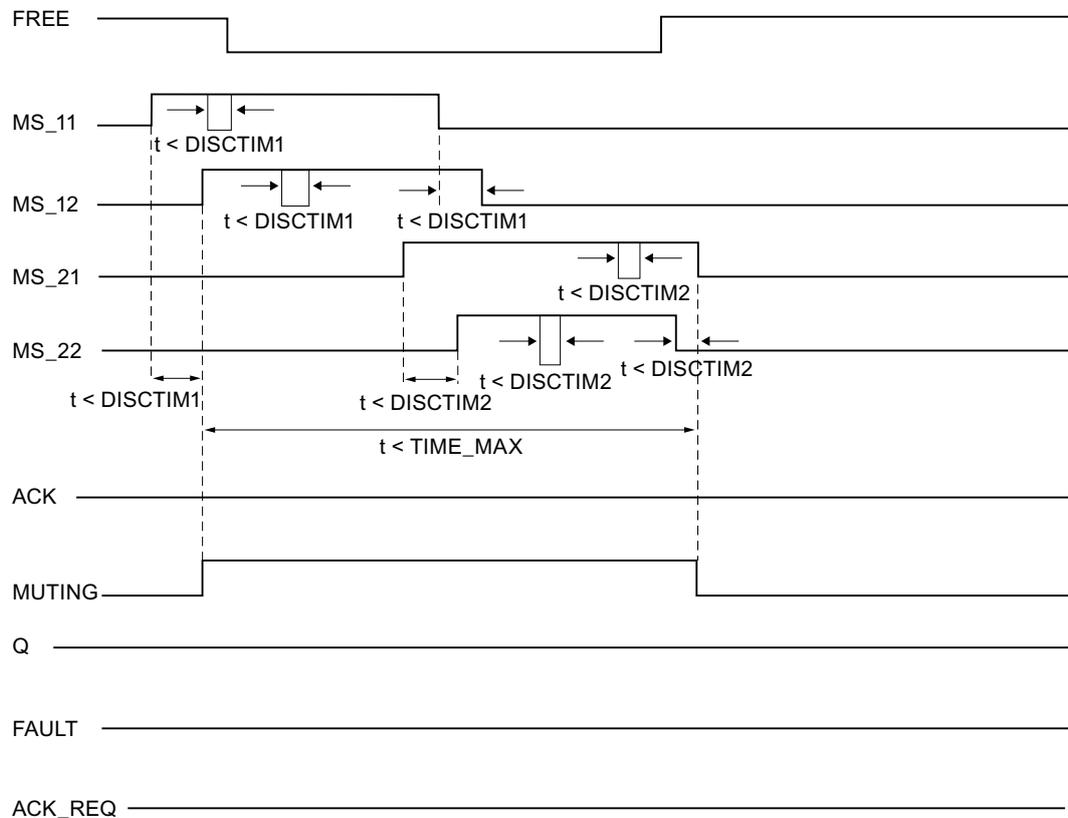


Only if muting sensors MS_21 and MS_22 are both switched to inactive (product enables sensors) is the MUTING function terminated ($Q = 1$, $MUTING = 0$). The maximum activation time for the MUTING function is the time set at input TIME_MAX.

Note

The MUTING function is also started if the product passes the light curtain in the reverse direction and the muting sensors are thus activated by the product in reverse order.

Timing diagrams for error-free muting procedure with 4 muting sensors

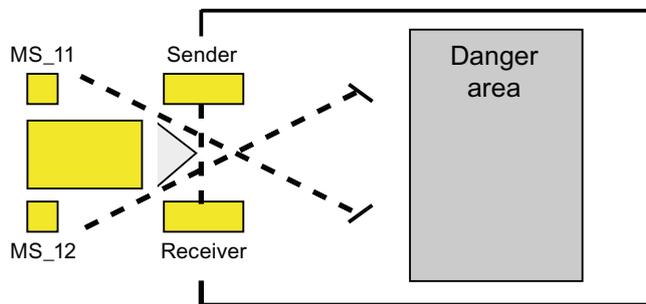


Schematic sequence of muting procedure with reflection light barriers

If reflection light barriers are used as muting sensors, they are generally arranged diagonally.

In general, this arrangement of reflection light barriers as muting sensors requires only two light barriers, and only MS_11 and MS_12 are interconnected.

The sequence is similar to that of the muting procedure with 4 muting sensors. Step 3 is omitted. In step 4, replace MS_21 and MS_22 with MS_11 and MS_12, respectively.



Restart inhibit upon interruption of light curtain (MUTING is not active), as well as when errors occur and during F-system startup

Enable signal Q cannot be set to 1 or becomes 0, if:

- Light curtain is interrupted (e.g., by a person or material transport) while the MUTING function is not active
- Light curtain is (being) interrupted and the muting lamp monitoring at input QBAD_MUT is set to 1
- Light curtain is (being) interrupted and the MUTING function is not enabled by setting input ENABLE to 1
- Sensor pair 1 (MS_11 and MS_12) or sensor pair 2 (MS_21 and MS_22) is not activated or deactivated during discrepancy time DISCTIM1 or DISCTIM2, respectively
- The MUTING function is active longer than the maximum muting time TIME_MAX
- Discrepancy times DISCTIM1 and DISCTIM2 have been set to values < 0 or > 3 s
- Maximum muting time TIME_MAX has been set to a value < 0 or > 10 min
- The F-system starts up (regardless of whether or not the light curtain is interrupted, because the F-I/O is passivated after F-system startup and, thus, the FREE input is initially supplied with 0)

In the identified cases, output FAULT (group error) is set to 1 (restart inhibit). If the MUTING function is started, it will be terminated and the Muting output becomes 0.

User acknowledgment of restart inhibit (no muting sensor is activated or ENABLE = 0)

Enable signal Q becomes 1 again, if:

- The light curtain is no longer interrupted
 - Errors, if present, are eliminated (see output DIAG)
- and
- A user acknowledgement with positive edge occurs at input ACK (see also Implementation of user acknowledgment (Page 99)).

The FAULT output is set to 0. Output ACK_REQ = 1 signals that user acknowledgment at input ACK is required to eliminate the restart inhibit. The instruction sets ACK_REQ = 1 as soon as the light curtain is no longer interrupted or the errors have been eliminated. Once acknowledgment has occurred, the instruction resets ACK_REQ to 0.

User Acknowledgment of restart inhibit (at least one muting sensor is activated and ENABLE = 1)

Enable signal Q becomes 1 again, if:

- Errors, if present, are eliminated (see output DIAG)
- FREE occurs until a valid combination of muting sensors is detected

The FAULT output is set to 0. The MUTING function is restarted, if necessary, and the MUTING output becomes 1 if a valid combination of muting sensors is detected. When ENABLE = 1, output ACK_REQ = 1 signals that FREE is necessary for error elimination and for removal of the restart inhibit. Once FREE has occurred, the instruction resets ACK_REQ to 0.

Note

Once the maximum muting time is exceeded, TIME_MAX is reset as soon as the MUTING function is restarted.

FREE function

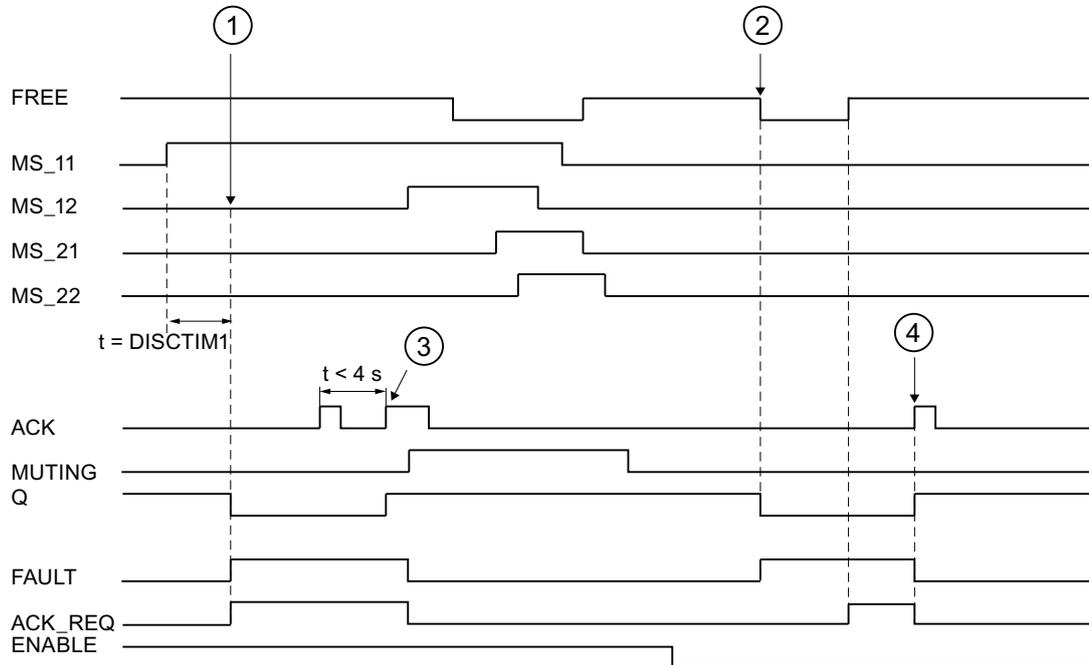
If an error cannot be corrected immediately, the FREE function can be used to free the muting range. Enable signal Q and output MUTING = 1 temporarily. The FREE function can be used if:

- ENABLE = 1
- At least one muting sensor is activated
- A user acknowledgment with rising edge at input ACK occurs twice within 4 s, and the second user acknowledgment at input ACK remains at a signal state of 1 (acknowledgment button remains activated)

 WARNING

When using the FREE function, the action must be observed. A dangerous situation must be able to be interrupted at any time by releasing the acknowledgment button. The acknowledgment button must be mounted in such a way the entire danger area can be observed. (S037)

Timing diagrams for discrepancy errors at sensor pair 1 or interruption of the light curtain (MUTING is not active)



- ① Sensor pair 1 (MS_11 and MS_22) is not activated within discrepancy time DISCTIM1.
- ② The light curtain is interrupted even though there is no enable (ENABLE=0)
- ③ FREE function
- ④ Acknowledgment

Behavior with stopped conveyor equipment

If, while the conveyor equipment has stopped, the monitoring for one of the following reasons is to be deactivated:

- To comply with discrepancy time DISCTIM1 or DISCTIM2
- To comply with maximum muting time TIME_MAX

you must supply input STOP with a "1" signal for as long as the conveyor equipment is stopped. As soon as the conveyor equipment is running again (STOP = 0), discrepancy times DISCTIM1 and DISCTIM2 and maximum muting time TIME_MAX are reset.

WARNING
<p>When STOP = 1 or ENABLE = 0, discrepancy monitoring is shut down. During this time, if inputs MSx1/MSx2 of a sensor pair both take a signal state of 1 due to an undetected error, e.g., because both muting sensors fail to 1, the fault is not detected and the MUTING function can be started unintentionally (when ENABLE = 1). (S038)</p>

Output DIAG

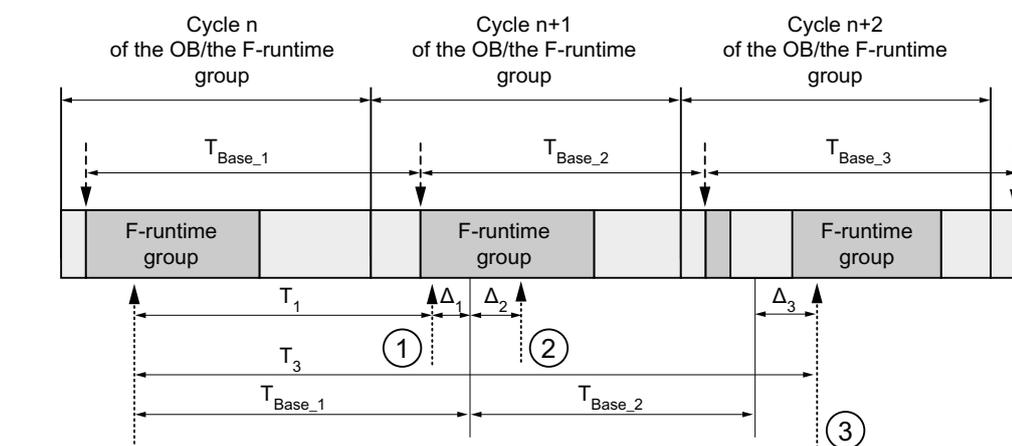
The DIAG output provides non-fail-safe information on errors for service purposes. You can read out this information by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program. DIAG bits 0 to 6 are saved until acknowledgment at input ACK.

Structure of DIAG

Bit no.	Assignment	Possible error causes	Remedies
Bit 0	Discrepancy error or incorrect discrepancy time DISCTIM 1 setting for sensor pair 1	Malfunction in production sequence	Malfunction in production sequence eliminated
		Sensor defective	Check sensors
		Wiring fault	Check wiring of sensors
		Sensors are wired to different F-I/O, and F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON on an F-I/O	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
		Discrepancy time setting is too low	If necessary, set a higher discrepancy time
		Discrepancy time setting is < 0 s or > 3 s	Set discrepancy time in range between 0 s and 3 s
Bit 1	Discrepancy error or incorrect discrepancy time DISCTIM 2 setting for sensor pair 2	Same as Bit 0	Same as Bit 0
Bit 2	Maximum muting time exceeded or incorrect muting time TIME_MAX setting	Malfunction in production sequence	Malfunction in production sequence eliminated
		Maximum muting time setting is too low	If necessary, set a higher maximum muting time
		Muting time setting is < 0 s or > 10 min	Set muting time in range from 0 s to 10 min
Bit 3	Light curtain interrupted and muting not active	ENABLE = 0	Set ENABLE = 1
		Light curtain is defective	Check light curtain
		Wiring fault	Check wiring of light curtain (FREE input)
		I/O fault, channel fault, or communication error, or passivation by means of PASS_ON of F-I/O of light curtain (FREE input)	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
		Startup of F-system	For FREE, see DIAG Bit 5
		See other DIAG bits	
Bit 4	Muting lamp is defective or cannot be set	Muting lamp is defective	Replace muting lamp
		Wiring fault	Check wiring of muting lamp
		F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON of F-I/O of muting lamp	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)

Bit no.	Assignment	Possible error causes	Remedies
Bit 5	FREE is necessary	See other DIAG bits	Two rising edges at ACK within 4 s, and activate acknowledgment button until ACK_REQ = 0
Bit 6	Acknowledgment necessary	—	—
Bit 7	State of output Q	—	—
Bit 8	State of output MUTING	—	—
Bit 9	FREE active	—	—
Bit 10	Reserved	—	—
...			
Bit 15	Reserved	—	—

Timing imprecision resulting from the update time of the time base used in the instruction:



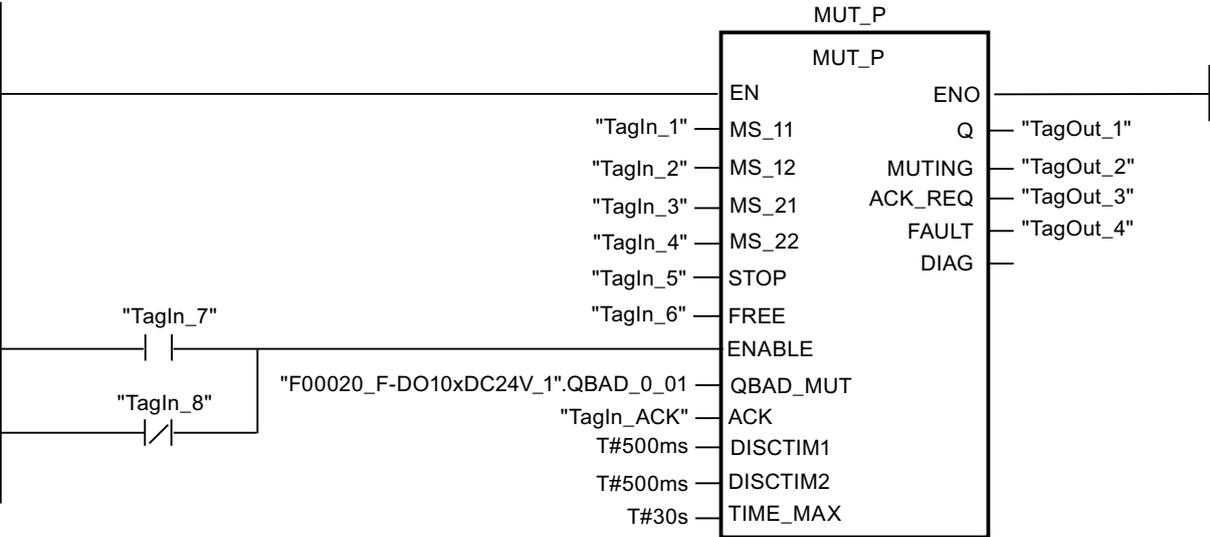
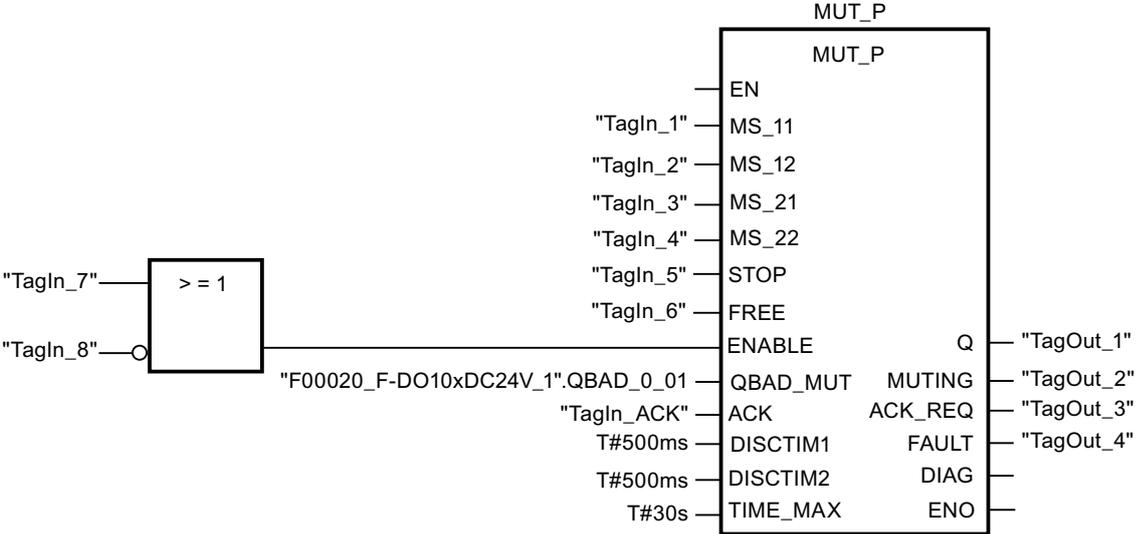
-----> = Time base update

.....> = Call time of an instruction with time processing

- ① For the first call in cycle n+1, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle n+1 are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle n+1. This does not involve another time update (by Δ_2).
- ③ For the call in cycle n+2, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle n+2. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle n+1.

Example

The following example shows how the instruction works:



13.2.3.6 EV1oo2DI: 1oo2 evaluation with discrepancy analysis (STEP 7 Safety Advanced V11)

Description

This instruction implements a 1oo2 evaluation of two single-channel sensors combined with a discrepancy analysis.

Output Q is set to 1, if the signal states of inputs IN1 and IN2 both equal 1 and no discrepancy error DISC_FLT is stored. If the signal state of one or both inputs is 0, output Q is set to 0.

As soon as the signal states of inputs IN1 and IN2 are different, the discrepancy time DISCTIME is started. If the signal states of the two inputs are still different once the discrepancy time expires, a discrepancy error is detected and DISC_FLT is set to 1 (restart inhibit).

If the discrepancy between inputs IN1 and IN2 is no longer detected, the discrepancy error is acknowledged according to the parameter assignment of ACK_NEC:

- If ACK_NEC = 0, the acknowledgment is automatic.
- If ACK_NEC = 1, you must use a rising edge at input ACK to acknowledge the discrepancy error.

ACK_REQ = 1 signals that a user acknowledgment at input ACK is necessary to acknowledge the discrepancy error (cancel the restart inhibit). The instruction sets ACK_REQ = 1 as soon as discrepancy is no longer detected. After acknowledgment or if, prior to acknowledgment, there is once again a discrepancy between inputs IN1 and IN2, the instruction resets ACK_REQ to 0.

Output Q can never be set to 1 if the discrepancy time setting is < 0 or > 60 s. In this case, output DISC_FLT is also set to 1 (restart inhibit). The call interval of the safety program (e.g., OB35) must be less than the discrepancy time setting.

Every call of the "1oo2 evaluation with discrepancy analysis" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., EV1oo2DI_DB_1) or a multi-instance (e.g., EV1oo2DI_Instance_1) for the "1oo2 evaluation with discrepancy analysis" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

WARNING

The ACK_NEC tag must not be assigned a value of 0 unless an automatic restart of the affected process is otherwise excluded. (S033)

⚠ WARNING

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction")
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision. (S034)

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN1	Input	BOOL	Sensor 1
IN2	Input	BOOL	Sensor 2
DISCTIME	Input	TIME	Discrepancy time (0 to 60 s)
ACK_NEC	Input	BOOL	1 = acknowledgment necessary for discrepancy error
ACK	Input	BOOL	Acknowledgment of discrepancy error
Q	Output	BOOL	Output
ACK_REQ	Output	BOOL	1 = acknowledgement required
DISC_FLT	Output	BOOL	1 = discrepancy error
DIAG	Output	BYTE	Service information

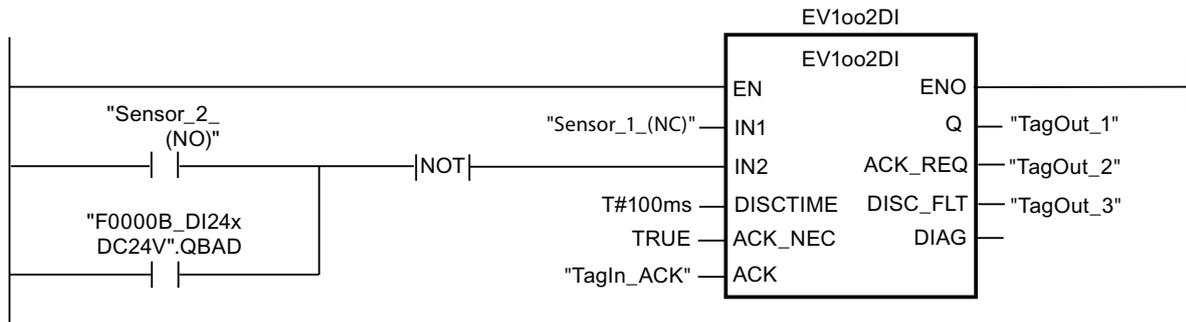
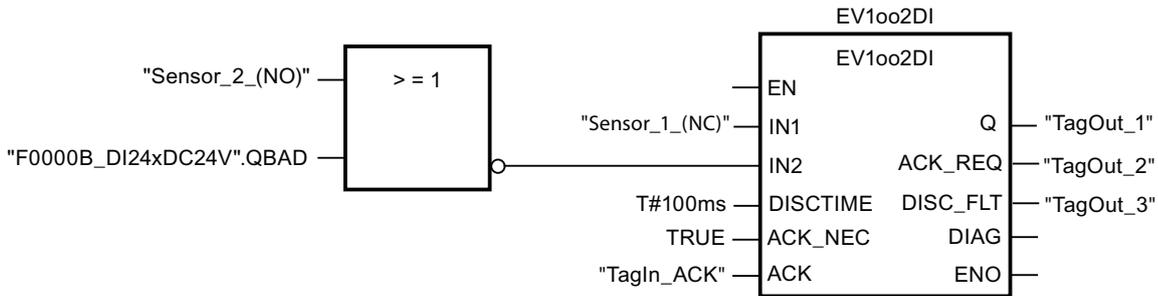
Activating inputs IN1 and IN2

Inputs IN1 and IN2 must both be activated in such a way that their safe state is 0.

Example

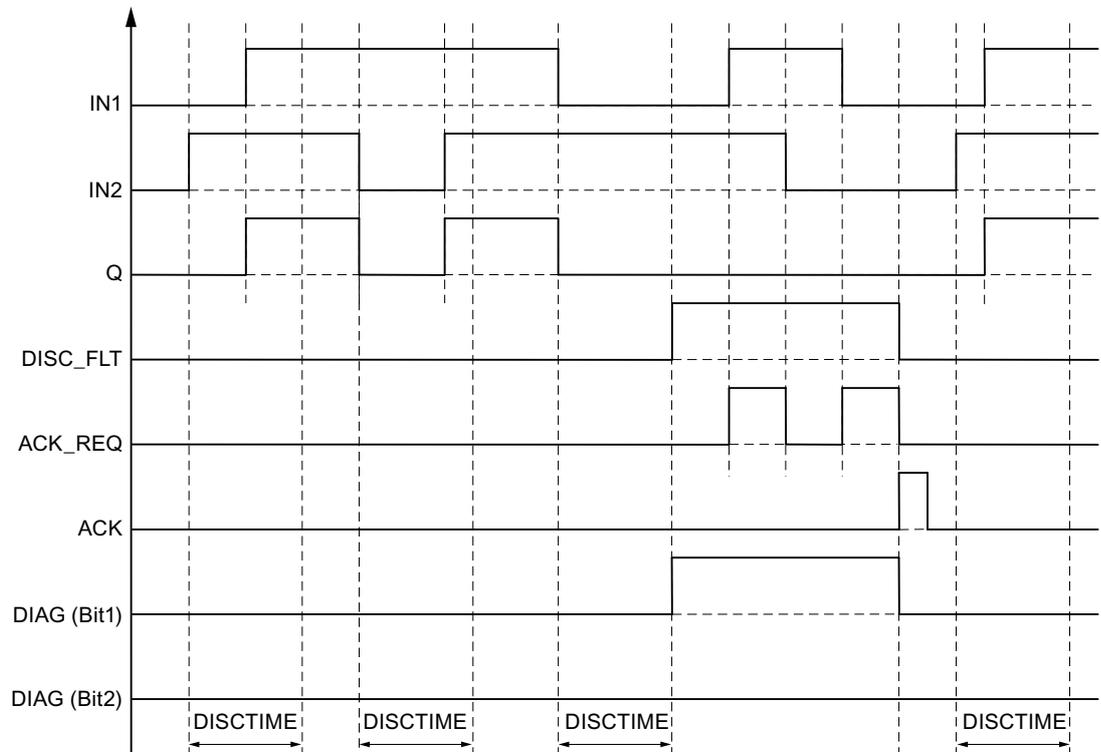
For nonequivalent signals, you have to invert the input (IN1 or IN2) to which you have assigned the sensor signal with a safe state of 1. You must also OR the sensor signal with the QBAD or QBAD_I_xx tag of the associated F-I/O DB or channel, so that a signal state of 0 is present at input IN1 or IN2 (after inversion) if fail-safe values are output.

Network1: EV1oo2DI with nonequivalent signals



Timing diagrams EV1oo2DI

If ACK_NEC = 1:



Startup characteristics

Note

If the sensors at inputs IN1 and IN2 are assigned to different F-I/O, it is possible that the fail-safe values are output for different lengths of time following startup of the F-system due to different startup characteristics of the F-I/O. If the signal states of inputs IN1 and IN2 remain different after the discrepancy time DISCTIME has expired, a discrepancy error is detected after the F-system starts up.

If ACK_NEC = 1 you must acknowledge the discrepancy error with a rising edge at input ACK.

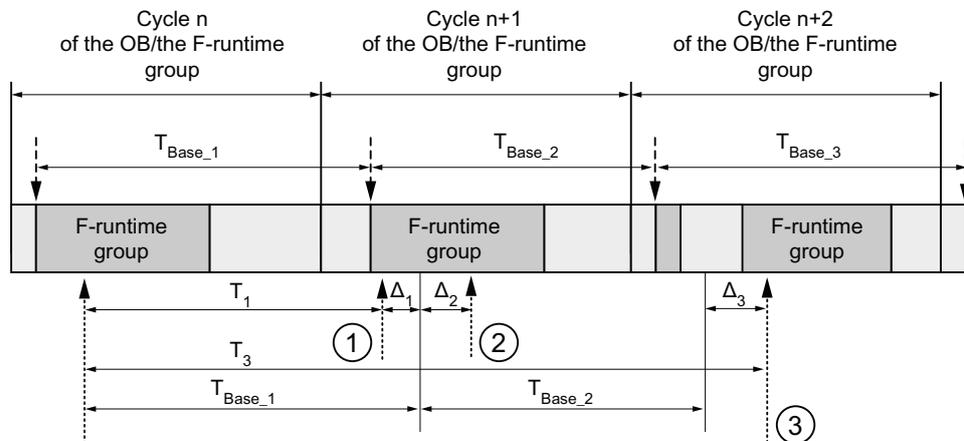
Output DIAG

The DIAG output provides non-fail-safe information on errors for service purposes. You can read out this information by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program. DIAG bits are saved until acknowledgment at input ACK.

Structure of DIAG

Bit No.	Assignment	Possible error causes	Remedies
Bit 0	Discrepancy error or incorrect discrepancy time setting (= status of DISC_FLT)	Sensor defective	Check sensors
		Wiring fault	Check wiring of sensors
		Sensors are wired to different F-I/O, and F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON on an F-I/O	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
		Discrepancy time setting is too low	If necessary, set a higher discrepancy time
		Discrepancy time setting is < 0 s or > 60 s	Set discrepancy time in range between 0 s and 60 s
Bit 1	For discrepancy errors: last signal state change was at input IN1	—	—
Bit 2	For discrepancy errors: last signal state change was at input IN2	—	—
Bit 3	Reserved	—	—
Bit 4	Reserved	—	—
Bit 5	For discrepancy errors: input ACK has a permanent signal state of 1	Acknowledgment button defective	Replace acknowledgment button
		Wiring fault	Check wiring of acknowledgment button
Bit 6	Acknowledgment necessary	—	—
Bit 7	State of output Q	—	—

Timing imprecision resulting from the update time of the time base used in the instruction:



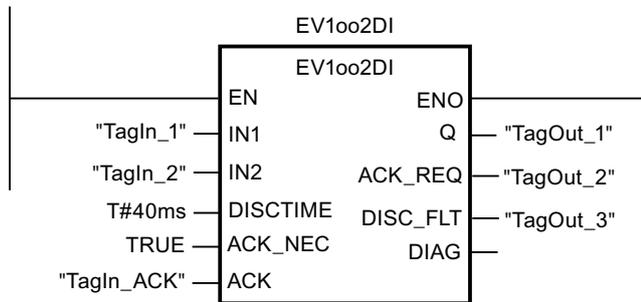
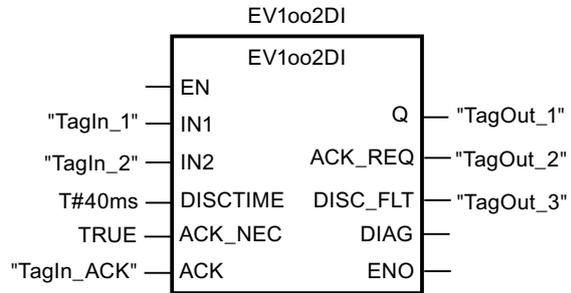
-----> = Time base update

.....> = Call time of an instruction with time processing

- ① For the first call in cycle n+1, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle n+1 are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle n+1. This does not involve another time update (by Δ_2).
- ③ For the call in cycle n+2, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle n+2. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle n+1.

Example

The following example shows how the instruction works:



13.2.3.7 FDBACK: Feedback monitoring (STEP 7 Safety Advanced V11)

Description

This instruction implements feedback monitoring.

To do this, the signal state of the output Q is checked for equality with the inverse signal state of the feedback input FEEDBACK.

Output Q is set to 1 as soon as input ON = 1. Requirement for this is that the feedback input FEEDBACK = 1 and no feedback error is saved.

Output Q is reset to 0, as soon as input ON = 0 or if a feedback error is detected.

A feedback error ERROR = 1 is detected if the inverse signal state of the feedback input FEEDBACK (to input Q) does not follow the signal state of output Q within the maximum tolerable feedback time. The feedback error is saved.

If a discrepancy is detected between the feedback input FEEDBACK and the output Q after a feedback error, the feedback error is acknowledged in accordance with the parameter assignment of ACK_NEC:

- If ACK_NEC = 0, the acknowledgment is automatic.
- If ACK_NEC = 1, you must acknowledge the feedback error with a rising edge at input ACK.

The ACK_REQ = 1 output then signals that a user acknowledgment is necessary at input ACK to acknowledge the feedback error. Following an acknowledgment, the instruction resets ACK_REQ to 0.

To avoid a feedback error from being detected and an acknowledgment from being required when the F-I/O controlled by output Q are passivated, you must supply input QBAD_FIO with the QBAD or QBAD_O_xx tag of the associated F-I/O.

Every call of the "Feedback monitoring" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., FDBACK_DB_1) or a multi-instance (e.g., FDBACK_Instance_1) for the "Feedback monitoring" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

WARNING

The ACK_NEC tag must not be assigned a value of 0 unless an automatic restart of the affected process is otherwise excluded. (S033)

⚠ WARNING

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction")
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

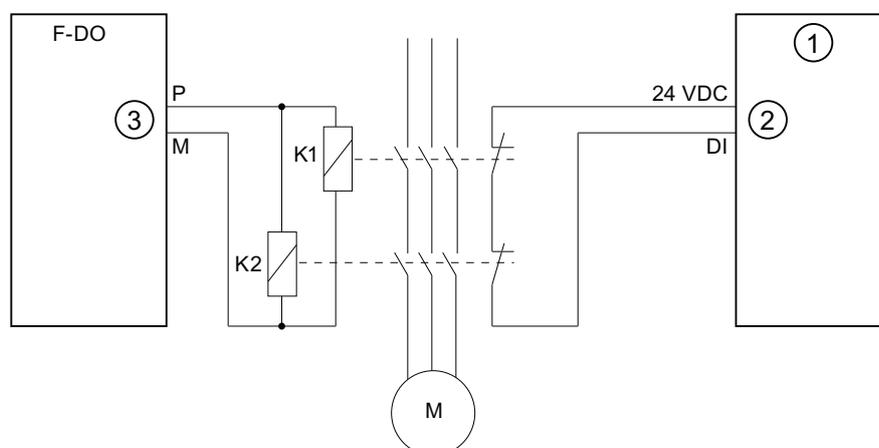
You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision (S034).

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
ON	Input	BOOL	1= Enable output
FEEDBACK	Input	BOOL	Feedback input
QBAD_FIO	Input	BOOL	QBAD or QBAD_O_xx signal of F-I/O/channel of output Q (F-I/O DB)
ACK_NEC	Input	BOOL	1=Acknowledgment necessary
ACK	Input	BOOL	Acknowledgment
FDB_TIME	Input	TIME	Feedback time
Q	Output	BOOL	Output
ERROR	Output	BOOL	Feedback error
ACK_REQ	Output	BOOL	Acknowledgment request
DIAG	Output	BYTE	Service information

Interconnection example



- ① Standard DI
- ② Input FEEDBACK
- ③ Output Q

The feedback contact is wired to a standard I/O module.

Instruction versions

Two versions are available for this instruction:

- Version 1.0

When projects that were created with *S7 Distributed Safety V5.4 SP5* are migrated, Version 1.0 of the instruction is used automatically.

Version 1.0 requires that the F_TOF block with the number FB 186 is available in the project tree in the "Program blocks/System blocks/STEP 7 Safety" folder.

If you want to compile a migrated safety program with *STEP 7 Safety Advanced V11* for the first time, we recommend that you update the version of the FDBACK instruction to Version 1.1 beforehand. You will then avoid number conflicts.

- Version 1.1

When a new project is created with *STEP 7 Safety Advanced V11*, Version V1.1 is preset automatically. This version is functionally identical to Version V1.0, but does not require the F_TOF block to have a particular number.

For more information on the use of instruction versions, refer to the help on *STEP 7 Professional* under "Using instruction versions".

Startup characteristics

After an F-system startup, the instruction does not have to be acknowledged when no errors are present.

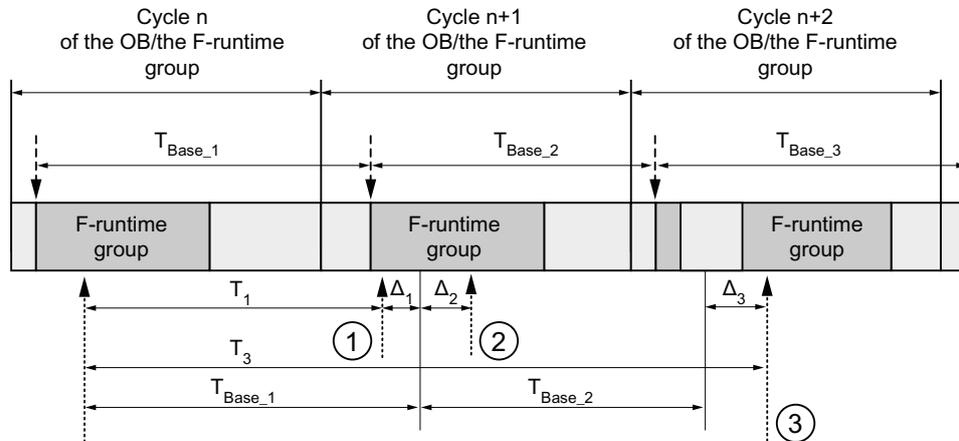
Output DIAG

The DIAG output provides non-fail-safe information on errors for service purposes. You can read out this information by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program. DIAG bits 0, 2, and 5 are saved until acknowledgment at input ACK.

Structure of DIAG

Bit No.	Assignment	Possible error causes	Remedies
Bit 0	Feedback error or incorrect feedback time setting (= state of ERROR)	Feedback time setting < 0	Set feedback time > 0
		Feedback time setting is too low	If necessary, set a higher feedback time
		Wiring fault	Check wiring of actuator and feedback contact
		Actuator or feedback contact is defective	Check actuator and feedback contact
		I/O fault or channel fault of feedback input	Check I/O
Bit 1	Passivation of F-I/O/channel controlled by output Q (= state of QBAD_FIO)	F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON of F-I/O	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
Bit 2	After feedback error: feedback input has permanent signal state of 0	I/O fault or channel fault of feedback input	Check I/O
		Feedback contact is defective	Check feedback contact
		F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON of F-I/O of feedback input	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
Bit 3	Reserved	—	—
Bit 4	Reserved	—	—
Bit 5	For feedback error: input ACK has a permanent signal state of 1	Acknowledgment button defective	Check acknowledgment button
		Wiring fault	Check wiring of acknowledgment button
Bit 6	Acknowledgement required (= state of ACK_REQ)	—	—
Bit 7	State of output Q	—	—

Timing imprecision resulting from the update time of the time base used in the instruction:



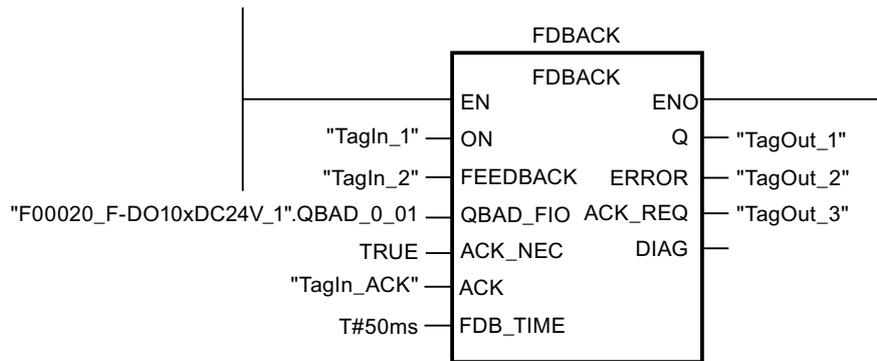
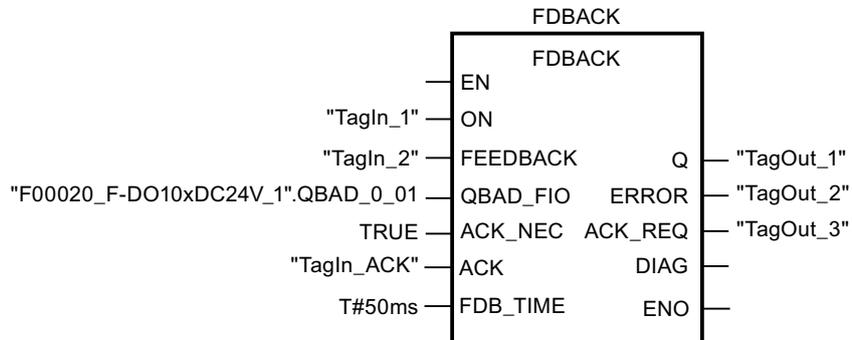
-----> = Time base update

.....> = Call time of an instruction with time processing

- ① For the first call in cycle n+1, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle n+1 are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle n+1. This does not involve another time update (by Δ_2).
- ③ For the call in cycle n+2, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle n+2. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle n+1.

Example

The following example shows how the instruction works:



13.2.3.8 SFDOOR: Safety door monitoring (STEP 7 Safety Advanced V11)

Description

This instruction implements safety door monitoring.

Enable signal Q is reset to 0 as soon as one of the inputs IN1 or IN2 take a signal state of 0 (safety door is opened). The enable signal can be reset to 1, only if:

- Inputs IN1 and IN2 both take a signal state of 0 prior to opening the door (safety door has been completely opened)
- Inputs IN1 and IN2 then both take a signal state of 1 (safety door is closed)
- An acknowledgment occurs

The acknowledgment for the enable takes place according to the parameter assignment at input ACK_NEC:

- If ACK_NEC = 0, the acknowledgment is automatic.
- If ACK_NEC = 1, you must use a rising edge at input ACK for acknowledging the enable.

Output ACK_REQ = 1 is used to signal that a user acknowledgment is required at input ACK for the acknowledgment. The instruction sets ACK_REQ = 1 as soon as the door is closed. Following an acknowledgment, the instruction resets ACK_REQ to 0.

In order for the instruction to recognize whether inputs IN1 and IN2 are 0 merely due to passivation of the associated F-I/O, you must supply inputs QBAD_IN1 or QBAD_IN2 with the QBAD or QBAD_I_xx tag of the associated F-I/O or channel. Among other things, this will prevent you from having to open the safety door completely prior to an acknowledgment in the event the F-I/O are passivated.

Every call of the "Safety door monitoring" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., SFDOOR_DB_1) or a multi-instance (e.g., SFDOOR_Instance_1) for the "Safety door monitoring" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

WARNING

The ACK_NEC tag must not be assigned a value of 0 unless an automatic restart of the affected process is otherwise excluded. (S033)

Parameters

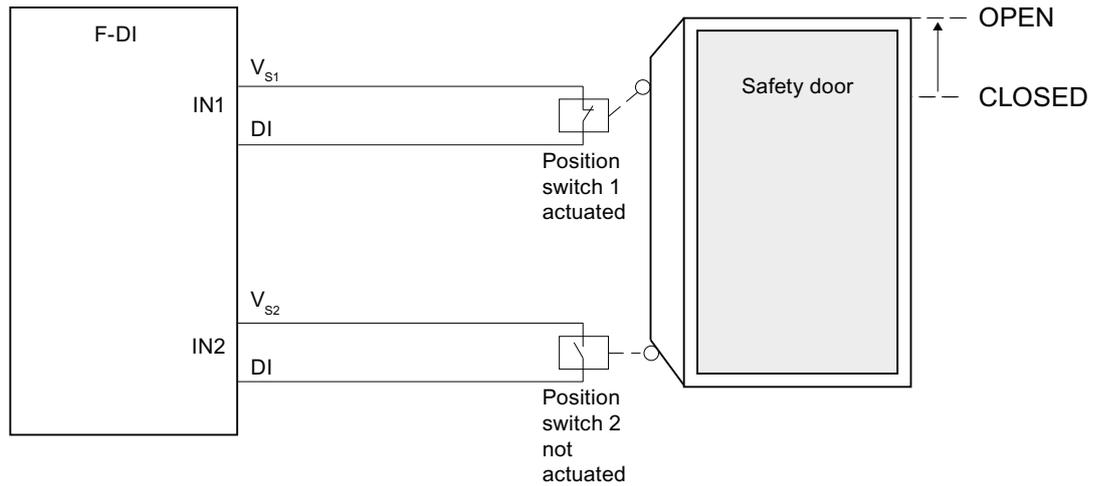
The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN1	Input	BOOL	Input 1
IN2	Input	BOOL	Input 2
QBAD_IN1	Input	BOOL	QBAD or QBAD_I_xx signal of F-I/O/channel of input IN1 (F-I/O)
QBAD_IN2	Input	BOOL	QBAD or QBAD_I_xx signal of F-I/O/channel of input IN2 (F-I/O)
OPEN_NEC	Input	BOOL	1= Open necessary at startup
ACK_NEC	Input	BOOL	1=Acknowledgment necessary
ACK	Input	BOOL	Acknowledgment
Q	Output	BOOL	1= Enable, safety door closed
ACK_REQ	Output	BOOL	Acknowledgment request
DIAG	Output	BYTE	Service information

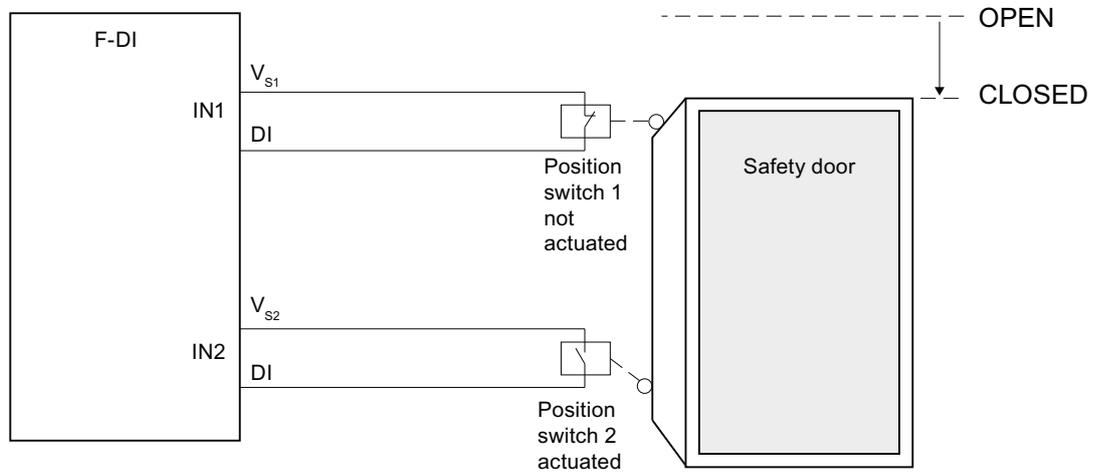
Interconnection example

You must interconnect the NC contact of position switch 1 of the safety door at input IN1 and the NO contact of position switch 2 at input IN2. Position switch 1 must be mounted in such a way that it is positively operated when the safety door is open. Position switch 2 must be mounted in such a way that it is operated when the safety door is closed.

Safety door open:



Safety door closed:



Startup characteristics

After an F-system startup, enable signal Q is reset to 0. The acknowledgment for the enable takes place according to the parameter assignment at inputs OPEN_NEC and ACK_NEC:

- When OPEN_NEC = 0, an automatic acknowledgement occurs **independently** of ACK_NEC, as soon as the two inputs IN1 and IN2 take signal state 1 for the first time following reintegration of the associated F-I/O (safety door is closed).
- When OPEN_NEC = 1 or if at least one of the IN1 and IN2 inputs still has a signal state of 0 after reintegration of the associated F-I/O, an automatic acknowledgment occurs **according** to ACK_NEC or you have to use a rising edge at input ACK for the enable. Prior to acknowledgment, inputs IN1 and IN2 both have to take a signal state of 0 (safety door has been completely opened) followed by a signal state of 1 (safety door is closed).

 **WARNING**

The OPEN_NEC tag must not be assigned a value of 0 unless an automatic restart of the affected process is otherwise excluded. (S039)

Output DIAG

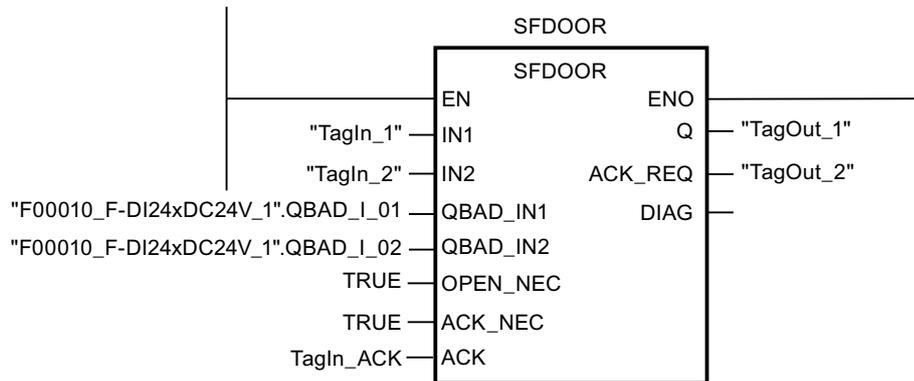
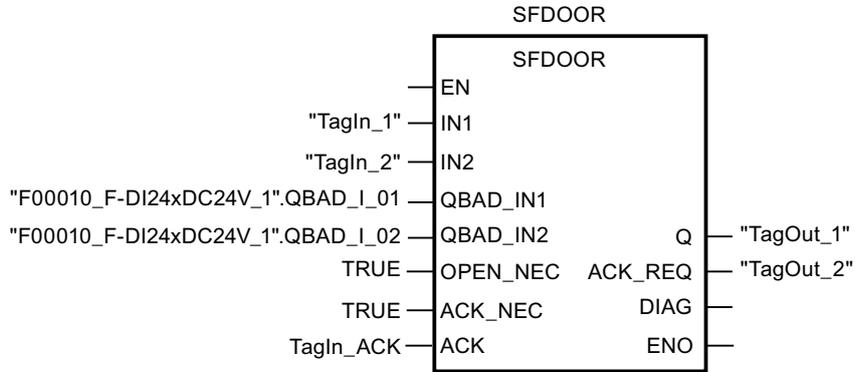
The DIAG output provides non-fail-safe information on errors for service purposes. You can read out this information by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program.

Structure of DIAG

Bit No.	Assignment	Possible error causes	Remedies
Bit 0	Reserved	—	—
Bit 1	Signal state 0 is missing at both IN1 and IN2 inputs	Safety door was not completely opened when OPEN_NEC = 1 after F-system startup	Open safety door completely
		Open safety door was not completely opened	Open safety door completely
		Wiring fault	Check wiring of position switch
		Position switch is defective	Check position switch
		Position switch is incorrectly adjusted	Adjust position switch properly
Bit 2	Signal state 1 is missing at both IN1 and IN2 inputs	Safety door was not closed	Close safety door
		Wiring fault	Check wiring of position switch
		Position switch is defective	Check position switch
		Position switch is incorrectly adjusted	Adjust position switch properly
Bit 3	QBAD_IN1 and/or QBAD_IN2 = 1	F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON of F-I/O or channel of IN1 and/or IN2	For a solution, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
Bit 4	Reserved	—	—
Bit 5	If enable is missing: input ACK has a permanent signal state of 1	Acknowledgment button defective	Check acknowledgment button
		Wiring fault	Check wiring of acknowledgment button
Bit 6	Acknowledgement required (= state of ACK_REQ)	—	—
Bit 7	State of output Q	—	—

Example

The following example shows how the instruction works:



13.2.3.9 ACK_GL: Global acknowledgment of all F-I/O in an F-runtime group (STEP 7 Safety Advanced V11)

Description

This instruction creates an acknowledgment for the simultaneous reintegration of all F-I/O or channels of the F-I/O of an F-runtime group after communication errors, F-I/O errors, or channel faults.

A user acknowledgment (Page 99) with a positive edge at input ACK_GLOB is required for reintegration. The acknowledgement occurs analogously to the user acknowledgment via the ACK_REI tag of the F-I/O DB (Page 79), but it acts simultaneously on all F-I/O of the F-runtime group in which the instruction is called.

If you use the instruction ACK_GL, you do not have to provide for a user acknowledgment for each F-I/O of the F-runtime group via the ACK_REI tag of the F-I/O DB.

Every call of the "Global acknowledgment of all F-I/O of a runtime group" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., ACK_GL_DB_1) or a multi-instance (e.g., ACK_GL_Instance_1) for the "Global acknowledgment of all F-I/O of a runtime group" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
ACK_GLOB	Input	BOOL	1=acknowledgment for reintegration

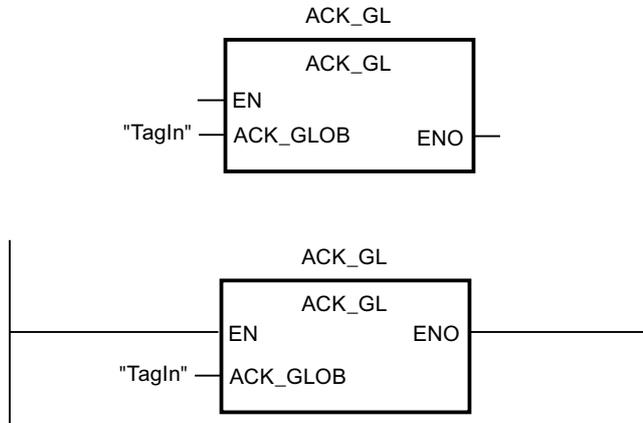
Note

An acknowledgment via the ACK_GL instruction is only possible if the tag ACK_REI of the F-I/O DB = 0. Accordingly, an acknowledgment via the tag ACK_REI of the F-I/O DB is only possible if the input ACK_GLOB of the instruction = 0.

The instruction is only allowed to be called once per F-runtime group.

Example

The following example shows how the instruction works:



13.2.4 Timer operations

13.2.4.1 TP: Generate pulse (STEP 7 Safety Advanced V11)

Description

You can use the "Generate pulse" instruction to set output Q for an assigned period. The instruction is started if the result of logic operation (RLO) changes from "0" to "1" (positive edge) at input IN. The assigned period PT starts running when the instruction starts. Output Q is set for period PT, regardless of the subsequent sequence of the input signal. Also the detection of a new positive signal edge does not influence the signal state at output Q as long as period PT runs.

The current time value can be queried at output ET. The time value begins at T#0s and ends when the value of period PT is reached. If period PT is reached and the signal state at input IN is "0", output ET is reset.

Every call of the "Generate pulse" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., F_IEC_Timer_DB_1) or a multi-instance (e.g., F_IEC_Timer_Instance_1) for the "Generate pulse" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the *help on STEP 7 Professional*.

WARNING

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction").
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision. (S034)

The operating system resets the instances of the "Generate pulse" instruction on a startup of the F-system.

Note

The functionality of this instruction differs from the corresponding standard TP instruction in the following points:

- When a call is made with PT = 0 ms, the TP instance is not reset completely (initialized). The instruction behaves in accordance with the timing diagrams: Only outputs Q and ET are reset. To restart the pulse, a new rising signal edge at input IN is required once PT is greater than 0 again.
- A call with PT < 0 ms resets outputs Q and ET. To restart the pulse, a new rising signal edge at input IN is required once PT is greater than 0 again.

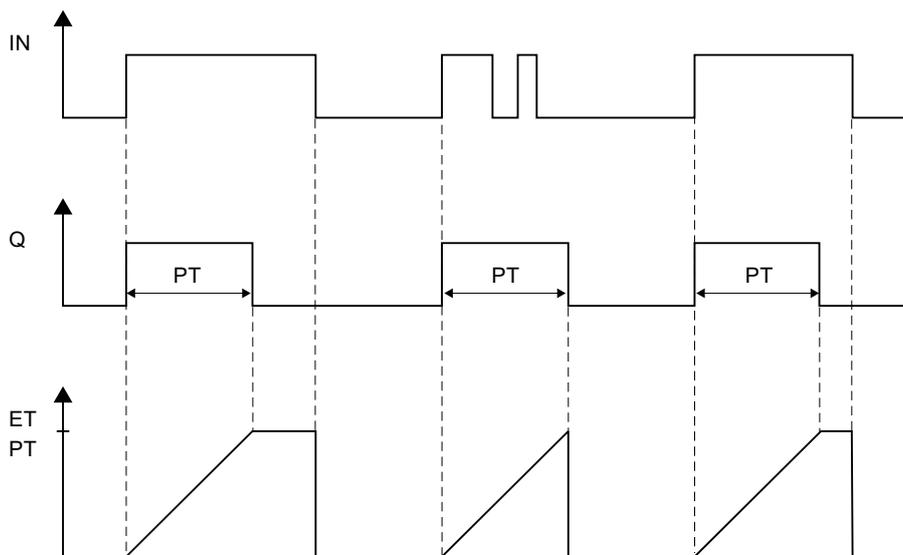
Parameters

The following table shows the parameters of the instruction:

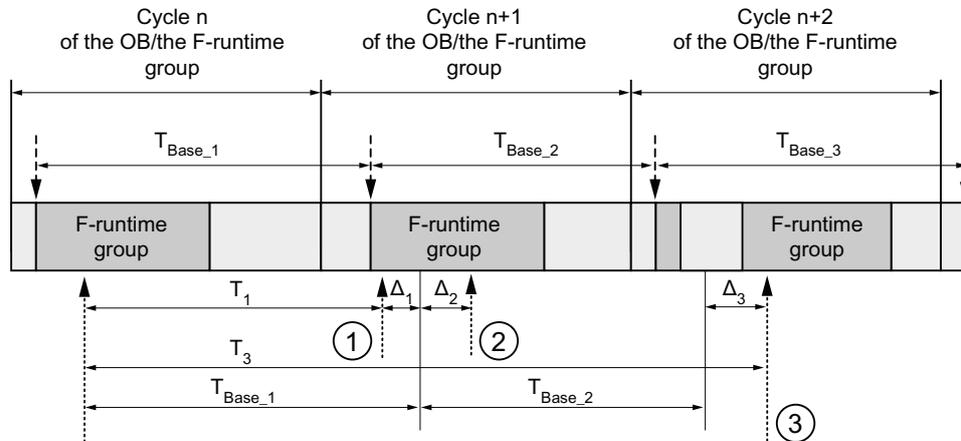
Parameter	Declaration	Data type	Description
IN	Input	BOOL	Start input
PT	Input	TIME	Duration of pulse; must be positive.
Q	Output	BOOL	Pulse output
ET	Output	TIME	Current time value

Pulse diagram

The following figure shows the pulse diagram of the instruction:



Timing imprecision resulting from the update time of the time base used in the instruction:



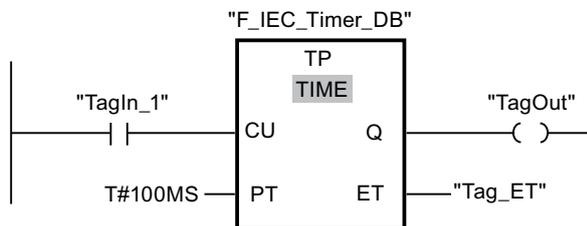
-----> = Time base update

.....> = Call time of an instruction with time processing

- ① For the first call in cycle n+1, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle n+1 are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle n+1. This does not involve another time update (by Δ_2).
- ③ For the call in cycle n+2, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle n+2. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle n+1.

Example

The following example shows how the instruction works:



If the signal state of operand "TagIn_1" changes from "0" to "1", the "Generate pulse" instruction is started and the period assigned at input PT (100 ms) runs, regardless of the further development of operand "TagIn_1".

Operand "TagOut" at output Q has signal state "1" as long as the period is running. Operand "Tag_ET" contains the current time value.

13.2.4.2 TON: Generate on-delay (STEP 7 Safety Advanced V11)

Description

You can use the "Generate on-delay" instruction to delay the setting of output Q by the assigned period PT. The "Generate on-delay" instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive edge). The assigned period PT starts running when the instruction starts. When period PT has expired, output Q delivers the signal state "1". Output Q remains set as long as start input delivers "1". When the signal state at the start input changes from "1" to "0", output Q is reset. The time function is restarted when a new positive signal edge is detected at the start input.

The current time value can be queried at output ET. The time value begins at T#0s and ends when the value of period PT is reached. Output ET is reset, as soon as the signal state at input IN changes to "0".

Every call of the "Generate on-delay" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., F_IEC_Timer_DB_1) or a multi-instance (e.g., F_IEC_Timer_Instance_1) for the "Generate on-delay" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the *help on STEP 7 Professional*.

WARNING

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction")
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision. (S034)

The operating system resets the instances of the "Generate on-delay" instruction on a startup of the F-system.

Note

The functionality of this instruction differs from the corresponding standard TON instruction in the following points:

- When a call is made with $PT = 0$ ms, the instance of the TON is not reset completely (initialized). The instruction behaves in accordance with the timing diagrams: Only output ET is reset. To restart the on-delay, a new rising signal edge at input IN is required once PT is greater than 0 again.
- A call with $PT < 0$ ms resets outputs Q and ET. To restart the on-delay, a new rising signal edge at input IN is required once PT is greater than 0 again.

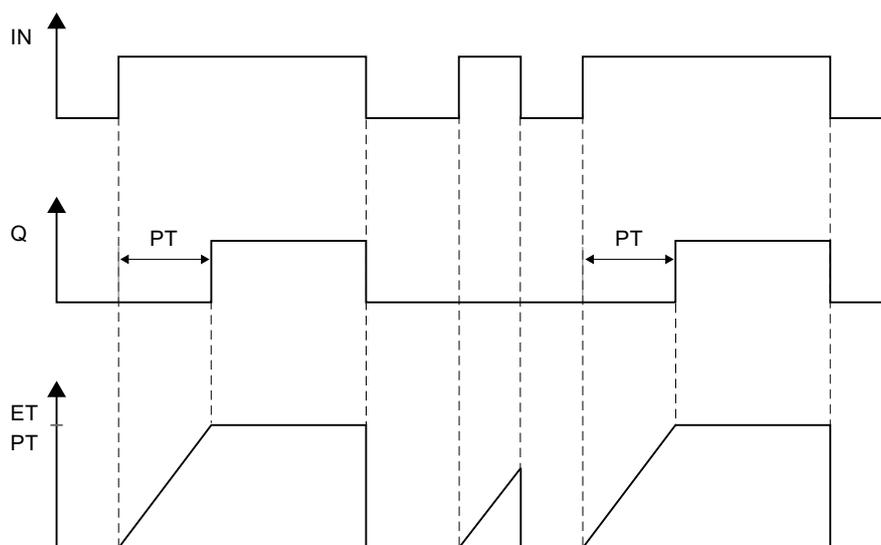
Parameters

The following table shows the parameters of the instruction:

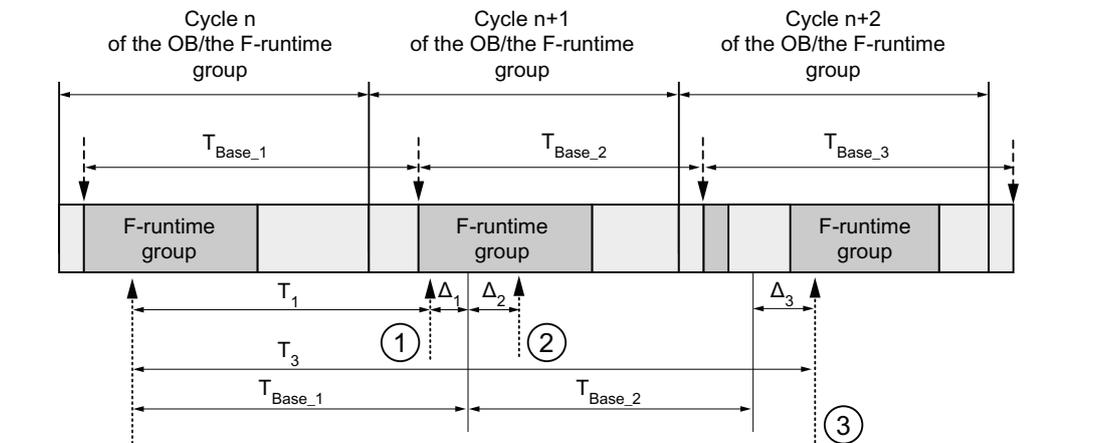
Parameter	Declaration	Data type	Description
IN	Input	BOOL	Start input
PT	Input	TIME	Duration of on-delay; must be positive.
Q	Output	BOOL	Output that is set after expiration of time PT.
ET	Output	TIME	Current time value

Pulse diagram

The following figure shows the pulse diagram of the instruction:



Timing imprecision resulting from the update time of the time base used in the instruction:



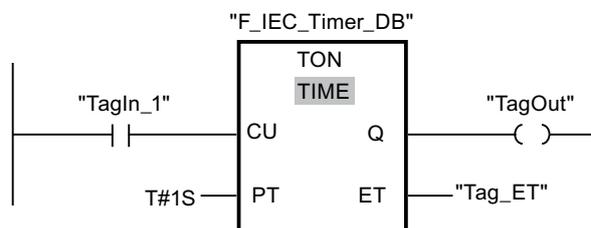
-----> = Time base update

.....> = Call time of an instruction with time processing

- ① For the first call in cycle $n+1$, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle $n+1$ are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle $n+1$. This does not involve another time update (by Δ_2).
- ③ For the call in cycle $n+2$, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle $n+2$. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle $n+1$.

Example

The following example shows how the instruction works:



When the signal state of operand "TagIn_1" changes from "0" to "1", the "Generate on-delay" instruction is started and the period assigned at input PT (1 s) runs. Operand "TagOut" at output Q feeds signal state "1" when the period has elapsed and remains set as long as operand "TagIn_1" still feeds signal state "1". Operand "Tag_ET" contains the current time value.

13.2.4.3 TOF: Generate off-delay (STEP 7 Safety Advanced V11)

Description

You can use the "Generate off-delay" instruction to delay resetting output Q by the assigned period PT. Output Q is set if the result of logic operation (RLO) changes from "0" to "1" (positive edge) at input IN. The assigned period PT starts when the signal state at input IN changes back to "0". Output Q remains set as long as period PT runs. After period PT expires, output Q is reset. If the signal state at input IN changes to "1" before period PT has expired, then the time is reset. The signal state at output Q remains at "1".

The current time value can be queried at output ET. The time value begins at T#0s and ends when the value of period PT is reached.

Every call of the "Generate off-delay" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., F_IEC_Timer_DB_1) or a multi-instance (e.g., F_IEC_Timer_Instance_1) for the "Generate off-delay" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the *help on STEP 7 Professional*.

WARNING

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction")
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision. (S034)

The operating system resets the instances of the "Generate off-delay" instruction on a startup of the F-system.

Note

The functionality of this instruction differs from the corresponding standard TOF instruction in the following points:

- When a call is made with PT = 0 ms, the instance of the TOF is not reset completely (initialized). The instruction behaves in accordance with the timing diagrams: only outputs Q and ET are reset. To restart the off-delay, another falling signal edge at input IN is required once PT is greater than 0 again.
- A call with PT < 0 ms resets outputs Q and ET. To restart the off-delay, another falling signal edge at input IN is required once PT is greater than 0 again.

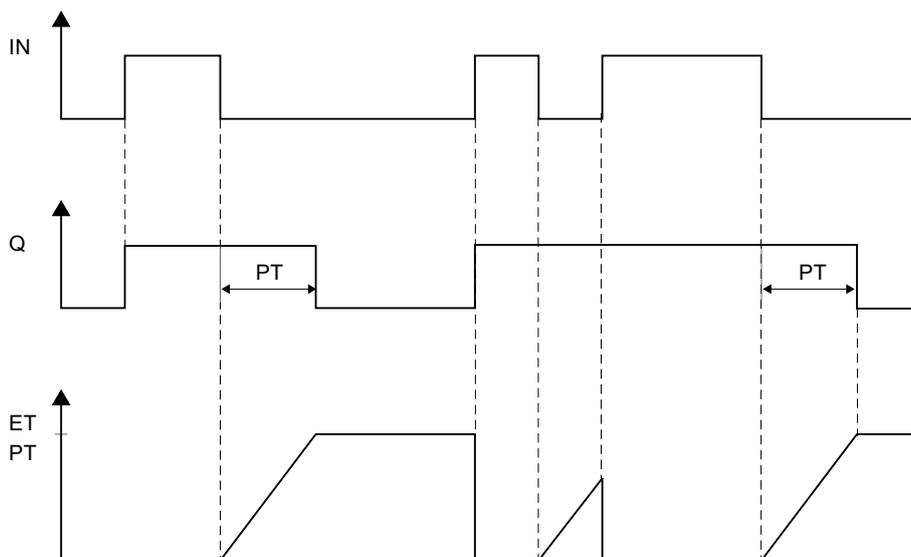
Parameters

The following table shows the parameters of the instruction:

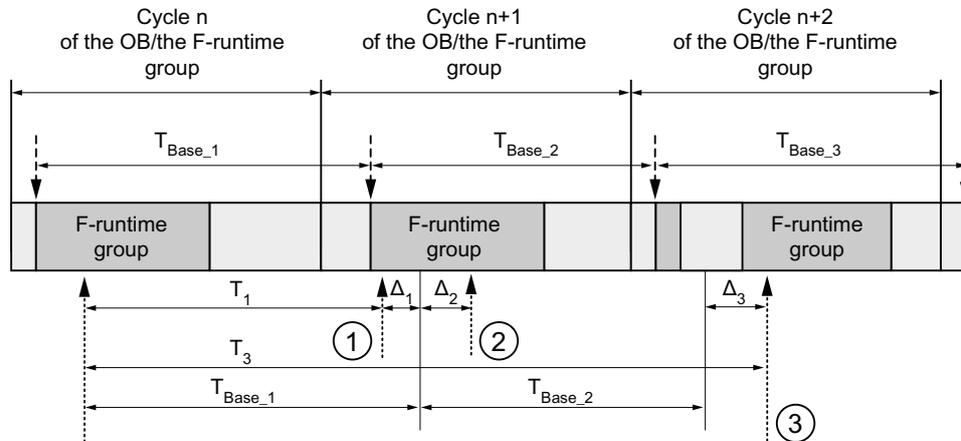
Parameter	Declaration	Data type	Description
IN	Input	BOOL	Start input
PT	Input	TIME	Duration of off delay; must be positive.
Q	Output	BOOL	Output that is reset after expiration of time PT.
ET	Output	TIME	Current time value

Pulse diagram

The following figure shows the pulse diagram of the instruction:



Timing imprecision resulting from the update time of the time base used in the instruction:



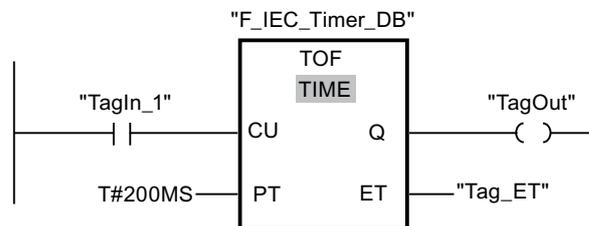
-----> = Time base update

.....> = Call time of an instruction with time processing

- ① For the first call in cycle n+1, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle n+1 are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle n+1. This does not involve another time update (by Δ_2).
- ③ For the call in cycle n+2, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle n+2. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle n+1.

Example

The following example shows how the instruction works:



If the signal state of operand "TagIn_1" changes from "0" to "1", the signal state of operand "TagOut" at output Q is set to "1".

If the signal state of operand "TagIn_1" changes back to "0", the period assigned at input PT (200 ms) runs.

The "TagOut" operand at output Q is set back to "0" when the period expires. Operand "Tag_ET" contains the current time value.

13.2.5 Counter operations

13.2.5.1 CTU: Count up (STEP 7 Safety Advanced V11)

Description

You can use the "Count up" instruction to increment the value at output CV. When the signal state at input CU changes from "0" to "1" (positive signal edge), the instruction is executed and the current count value at output CV is increased by one. The count value is increased on each detection of a positive signal edge until it reaches the high limit of the data type specified at output CV. When the high limit is reached, the signal state at input CU no longer affects the instruction.

The counter status can be queried at output Q. The signal state at output Q is determined by parameter PV. When the current count value is greater than or equal to the value of parameter PV, output Q is set to signal state "1". In all other cases, the signal state at output Q is "0".

The value at output CV is reset to zero when the signal state at input R changes to "1". As long as signal state "1" exists at input R, the signal state at input CU has no effect on the instruction.

Every call of the "Count up" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., F_IEC_Counter_DB_1) or a multi-instance (e.g., F_IEC_Counter_Instance_1) for the "Count up" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the *help on STEP 7 Professional*.

The operating system resets the instances of the "Count up" instruction on a startup of the F-system.

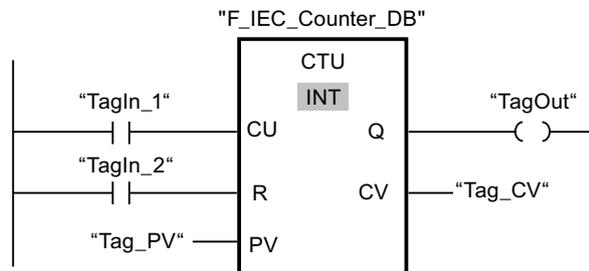
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
CU	Input	BOOL	Counter input
R	Input	BOOL	Reset input
PV	Input	INT	Value for which output Q is set
Q	Output	BOOL	Counter status
CV	Output	INT	Current count value

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count up" instruction is executed and the current count value of the "Tag_CV" operand is increased by one. The count value is increased on every additional positive signal edge until the high limit of the specified data type (32767) is reached.

The value at parameter PV is taken as the limit for the determination of output "TagOut". Output "TagOut" delivers the signal state "1" as long as the current count value is greater than or equal to the value of operand "Tag_PV". In all other cases, output TagOut has signal state "0".

13.2.5.2 CTD: Count down (STEP 7 Safety Advanced V11)

Description

You can use the "Count down" instruction to decrement the value at output CV. When the signal state at input CD changes from "0" to "1" (positive signal edge), the instruction is executed and the current count value at output CV is decreased by one. The count value is decreased on each detection of a positive signal edge until it reaches the low limit of the specified data type. When the low limit is reached, the signal state at input CD no longer affects the instruction.

The counter status can be queried at output Q. When the current count value is less than or equal to zero, output Q is set to signal state "1". In all other cases, the signal state at output Q is "0".

The value at output CV is set to the value of parameter "PV" when the signal state at input LD changes to "1". As long as signal state "1" exists at input LD, the signal state at input CD has no effect on the instruction.

Every call of the "Count down" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can generate a data block (single instance) (e.g., F_IEC_Counter_DB_1) or a multi-instance (e.g., F_IEC_Counter_Instance_1) for the "Count down" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the *help on STEP 7 Professional*.

The operating system resets the instances of the "Count down" instruction on a startup of the F-system.

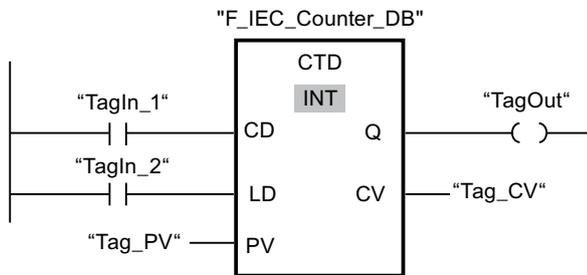
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
CD	Input	BOOL	Counter input
LD	Input	BOOL	Load input
PV	Input	INT	Value for which output Q is set
Q	Output	BOOL	Counter status
CV	Output	INT	Current count value

Example

The following example shows how the instruction works:



If the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count down" instruction is executed and the current count value at output "Tag_CV" is decreased by one. The count value is decreased on each additional positive signal edge until the low limit of the specified data type (-32768) is reached.

Output "TagOut" delivers the signal state "1" as long as the current count value is less than or equal to zero. In all other cases, output TagOut has signal state "0".

13.2.5.3 CTUD: Count up and down (STEP 7 Safety Advanced V11)

Description

You can use the "Count up and down" instruction to increment and decrement the count value at output CV. If the signal state at input CU changes from "0" to "1" (positive signal edge), the current count value at output CV is increased by one. If the signal state at input CD changes from "0" to "1" (positive signal edge), the count value at output CV is decreased by one. If a positive signal edge is present at inputs CU and CD in one program cycle, the current count value at output CV remains unchanged.

The count value can be increased until it reaches the high limit of the data type specified at output CV. When the high limit is reached, the count value is no longer incremented on a positive signal edge. When the low limit of the specified data type is reached, the count value is no longer decreased.

When the signal state at input LD changes to "1", the count value at output CV is set to the value of parameter PV. As long as signal state "1" exists at input LD, the signal state at inputs CU and CD has no effect on the instruction.

The count value is set to zero, when the signal state at input R changes to "1". As long as signal state "1" exists at input R, the signal state at inputs CU, CD, and LD has no effect on the "Count up and down" instruction.

The status of the up counter can be queried at output QU. When the current count value is greater than or equal to the value of parameter PV, output QU delivers signal state "1". In all other cases, the signal state at output QU is "0".

The status of the down counter can be queried at output QD. When the current count value is lesser than or equal to zero, output QD delivers signal state "1". In all other cases, the signal state at output QD is "0".

Every call of the "Count up and down" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., F_IEC_Counter_DB_1) or a multi-instance (e.g., F_IEC_Counter_Instance_1) for the "Count up and down" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the *help on STEP 7 Professional*.

The operating system resets the instances of the "Count up and down" instruction on a cold restart of the F-system.

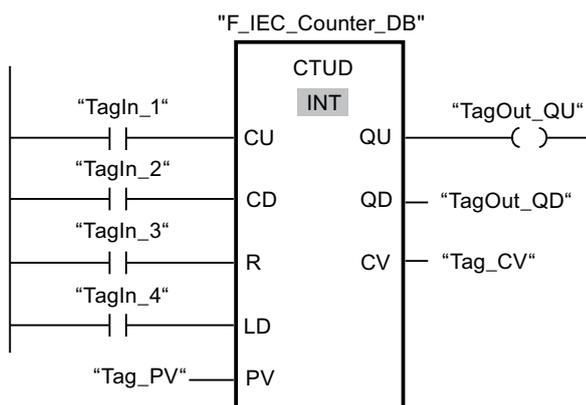
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
CU	Input	BOOL	Count up input
CD	Input	BOOL	Count down input
R	Input	BOOL	Reset input
LD	Input	BOOL	Load input
PV	Input	INT	Value for which output QU is set
QU	Output	BOOL	Status of up counter
QD	Output	BOOL	Status of down counter
CV	Output	INT	Current count value

Example

The following example shows how the instruction works:



When the signal state at input "TagIn_1" or at input "TagIn_2" changes from "0" to "1" (positive signal edge), the "Count up and down" instruction is executed. When a positive signal edge is present at input "TagIn_1", the current count value of the "Tag_CV" operand is increased by one. When a positive signal edge is present at input "TagIn_2", the current count value at output "Tag_CV" is decreased by one. The count value is increased on each positive signal edge at input CU until it reaches the high limit of 32767. The count value is decreased on each positive signal edge at input CD until it reaches the low limit of -32768.

Output "TagOut_QU" delivers the signal state "1" as long as the current count value is greater than or equal to the value at input "Tag_PV". In all other cases, output TagOut_QU has signal state "0".

Output "TagOut_QD" delivers the signal state "1" as long as the current count value is less than or equal to zero. In all other cases, output TagOut_QD has signal state "0".

13.2.6 Comparator operations

13.2.6.1 CMP ==: Equal (STEP 7 Safety Advanced V11)

Description

You can use the "Equal" instruction to determine if the first comparison value (<Operand1>) is equal to the second comparison value (<Operand2>).

If the condition of the comparison is satisfied, the instruction returns result of logic operation (RLO) "1". If the condition of the comparison is not satisfied, the instruction returns RLO "0". The RLO of the instruction is linked to the RLO of the entire current path as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Enter the first comparison value (<Operand1>) in the operand placeholder above the instruction. Enter the second comparison value (<Operand2>) in the operand placeholder below the instruction.

Parameters

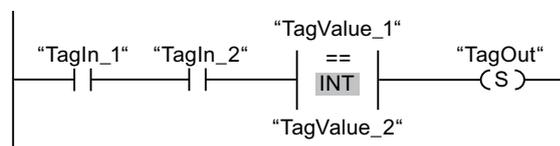
The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand1>	Input	INT, DINT, TIME, WORD, DWORD	First value to compare
<Operand2>	Input	INT, DINT, TIME, WORD, DWORD	Second value to compare

You can select the data type of the instruction in the "<???"> drop-down list in the instruction box.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- Operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled ("TagValue_1" = "TagValue_2").

13.2.6.3 CMP >=: Greater than or equal (STEP 7 Safety Advanced V11)

Description

You can use the "Greater or equal" instruction to determine if the first comparison value (<Operand1>) is greater than or equal to the second comparison value (<Operand2>). Both comparison values must be of the same data type.

If the condition of the comparison is satisfied, the instruction returns result of logic operation (RLO) "1". If the condition of the comparison is not satisfied, the instruction returns RLO "0".

The RLO of the instruction is linked to the RLO of the entire current path as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Enter the first comparison value (<Operand1>) in the operand placeholder above the instruction. Enter the second comparison value (<Operand2>) in the operand placeholder below the instruction.

Parameters

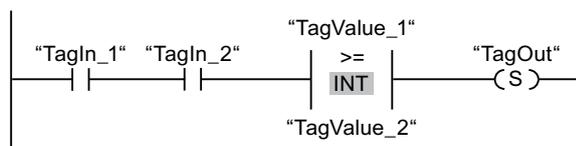
The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand1>	Input	INT, DINT, TIME	First value to compare
<Operand2>	Input	INT, DINT, TIME	Second value to compare

You can select the data type of the instruction in the "<???"> drop-down list in the instruction box.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- Operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled ("TagValue_1" >= "TagValue_2").

13.2.6.4 CMP <=: Less than or equal (STEP 7 Safety Advanced V11)

Description

You can use the "Less or equal" instruction to determine if the first comparison value (<Operand1>) is less than or equal to the second comparison value (<Operand2>). Both comparison values must be of the same data type.

If the condition of the comparison is satisfied, the instruction returns result of logic operation (RLO) "1". If the condition of the comparison is not satisfied, the instruction returns RLO "0".

The RLO of the instruction is linked to the RLO of the entire current path as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Enter the first comparison value (<Operand1>) in the operand placeholder above the instruction. Enter the second comparison value (<Operand2>) in the operand placeholder below the instruction.

Parameters

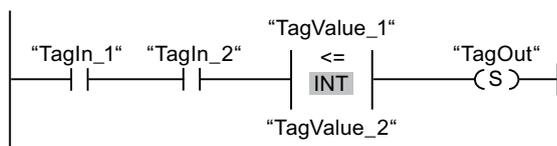
The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand1>	Input	INT, DINT, TIME	First value to compare
<Operand2>	Input	INT, DINT, TIME	Second value to compare

You can select the data type of the instruction in the "<???"> drop-down list in the instruction box.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- Operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled ("TagValue_1" <= "TagValue_2").

13.2.6.5 CMP >: Greater than (STEP 7 Safety Advanced V11)

Description

You can use the "Greater than" instruction to determine if the first comparison value (<Operand1>) is greater than the second comparison value (<Operand2>). Both comparison values must be of the same data type.

If the condition of the comparison is satisfied, the instruction returns result of logic operation (RLO) "1". If the condition of the comparison is not satisfied, the instruction returns RLO "0".

The RLO of the instruction is linked to the RLO of the entire current path as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Enter the first comparison value (<Operand1>) in the operand placeholder above the instruction. Enter the second comparison value (<Operand2>) in the operand placeholder below the instruction.

Parameters

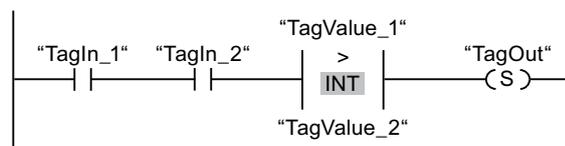
The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand1>	Input	INT, DINT, TIME	First value to compare
<Operand2>	Input	INT, DINT, TIME	Second value to compare

You can select the data type of the instruction in the "<???">" drop-down list in the instruction box.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- Operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled ("TagValue1" > "TagValue2").

13.2.6.6 CMP <: Less than (STEP 7 Safety Advanced V11)

Description

You can use the "Less than" instruction to determine if the first comparison value (<Operand1>) is less than the second comparison value (<Operand2>). Both comparison values must be of the same data type.

If the condition of the comparison is satisfied, the instruction returns result of logic operation (RLO) "1". If the condition of the comparison is not satisfied, the instruction returns RLO "0".

The RLO of the instruction is linked to the RLO of the entire current path as follows:

- By AND, when the comparison instruction is connected in series.
- By OR, when the comparison instruction is connected in parallel.

Enter the first comparison value (<Operand1>) in the operand placeholder above the instruction. Enter the second comparison value (<Operand2>) in the operand placeholder below the instruction.

Parameters

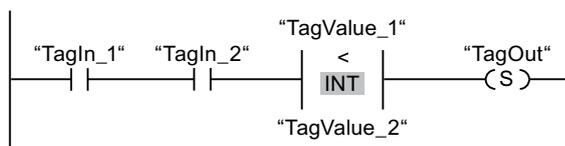
The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand1>	Input	INT, DINT, TIME	First value to compare
<Operand2>	Input	INT, DINT, TIME	Second value to compare

You can select the data type of the instruction in the "<???"> drop-down list in the instruction box.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- Operands "TagIn_1" and "TagIn_2" have signal state "1".
- The condition of the comparison instruction is fulfilled ("TagValue_1" < "TagValue_2").

13.2.7 Math functions

13.2.7.1 ADD: Add (STEP 7 Safety Advanced V11)

Description

You can use the "Add" instruction to add the value at input IN1 and the value at input IN2 and query the sum at the OUT output ($OUT = IN1 + IN2$).

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Note

If the result of the instruction is outside the range permitted for the data type, then the F-CPU goes to STOP when the result is sent to an output at an F-I/O or to a partner F-CPU via safety-related CPU-CPU communication.

One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"
- "Safety program: internal CPU fault; internal error information: 404"

You must therefore ensure that the permitted range for the data type is observed when creating the program!

If this is not possible, you can avoid a STOP of the F-CPU by inserting a "Get status bit OV" instruction in the next network, thereby programming overflow detection into it.

Note the following:

- The result of the instruction behaves like the analogous instruction in a standard block.
 - The network with the "Get status bit OV" instruction must not contain any jump labels.
 - The execution time of the instruction is extended (see also Excel file for response time calculation (<http://support.automation.siemens.com/WW/view/en/49368678/134200>)).
 - A warning is issued if you do not insert a "Get status bit OV" instruction.
-

Parameters

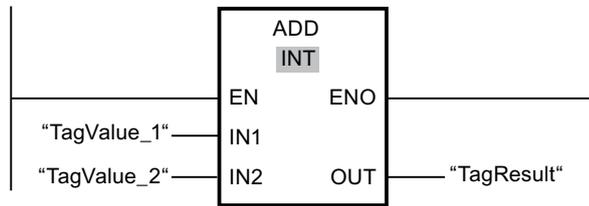
Parameter	Declaration	Data type	Description
IN1	Input	INT, DINT	First addend
IN2	Input	INT, DINT	Second addend
OUT	Output	INT, DINT	Total

You can select the data type of the instruction in the "<???">" drop-down list in the instruction box.

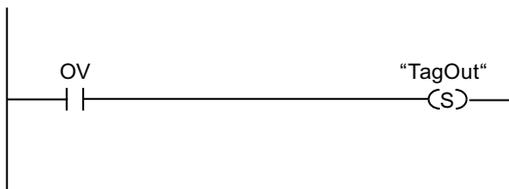
Example

The following example shows how the instruction works:

Network 1:



Network 2:



The "Add" instruction is always executed (regardless of the signal state at enable input EN).

The value of the "TagValue_1" operand is added to value of the TagValue_2 operand. The result of the addition is stored in the "TagResult" operand.

If an overflow occurs during execution of the "Add" instruction, the status bit OV is set to "1". In network 2, following the query of the status bit OV, the "Set output" (S) instruction is executed and the "TagOut" operand is set.

See also

---| |--- OV: Get status bit OV (STEP 7 Safety Advanced V11) (Page 354)

---| / |--- OV: Get negated status bit OV (STEP 7 Safety Advanced V11) (Page 355)

13.2.7.2 SUB: Subtract (STEP 7 Safety Advanced V11)

Description

You can use the "Subtract" instruction to subtract the value at input IN2 from the value at input IN1 and query the difference at the OUT output ($OUT = IN1 - IN2$).

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Note

If the result of the instruction is outside the range permitted for the data type, then the F-CPU goes to STOP when the result is sent to an output at an F-I/O or to a partner F-CPU via safety-related CPU-CPU communication.

One of the following diagnostic events is then

entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"
- "Safety program: internal CPU fault; internal error information: 404"

You must therefore ensure that the permitted range for the data type is observed when creating the program!

If this is not possible, you can avoid a STOP of the F-CPU by inserting a "Get status bit OV" instruction in the next network, thereby programming overflow detection into it.

Note the following:

- The result of the instruction behaves like the analogous instruction in a standard block.
 - The network with the "Get status bit OV" instruction must not contain any jump labels.
 - The execution time of the instruction is extended (see also Excel file for response time calculation (<http://support.automation.siemens.com/WWW/view/en/49368678/134200>)).
 - A warning is issued if you do not insert a "Get status bit OV" instruction.
-

Parameters

The following table shows the parameters of the instruction:

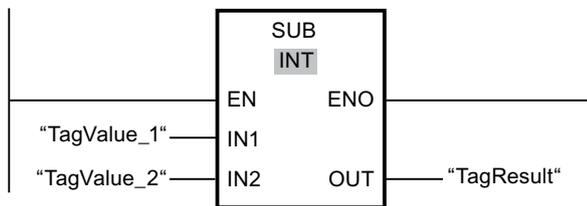
Parameter	Declaration	Data type	Description
IN1	Input	INT, DINT	Minuend
IN2	Input	INT, DINT	Subtrahend
OUT	Output	INT, DINT	Difference

You can select the data type of the instruction in the "<???">" drop-down list in the instruction box.

Example

The following example shows how the instruction works:

Network 1:



Network 2:



The "Subtract" instruction is always executed (regardless of the signal state at enable input EN).

The value of operand "TagValue_2" is subtracted from the value of operand "TagValue_1". The result of the addition is stored in operand "TagResult".

If an overflow occurs during execution of the "Subtract" instruction, the status bit OV is set to "1". In network 2, following the query of the status bit OV, the "Set output" (S) instruction is executed and the "TagOut" operand is set.

See also

---| |--- OV: Get status bit OV (STEP 7 Safety Advanced V11) (Page 354)

---| / |--- OV: Get negated status bit OV (STEP 7 Safety Advanced V11) (Page 355)

13.2.7.3 MUL: Multiply (STEP 7 Safety Advanced V11)

Description

You can use the "Multiply" instruction to multiply the value at input IN1 by the value at input IN2 and query the product at output OUT ($OUT = IN1 \times IN2$).

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Note

If the result of the instruction is outside the range permitted for the data type, then the F-CPU goes to STOP when the result is sent to an output at an F-I/O or to a partner F-CPU via safety-related CPU-CPU communication.

One of the following diagnostic events is then

entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"
- "Safety program: internal CPU fault; internal error information: 404"

You must therefore ensure that the permitted range for the data type is observed when creating the program!

If this is not possible, you can avoid a STOP of the F-CPU by inserting a "Get status bit OV" instruction in the next network, thereby programming overflow detection into it.

Note the following:

- The result of the instruction behaves like the analogous instruction in a standard block.
 - The network with the "Get status bit OV" instruction must not contain any jump labels.
 - The execution time of the instruction is extended (see also Excel file for response time calculation (<http://support.automation.siemens.com/WWW/view/en/49368678/134200>)).
 - A warning is issued if you do not insert a "Get status bit OV" instruction.
-

Parameters

The following table shows the parameters of the instruction:

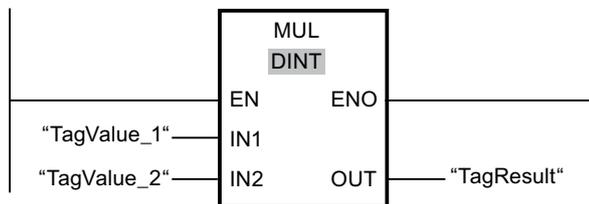
Parameter	Declaration	Data type	Description
IN1	Input	INT, DINT	Multiplier
IN2	Input	INT, DINT	Multiplicand
OUT	Output	INT, DINT	Product

You can select the data type of the instruction in the "<???">" drop-down list in the instruction box.

Example

The following example shows how the instruction works:

Network 1:



Network 2:



The "Multiply" instruction is always executed (regardless of the signal state at enable input EN).

The value of the "TagValue_1" operand is multiplied by the value of the "TagValue_2" operand. The result of the multiplication is stored in the "TagResult" operand.

If an overflow occurs during execution of the "Multiply" instruction, the status bit OV is set to "1". In network 2, following the query of the status bit OV, the "Set output" (S) instruction is executed and the "TagOut" operand is set.

See also

---| |--- OV: Get status bit OV (STEP 7 Safety Advanced V11) (Page 354)

---| / |--- OV: Get negated status bit OV (STEP 7 Safety Advanced V11) (Page 355)

13.2.7.4 DIV: Divide (STEP 7 Safety Advanced V11)

Description

You can use the "Divide" instruction to divide the value at input IN1 by the value at input IN2 and query the quotient at the OUT output ($OUT = IN1 / IN2$).

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Note

If the result of the instruction is outside the range permitted for the data type, then the F-CPU goes to STOP when the result is sent to an output at an F-I/O or to a partner F-CPU via safety-related CPU-CPU communication.

One of the following diagnostic events is then

entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"
- "Safety program: internal CPU fault; internal error information: 404"

You must therefore ensure that the permitted range for the data type is observed when creating the program!

If this is not possible, you can avoid a STOP of the F-CPU by inserting a "Get status bit OV" instruction in the next network, thereby programming overflow detection into it.

Note the following:

- The result of the instruction behaves like the analogous instruction in a standard block.
- The network with the "Get status bit OV" instruction must not contain any jump labels.
- The execution time of the instruction is extended (see also Excel file for response time calculation (<http://support.automation.siemens.com/WWW/view/en/49368678/134200>)).
- A warning is issued if you do not insert a "Get status bit OV" instruction.

Note

If the divisor (input IN2) of a DIV instruction = 0, the quotient of the division (result of division at output OUT) = 0. The result behaves like the corresponding instruction in a standard block. The F-CPU does *not* go to STOP mode. The behavior occurs regardless of whether a "Get status bit OV" instruction has been inserted in the next network.

Parameters

The following table shows the parameters of the instruction:

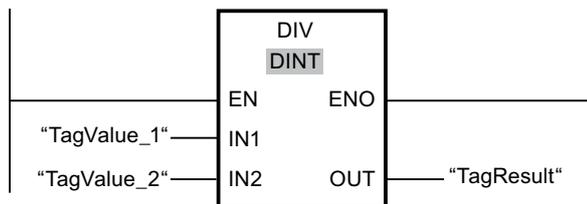
Parameter	Declaration	Data type	Description
IN1	Input	INT, DINT	Dividend
IN2	Input	INT, DINT	Divisor
OUT	Output	INT, DINT	Quotient

You can select the data type of the instruction in the "<???">" drop-down list in the instruction box.

Example

The following example shows how the instruction works:

Network 1:



Network 2:



The "Divide" instruction is always executed (regardless of the signal state at enable input EN).

The value of operand "TagValue_1" is divided by the value of operand "TagValue_2". The result of the division is stored in operand "TagResult".

If an overflow occurs during execution of the "Divide" instruction, the status bit OV is set to "1". In network 2, following the query of the status bit OV, the "Set output" (S) instruction is executed and the "TagOut" operand is set.

See also

---| |--- OV: Get status bit OV (STEP 7 Safety Advanced V11) (Page 354)

---| / |--- OV: Get negated status bit OV (STEP 7 Safety Advanced V11) (Page 355)

13.2.7.5 NEG: Create twos complement (STEP 7 Safety Advanced V11)

Description

You can use the "Create twos complement" instruction to change the sign of the value at input IN input and query the result at output OUT. If there is a positive value at input IN, for example, the negative equivalent of this value is sent to output OUT.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Note

If the result of the instruction is outside the range permitted for the data type, then the F-CPU goes to STOP when the result is sent to an output at an F-I/O or to a partner F-CPU via safety-related CPU-CPU communication.

One of the following diagnostic events is then

entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"
- "Safety program: internal CPU fault; internal error information: 404"

You must therefore ensure that the permitted range for the data type is observed when creating the program!

If this is not possible, you can avoid a STOP of the F-CPU by inserting a "Get status bit OV" instruction in the next network, thereby programming overflow detection into it.

Note the following:

- The result of the instruction behaves like the analogous instruction in a standard block.
 - The network with the "Get status bit OV" instruction must not contain any jump labels.
 - The execution time of the instruction is extended (see also Excel file for response time calculation (<http://support.automation.siemens.com/WWW/view/en/49368678/134200>)).
 - A warning is issued if you do not insert a "Get status bit OV" instruction.
-

Parameters

The following table shows the parameters of the instruction:

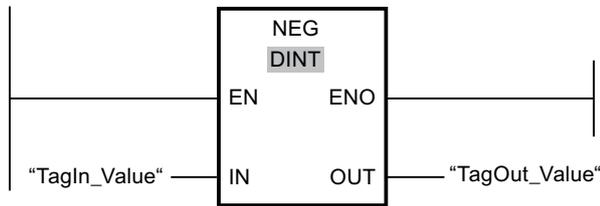
Parameter	Declaration	Data type	Description
IN	Input	INT, DINT	Input value
OUT	Output	INT, DINT	Twos complement of the input value

You can select the data type of the instruction in the "<???">" drop-down list in the instruction box.

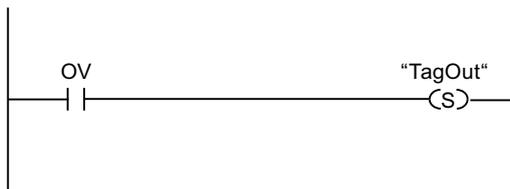
Example

The following example shows how the instruction works:

Network 1:



Network 2:



The "Create twos complement" instruction is always executed (regardless of the signal state at enable input EN).

The sign of the "TagIn_Value" operand is changed and the result is stored in the "TagOut_Value" operand.

If an overflow occurs during execution of the "Create twos complement" instruction, the status bit OV is set to "1". In network 2, following the query of the status bit OV, the "Set output" (S) instruction is executed and the "TagOut" operand is set.

See also

---| |--- OV: Get status bit OV (STEP 7 Safety Advanced V11) (Page 354)

---| / |--- OV: Get negated status bit OV (STEP 7 Safety Advanced V11) (Page 355)

13.2.8 Move operations

13.2.8.1 MOVE: Move value (STEP 7 Safety Advanced V11)

Description

You can use the "Move value" instruction to transfer the content of the operand at input IN to the operand at output OUT1.

Only identical operand widths can be specified for input IN and output OUT1.

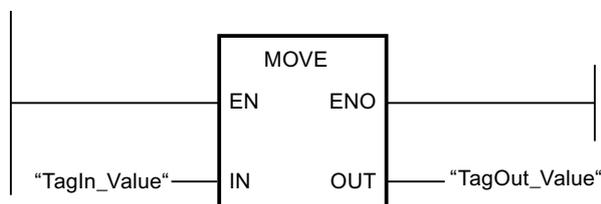
Enable input "EN" or enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Parameters

Parameter	Declaration	Data type	Description
IN	Input	INT, DINT, WORD, DWORD, TIME	Source value
OUT1	Output	INT, DINT, WORD, DWORD, TIME	Destination address

Example

The following example shows how the instruction works:



The instruction is always executed regardless of the signal state at enable input "EN". The instruction copies the content of operand "TagIn_Value" to operand "TagOut_Value".

13.2.8.2 WR_FDB: Write value indirectly to an F-DB (STEP 7 Safety Advanced V11)

Description

This instruction writes the value specified in input IN to the tag addressed by INI_ADDR and OFFSET in an F-DB.

The address of the tags addressed using INI_ADDR and OFFSET must be within the address range defined by addresses INI_ADDR and END_ADDR.

If the F-CPU has gone to STOP mode with diagnostic event ID 75E2, check to determine if this condition is satisfied.

The start address of the area in an F-DB to which the value at input IN is to be written is transferred using input INI_ADDR. The associated offset in this area is transferred using input OFFSET.

The addresses transferred in input INI_ADDR or END_ADDR must point to a tag of the selected data type in an F-DB. Only tags of the selected data type are permitted between the INI_ADDR and END_ADDR addresses. The INI_ADDR address must be smaller than the END_ADDR address.

As shown in the following example, the INI_ADDR and END_ADDR addresses must be transferred fully-qualified as "DBx".DBWy or in the corresponding symbolic representation. Transfers in other forms are not permitted.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN	Input	INT, DINT	Value to be written to the F-DB
INI_ADDR	Input	POINTER	Start address of the area in an F-DB
END_ADDR	Input	POINTER	End address of the area in an F-DB
OFFSET	Input	INT	Offset

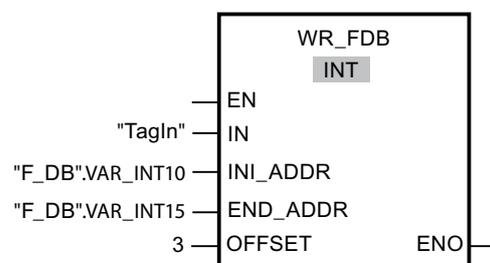
You can select the data type of the instruction in the "<???" drop-down list in the instruction box.

Examples of parameter assignment of INI_ADDR, END_ADDR, and OFFS

Name	Data type	Initial value	Comment
Static			
VAR_BOOL10	Bool	false	
VAR_BOOL11	Bool	false	
VAR_BOOL12	Bool	false	
VAR_BOOL13	Bool	false	
VAR_TIME10	Time	T#0MS	
VAR_TIME11	Time	T#0MS	
VAR_INT10	Int	0	<- INI_ADDR = "F-DB".VAR_INT10 Example 1
VAR_INT11	Int	0	
VAR_INT12	Int	0	
VAR_INT13	Int	0	<- OFFSET = 3
VAR_INT14	Int	0	
VAR_INT15	Int	0	<- END_ADDR = "F-DB".VAR_INT15
VAR_BOOL20	Bool	false	
VAR_BOOL21	Bool	false	
VAR_BOOL22	Bool	false	
VAR_BOOL23	Bool	false	
VAR_INT20	Int	0	<- INI_ADDR = "F-DB".VAR_INT20 Example 2
VAR_INT21	Int	0	
VAR_INT22	Int	0	
VAR_INT23	Int	0	<- INI_END = "F-DB".VAR_INT23
VAR_INT30	Int	0	<- INI_ADDR = "F-DB".VAR_INT30 Example 3
VAR_INT31	Int	0	<- OFFSET = 1
VAR_INT32	Int	0	
VAR_INT33	Int	0	
VAR_INT34	Int	0	<- END_ADDR = "F-DB".VAR_INT34
VAR_TIME20	TIME	T#0MS	
VAR_DINT10	DInt	0	<- INI_ADDR = "F-DB".VAR_DINT10 Example 4
VAR_DINT11	DInt	0	
VAR_DINT12	DInt	0	<- OFFSET = 2
VAR_DINT13	DInt	0	<- END_ADDR = "F-DB".VAR_DINT13

Example

The following example shows how the instruction works:



13.2.8.3 RD_FDB: Read value indirectly from an F-DB (STEP 7 Safety Advanced V11)

Description

This instruction reads the tag addressed via INI_ADDR and OFFSET in an F-DB and provides it at output OUT.

The address of the tags addressed using INI_ADDR and OFFSET must be within the address range defined by addresses INI_ADDR and END_ADDR.

If the F-CPU has gone to STOP mode with diagnostic event ID 75E2, check to determine if this condition is satisfied.

The start address of the area in an F-DB from which the tag is to be read is transferred using input INI_ADDR. The associated offset in this area is transferred using input OFFSET.

The addresses transferred in input INI_ADDR or END_ADDR must point to a tag of the selected data type in an F-DB. Only tags of the selected data type are permitted between the INI_ADDR and END_ADDR addresses. The INI_ADDR address must be smaller than the END_ADDR address.

The INI_ADDR and END_ADDR addresses must be transferred fully-qualified as "DBx".DBWy or in the corresponding symbolic representation. Transfers in other forms are not permitted. Examples of parameter assignment of INI_ADDR, END_ADDR, and OFFSET are contained in WR_FDB: Write value indirectly to an F-DB (STEP 7 Safety Advanced V11) (Page 328).

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Parameters

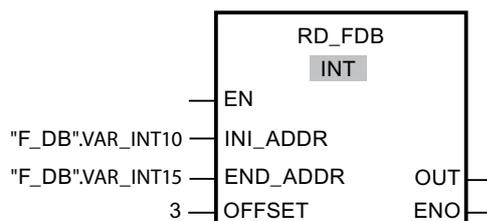
The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
INI_ADDR	Input	POINTER	Start address of the area in an F-DB
END_ADDR	Input	POINTER	End address of the area in an F-DB
OFFSET	Input	INT	Offset
OUT	Output	INT, DINT	Value to be read from the F-DB

You can select the data type of the instruction in the "<???">" drop-down list in the instruction box.

Example

The following example shows how the instruction works:



13.2.9 Conversion operations

13.2.9.1 CONVERT: Convert value (STEP 7 Safety Advanced V11)

Description

The "Convert value" instruction reads the content of parameter IN and converts it according to the data types selected in the instruction box. The converted value is output at output OUT.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

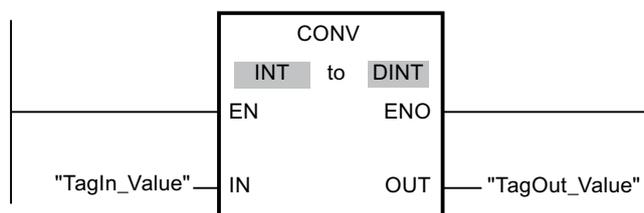
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN	Input	INT	Value to be converted.
OUT	Output	DINT	Result of the conversion

Example

The following example shows how the instruction works:



The instruction is always executed regardless of the signal state at enable input EN. The content of the operand "TagIn_Value" is read and converted to an integer (32 bit). The result is stored in operand "TagOut_Value".

13.2.9.2 BO_W: Convert 16 data elements of data type BOOL to a data element of data type WORD (STEP 7 Safety Advanced V11)

Description

This instruction converts the 16 values of data type BOOL at inputs IN0 to IN15 to a value of data type WORD, which is made available at output OUT. The conversion takes place as follows: The *i*th bit of the WORD value is set to 0 (or 1), if the value at input IN_{*i*} = 0 (or 1).

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

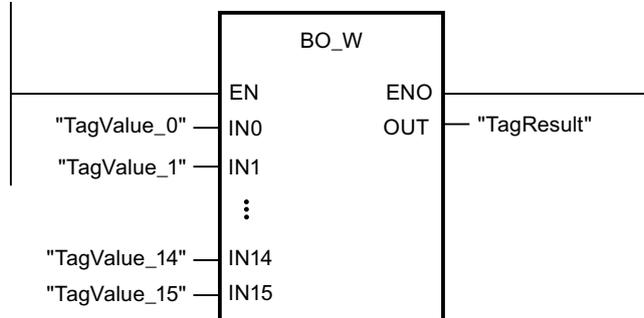
Note: To supply inputs IN0 to IN15 with Boolean constants "0" and "1", you can access tags "VKE0" and "VKE1" in the F-shared DB using a fully-qualified DB access ("F_GLOBDB".VKE0 or "F_GLOBDB".VKE1).

Parameters

Parameter	Declaration	Data type	Description
IN0	Input	BOOL	Bit 0 of WORD value
IN1	Input	BOOL	Bit 1 of WORD value
...			...
IN15	Input	BOOL	Bit 15 of WORD value
OUT	Output	WORD	WORD value consisting of IN0 to IN15

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN0	TagValue_0	FALSE
IN1	TagValue_1	FALSE
...		...
IN13	TagValue_13	FALSE
IN14	TagValue_14	TRUE
IN15	TagValue_15	TRUE
OUT	TagResult	W#16#0003

The values of operands "TagValue_0" to " TagValue_15" are combined to form data type WORD and assigned to operand "TagResult".

13.2.9.3 W_BO: Convert a data element of data type WORD to 16 data elements of data type BOOL (STEP 7 Safety Advanced V11)

Description

This instruction converts the value of data type WORD at input IN to 16 values of data type BOOL, which are provided at outputs OUT0 to OUT15. The conversion takes place as follows: Output OUT_i is set to 0 (or 1), if the ith bit of the WORD value is 0 (or 1).

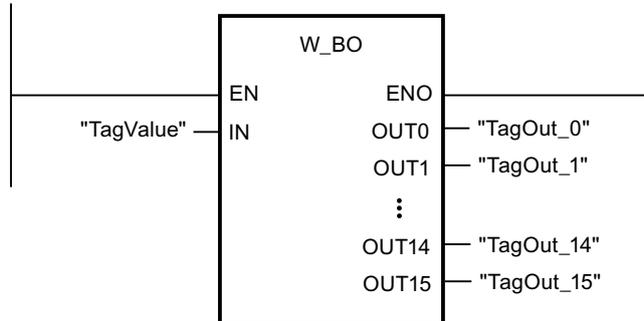
Enable input "EN" or enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Parameters

Parameter	Declaration	Data type	Description
IN	Input	WORD	WORD value
OUT0	Output	BOOL	Bit 0 of WORD value
OUT1	Output	BOOL	Bit 1 of WORD value
...			...
OUT15	Output	BOOL	Bit 15 of WORD value

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagValue	W#16#0003
OUT0	TagOUT_0	FALSE
OUT1	TagOUT_1	FALSE
...		...
OUT13	TagOUT_13	FALSE
OUT14	TagOUT_14	TRUE
OUT15	TagOUT_15	TRUE

The value of operand "TagValue" of data type WORD is converted to the 16 values "TagOUT_0" to "TagOUT_15" of data type BOOL.

13.2.9.4 SCALE: Scale values (STEP 7 Safety Advanced V11)

Description

This instruction scales the value at input IN in physical units between the low limit value at input LO_LIM and the high limit value at input HI_LIM. It is assumed that the value at input IN is between 0 and 27648. The scaling result is provided at output OUT.

The instruction uses the following equation:

$$\text{OUT} = [\text{IN} \times (\text{HI_LIM} - \text{LO_LIM})] / 27648 + \text{LO_LIM}$$

As long as the value at input IN is greater than 27648, output OUT is linked to HI_LIM and OUT_HI is set to 1.

As long as the value at input IN is less than 0, output OUT is linked to LO_LIM and OUT_LO is set to 1.

For inverse scaling, you must assign LO_LIM > HI_LIM. With inverse scaling, the output value at output OUT decreases while the input value at input IN increases.

Every call of the "Scale values" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., SCALE_DB_1) or a multi-instance (e.g., SCALE_Instance_1) for the "Scale values" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Parameters

Parameter	Declaration	Data type	Description
IN	Input	INT	Input value to be scaled in physical units
HI_LIM	Input	INT	High limit value of value range of OUT
LO_LIM	Input	INT	Low limit value of value range of OUT
OUT	Output	INT	Result of scaling
OUT_HI	Output	BOOL	1 = Input value > 27648: OUT = HI_LIM
OUT_LO	Output	BOOL	1 = Input value < 0: OUT = LO_LIM

Behavior in the event of overflow or underflow of analog values and fail-safe value output

Note

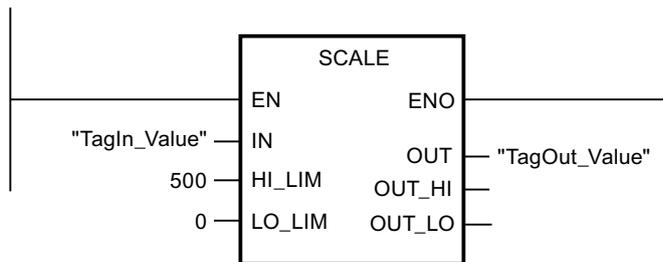
If inputs from the PII of an SM 336; AI 6 x 13Bit or SM 336; F-AI 6 x 0/4 ... 20 mA HART are used as input values, note that the F-system detects an overflow or underflow of a channel of this F-SM as an F-I/O fault or channel fault. The fail-safe value 0 is provided in place of 7FFF_H (for overflow) or 8000_H (for underflow) in the PII for the safety program.

If other fail-safe values are to be output in this case, you must evaluate the QBAD tag in the F-I/O DB (branch to output of an individual fail-safe value).

If the value in the PII of the F-SM is within the overrange or underrange, but is > 27648 or < 0, you can likewise branch to the output of an individual fail-safe value by evaluating outputs OUT_HI and OUT_LO, respectively.

Example

The following example shows how the instruction works:



When operand "TagIn_Value" = 20000, the result is "TagOut_Value" 361.

13.2.10 Program control operations

13.2.10.1 ---(JMP): Jump if RLO = 1 (STEP 7 Safety Advanced V11)

Description

You can use the "Jump if RLO = 1" instruction to interrupt the linear execution of the program and resume it in another network. The destination network must be identified by a jump label (Page 340) (LABEL). The description of the jump label is specified in the placeholder above the instruction.

The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block.

If the result of logic operation (RLO) at the input of the instruction is "1" or the input is not connected, the jump to the network identified by the jump label is executed. The jump direction can be towards higher or lower network numbers.

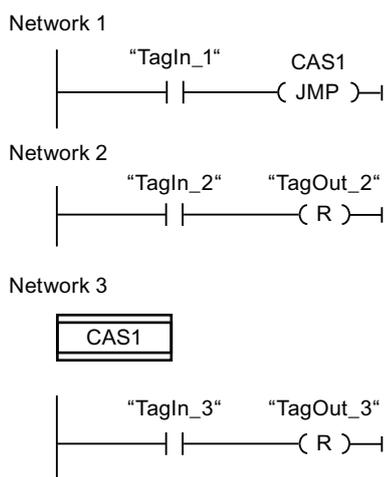
If the result of logic operation (RLO) at the input of the instruction is "0", the program continues executing in the next network.

Note

You are not permitted to program a SENDDP or SENDS7 call between a jump instruction and the associated destination of the jump instruction.

Example

The following example shows how the instruction works:



When operand "TagIn_1" has signal state "1", the "Jump if RLO = 1" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. When input "TagIn_3" has signal state "1", output "TagOut_3" is reset.

13.2.10.3 LABEL: Jump label (STEP 7 Safety Advanced V11)

Description

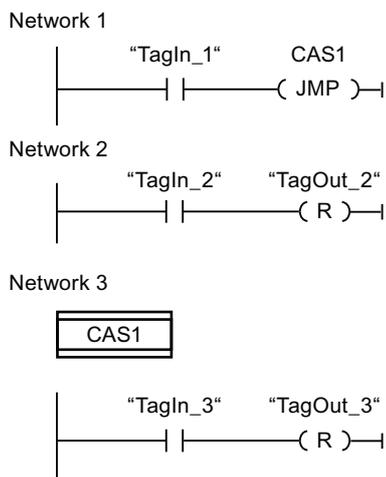
You can use a jump label to specify a destination network, in which the program execution should resume after a jump.

The jump label and the instruction in which the jump label is specified must be located in the same block. The name of a jump label can only be assigned once in a block.

Only one jump label can be placed in a network. To each jump label can be jumped from several locations.

Example

The following example shows how the instruction works:



When operand "TagIn_1" has signal state "1", the "Jump if RLO = 1" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. When input "TagIn_3" has signal state "1", output "TagOut_3" is reset.

See also

---(JMP): Jump if RLO = 1 (STEP 7 Safety Advanced V11) (Page 338)

---(JMPN): Jump if RLO = 0 (STEP 7 Safety Advanced V11) (Page 339)

--(RET): Return (STEP 7 Safety Advanced V11) (Page 341)

13.2.10.4 --(RET): Return (STEP 7 Safety Advanced V11)

Description

You can use the "Return" operation to stop the processing of a block.

If the result of logic operation (RLO) at the input of the "Return" instruction is "1", program execution is terminated in the currently called block and continued in the calling block (for example, in the main safety block) after the call function. If the RLO at the input of the "Return" instruction is "0", the instruction is not executed. Program execution continues in the next network of the called block.

Influencing the status of the call function (ENO) is irrelevant, because the enable output "ENO" cannot be connected.

Note

You must not assign any RET instruction in the Main Safety Block.

Example

The following example shows how the instruction works:



When the "TagIn" operand delivers signal state "1", the "Return" instruction is executed. Program execution is terminated in the called block and continues in the calling block.

See also

---(JMP): Jump if RLO = 1 (STEP 7 Safety Advanced V11) (Page 338)

---(JMPN): Jump if RLO = 0 (STEP 7 Safety Advanced V11) (Page 339)

LABEL: Jump label (STEP 7 Safety Advanced V11) (Page 340)

13.2.10.5 ---(OPN): Open global data block (STEP 7 Safety Advanced V11)

Description

You can use the "Open global data block" instruction to open a data block. The number of the data block is transferred to the DB register. Subsequent DB commands access the relevant blocks depending on the register contents.

Note

Note when using the "Open global data block" instruction that the content of the DB register can be changed following calls of F-FB/F-FC and "fully qualified DB accesses," such that there is no guarantee that the last data block you opened with "Open global data block" is still open.

You should therefore use the following method for addressing data to avoid errors when accessing data of the DB register:

- Use symbolic addressing.
- Use only fully qualified DB accesses.

If you still want to use the "Open global data block" instruction, you must ensure that the DB register is restored by repeating the "Open global data block" instruction following calls of F-FB/F-FC and "fully qualified DB accesses." Otherwise, a malfunction could result.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Data block>	Input	BLOCK_DB	Data block that is opened

"Fully qualified DB access"

The initial access to data of a data block in an F-FB/F-FC must always be a "fully qualified DB access," or it must be preceded by the "Open global data block" instruction. This also applies to the initial access to data of a data block after a jump label.

An example of "fully-qualified DB access" and "non-fully-qualified DB access" can be found under Restrictions in the programming languages FBD/LAD (Page 59).

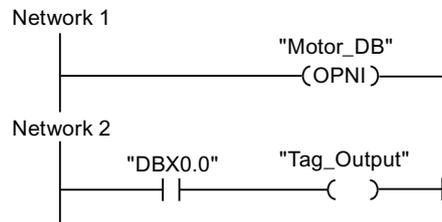
Access to instance DBs

You can also access instance DBs of F-FBs with fully qualified access, e.g., for transfer of block parameters. It is not possible to access static data in instance DBs of other F-FBs.

Note that accessing instance DBs of F-FBs that are not called in the safety program can cause the F-CPU to go to STOP mode.

Example

The following example shows how the instruction works:



The "Motor_DB" data block is called in network 1. The number of the data block is transferred to the DB register. The "DBX0.0" operand is queried in network 2. The signal state of the "DBX0.0" operand is assigned to the "Tag_Output" operand.

13.2.11 Word logic operations

13.2.11.1 AND: AND logic operation (STEP 7 Safety Advanced V11)

Description

You can use the "AND logic operation" instruction to combine the value at input IN1 to the value at input IN2 bit-by-bit by AND logic and query the result at output OUT.

When the instruction is executed, bit 0 of the value at input IN1 and bit 0 of the value at input IN2 are ANDed. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified values.

The result bit has signal state "1" only when both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has signal state "0", the corresponding result bit is reset.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

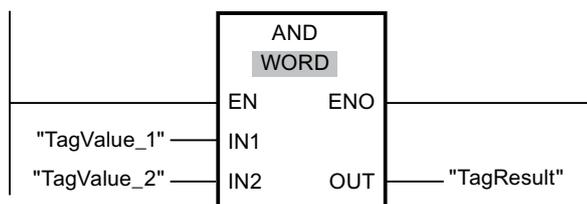
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN1	Input	WORD	First value of logic operation
IN2	Input	WORD	Second value of logic operation
OUT	Output	WORD	Result of the instruction

Example

The following example shows how the instruction works:



IN1	"TagValue_1" = 01010101 01010101
IN2	"TagValue_2" = 00000000 00001111
OUT	"TagResult" = 00000000 00000101

The instruction is always executed regardless of the signal state at enable input "EN". The value of the "TagValue_1" operand and the value of the "TagValue_2" operand are ANDed. The result is mapped bit-by-bit and output in the "TagResult" operand.

13.2.11.2 OR: OR logic operation (STEP 7 Safety Advanced V11)

Description

You can use the "OR logic operation" instruction to connect the value at input IN1 input to the value at input IN2 bit-by-bit by OR logic and query the result at output OR.

When the instruction is executed, bit 0 of the value at input IN1 and bit 0 of the value at input IN2 are ORed. The result is stored in bit 0 of output OUT. The same logic operation is executed for all bits of the specified tags.

The result bit has signal state "1" when at least one of the two bits in the logic operation has signal state "1". If both of the bits of the logic operation have signal state "0", the corresponding result bit is reset.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

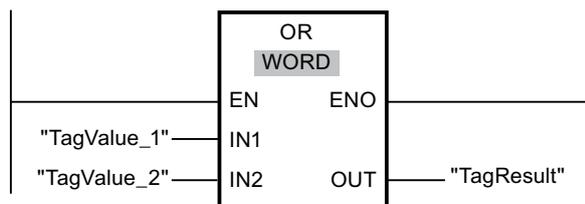
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN1	Input	WORD	First value of logic operation
IN2	Input	WORD	Second value of logic operation
OUT	Output	WORD	Result of the instruction

Example

The following example shows how the instruction works:



IN1	"TagValue_1" = 01010101 01010101
IN2	"TagValue_2" = 00000000 00001111
OUT	"TagResult" = 01010101 01011111

The instruction is always executed regardless of the signal state at enable input "EN". The value of the "TagValue_1" operand and the value of the "TagValue_2" operand are ORed. The result is mapped bit-by-bit and output in the "TagResult" operand.

13.2.11.3 XOR: EXCLUSIVE OR logic operation (STEP 7 Safety Advanced V11)

Description

You can use the "EXCLUSIVE OR logic operation" instruction to combine the value at input IN1 and the value at input IN2 bit-by-bit by EXCLUSIVE OR logic and query the result at output OUT.

When the instruction is executed, bit 0 of the value at input IN1 input and bit 0 of the value at input IN2 are logically combined by EXCLUSIVE OR. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified value.

The result bit has signal state "1" when one of the two bits in the logic operation has signal state "1". If both of the bits of the logic operation have signal state "1" or "0", the corresponding result bit is reset.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

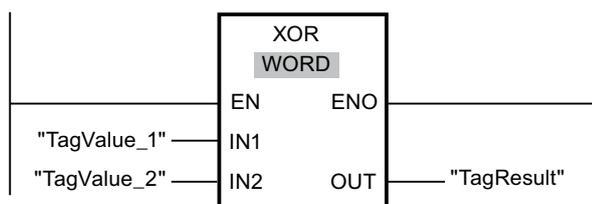
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN1	Input	WORD	First value of logic operation
IN2	Input	WORD	Second value of logic operation
OUT	Output	WORD	Result of the instruction

Example

The following example shows how the instruction works:



IN1	"TagValue_1" = 01010101 01010101
IN2	"TagValue_2" = 00000000 00001111
OUT	"TagResult" = 01010101 01011010

The instruction is always executed regardless of the signal state at enable input "EN". The value of the "TagValue_1" operand and the value of the "TagValue_2" operand are logically combined by EXCLUSIVE OR. The result is mapped bit-by-bit and output in the "TagResult" operand.

13.2.12 Shift and rotate

13.2.12.1 SHR: Shift right (STEP 7 Safety Advanced V11)

Description

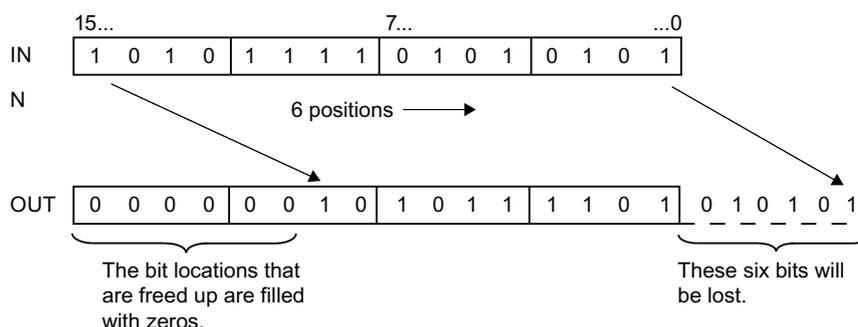
You can use the "Shift right" instruction to move the content of the operand at input IN bit-by-bit to the right and query the result at output OUT. You use parameter N (low-byte) to specify the number of bit positions by which the specified value is moved.

When the value at parameter N (low-byte) is "0", the value at input IN is copied into the operand at output OUT.

When the value at parameter N (low-byte) is greater than the number of available bit positions, the operand value at input IN is moved by the available number of bit positions to the right.

The bit locations that are freed up in the left area of the operand during the shift operation are filled with zeros.

The following figure shows how the content of an operand of data type WORD is moved by 6 bit positions to the right:



Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

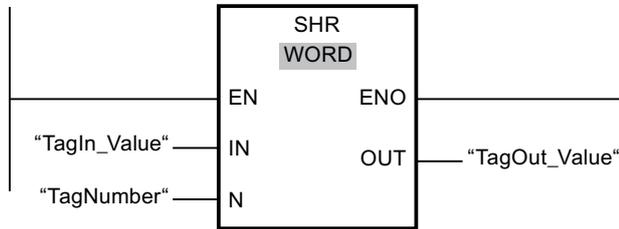
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN	Input	WORD	Value that is moved
N	Input	INT	Number of bit positions by which the value is moved
OUT	Output	WORD	Result of the instruction

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	TagNumber	3
OUT	TagOut_Value	0000 0111 1111 0101

The instruction is always executed regardless of the signal state at enable input "EN". The content of the operand "TagIn_Value" is moved three bit positions to the right. The result is output at output "TagOut_Value".

13.2.12.2 SHL: Shift left (STEP 7 Safety Advanced V11)

Description

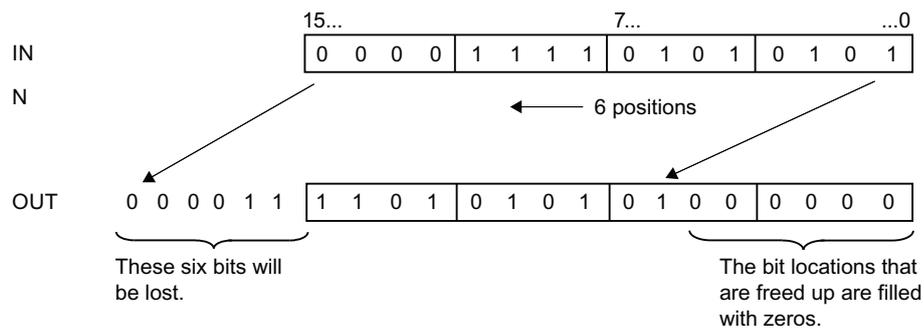
You can use the "Shift left" instruction to move the content of the operand at input IN bit-by-bit to the left and query the result at output OUT. You use parameter N (low-byte) to specify the number of bit positions by which the specified value is moved.

When the value at parameter N (low-byte) is "0", the value at input IN is copied into the operand at output OUT.

When the value at parameter N (low-byte) is greater than the number of available bit positions, the operand value at input IN is moved by the available number of bit positions to the left.

The bit positions that are freed up in the right area of the operand during the shift operation are filled with zeros.

The following figure shows how the content of an operand of data type WORD is moved by 6 bit positions to the left:



Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

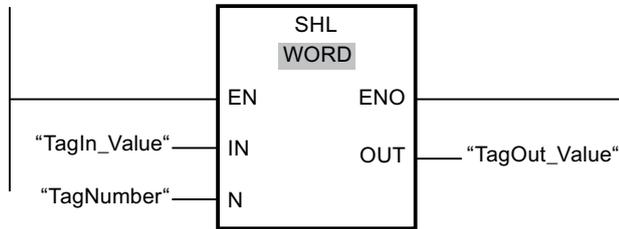
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN	Input	WORD	Value that is moved
N	Input	INT	Number of bit positions by which the value is moved
OUT	Output	WORD	Result of the instruction

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	TagNumber	4
OUT	TagOut_Value	1111 1010 1111 0000

The instruction is always executed regardless of the signal state at enable input "EN". The content of the operand "TagIn_Value" is moved four bit positions to the left. The result is output at output "TagOut_Value".

13.2.13 Operating

13.2.13.1 ACK_OP: Fail-safe acknowledgment (STEP 7 Safety Advanced V11)

Description

This instruction enables fail-safe acknowledgment from an operator control and monitoring system. It allows, for example, reintegration of F-I/O to be controlled from the operator control and monitoring system. Acknowledgment takes place in two steps:

- In/out parameter IN changes to a value of 6.
- In/out parameter IN changes to a value of 9 within 1 minute.

Once the in/out parameter IN has changed to a value of 6, the instruction evaluates whether this parameter has changed to a value of 9 after 1 second, at the earliest, or 1 minute, at the latest. Output OUT (output for acknowledgment) is then set to 1 for one cycle.

If an invalid value is input or if in/out parameter IN has not changed to 9 within 1 minute or the change occurred before 1 second has elapsed, then in/out parameter IN is reset to 0, and both steps listed above must be repeated.

During the time in which in/out parameter IN must change from 6 to 9, output Q is set to 1. Otherwise, Q has a value of 0.

Every call of the "Fail-safe acknowledgment" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., ACK_OP_DB_1) or a multi-instance (e.g., ACK_OP_Instance_1) for the "Fail-safe acknowledgment" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Note

A separate data area must be used for each call of ACK_OP. Each call can be processed only once in an F-runtime group cycle.

The F-CPU can go to STOP mode if the information above is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
 - "Data corruption in the safety program prior to output to partner F-CPU"
 - "Safety program: internal CPU fault; internal error information: 404"
-

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

 **WARNING**

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction")
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision. (S034)

Note

You can read out output Q by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program.

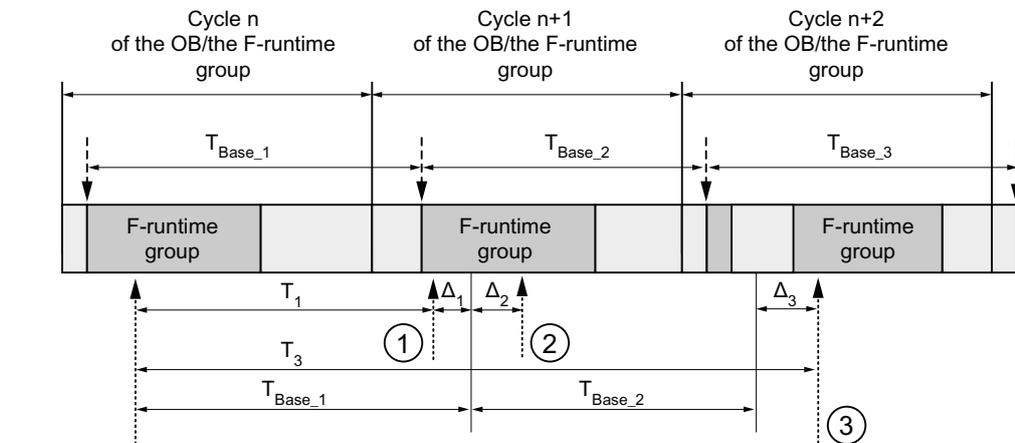
You can supply in/out parameter IN with just a memory word or nothing at all. In the safety program, read and write access to in/out parameter IN in the associated instance DB is not permitted!

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN	InOut	INT	Input variable from operator control and monitoring system
OUT	Output	BOOL	Output for acknowledgment
Q	Output	BOOL	Time status

Timing imprecision resulting from the update time of the time base used in the instruction:



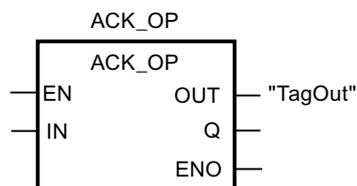
-----> = Time base update

.....> = Call time of an instruction with time processing

- ① For the first call in cycle $n+1$, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle $n+1$ are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle $n+1$. This does not involve another time update (by Δ_2).
- ③ For the call in cycle $n+2$, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle $n+2$. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle $n+1$.

Example

The following example shows how the instruction works:



See also

Implementing User Acknowledgment in the Safety Program of the F-CPU of a DP Master or IO Controller (Page 99)

Implementing user acknowledgment in the safety program of the F-CPU of a I-slave or I-device (Page 102)

13.2.14 Additional instructions

13.2.14.1 --| |-- OV: Get status bit OV (STEP 7 Safety Advanced V11)

Description

You can use the "Get status bit OV" instruction to detect whether a number range overflow occurred in the last arithmetic instruction processed. The "Get status bit OV" instruction functions like a normally open contact. If the query is fulfilled, the instruction has signal state "1". If the condition is not fulfilled the instruction has signal state "0".

The "Get status bit OV" evaluation must be inserted in the network that follows the instruction that influences the OV. This network must not contain any jump labels.

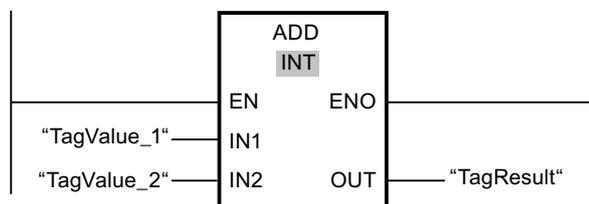
Note

The execution time of the OV-affecting instruction is extended when the "Get status bit OV" instruction is used (see also Excel file for response time calculation (<http://support.automation.siemens.com/WW/view/en/49368678/134200>)).

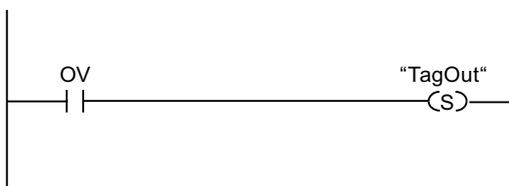
Example

The following example shows how the instruction works:

Network 1:



Network 2:



The "Add" instruction is always executed (regardless of the signal state at enable input EN).

The value of the "TagValue_1" operand is added to value of the TagValue_2 operand. The result of the addition is stored in the "TagResult" operand.

If an overflow occurs during execution of the "Add" instruction, the status bit OV is set to "1". In network 2, following the query of the status bit OV, the "Set output" (S) instruction is executed and the "TagOut" operand is set.

13.2.14.2 ---| / |--- OV: Get negated status bit OV (STEP 7 Safety Advanced V11)

Description

You can use the "Get negated status bit OV" instruction to detect whether a number range overflow occurred in the last arithmetic instruction processed. The "Get negated status bit OV" instruction functions like a normally closed contact. If the query is satisfied, the instruction has signal state "0". If the query not satisfied, the instruction has signal state "1".

The "Get negated status bit OV" evaluation must be inserted in the network following the instruction that influences the OV. This network must not contain any jump labels.

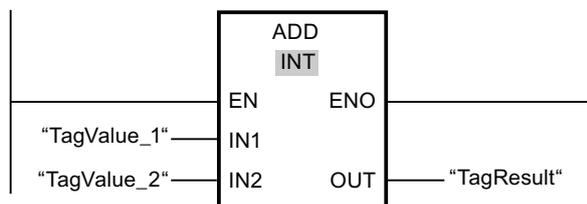
Note

The execution time of the OV-affecting instruction is extended when the "Get negated status bit OV" instruction is used (see also Excel file for response time calculation (<http://support.automation.siemens.com/WW/view/en/49368678/134200>)).

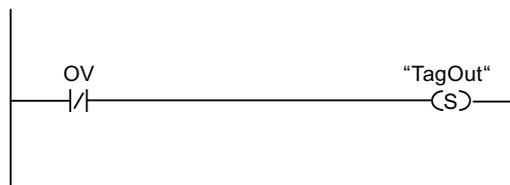
Example

The following example shows how the instruction works:

Network 1:



Network 2:



The "Add" instruction is always executed (regardless of the signal state at enable input EN).

The value of the "TagValue_1" operand is added to value of the TagValue_2 operand. The result of the addition is stored in the "TagResult" operand.

If an overflow does *not* occur during execution of the "Add" instruction, the status bit OV is reset to "0". In network 2, following the query of the status bit OV, the "Set output" (S) instruction is executed and the "TagOut" operand is set.

13.2.15 Communication

13.2.15.1 PROFIBUS/PROFINET

SENDDP and RCVDP: Send and receive data via PROFIBUS DP/PROFINET IO (STEP 7 Safety Advanced V11)

Introduction

You use the SENDDP and RCVDP instructions for fail-safe sending and receiving of data using:

- Safety-related master-master communication
- Safety-related master-master communication for S7 Distributed Safety
- Safety-related master-I-slave communication
- Safety-related I-slave-I-slave communication
- Safety-related IO controller-IO controller communication
- Safety-related IO Controller-IO Controller communication for S7 Distributed Safety
- Safety-related IO controller-I-Device communication
- Safety-related IO controller-I-Slave communication

Description

The SENDDP instruction sends 16 data elements of data type BOOL and 2 data elements of data type INT in a fail-safe manner to another F-CPU via PROFIBUS DP/PROFINET IO. The data can be received there by the related RCVDP instruction.

Every call of this instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., RCVDP_DB_1) or a multi-instance (e.g., RCVDP_Instance_1) for this instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

With the SENDDP instruction, the data to be sent (for example, outputs of other F-blocks/instructions) are available at input SD_BO_xx or SD_I_xx.

With the RCVDP instruction, the data received are available at output RD_BO_xx or RD_I_xx for additional processing by other F-blocks/instructions.

The operating mode of the F-CPU with the SENDDP instruction is provided at output SENDMODE. If the F-CPU with the SENDDP instruction is in deactivated safety mode, output SENDMODE = 1.

Communication between F-CPU's takes place in the background by means of a special safety protocol. You must define the communication relationship between a SENDDP instruction in one F-CPU and a RCVDP instruction in the other F-CPU by specifying an address relationship at the DP_DP_ID inputs of the SENDDP and RCVDP instructions. Associated SENDDP and RCVDP instructions are assigned the same value for DP_DP_ID.

⚠ WARNING

The value for each address relationship (input parameter DP_DP_ID; data type: INT) is user-defined; however, it must be unique from all other safety-related communication connections in the network. The uniqueness must be checked in the print-out of the safety program during acceptance testing of the safety program. Additional information can be found in Correctness of the communication configuration (Page 208).

You must supply inputs DP_DP_ID and LADDR with constant values when calling the instruction. Direct read or write access to the associated instance DB is not permitted in the safety program! (S016)

Note

Within a safety program, you must assign a different start address for every call of the SENDDP and RCVDP instructions at input LADDR. A separate instance DB must be used for each call of the SENDDP and RCVDP instructions. You must not declare and call these instructions as multi-instances.

The input and output parameters of the RCVDP instruction must not be supplied with temporary or static local data of the main safety block.

The input parameters of the RCVDP instruction must not be initialized with output parameters (using fully qualified DB accesses) of a RCVDP or RCVS7 instruction called in a preceding network.

You must not use an actual parameter for an output parameter of a RCVDP, if it is already being used for an input parameter of the same or another RCVDP or RCVS7 instruction. The F-CPU can go to STOP if this is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"
- "Safety program: internal CPU fault; internal error information: 404"

Note

You must not program any SENDDP instruction between a JMP or JMPN instruction and the associated destination network of the JMP or JMPN instruction.

You must not program any RET instruction before a SENDDP instruction.

Parameters of the SENDDP instruction

Parameter	Declaration	Data type	Description
SD_BO_00	Input	BOOL	Send data BOOL 00
...			...
SD_BO_15	Input	BOOL	Send data BOOL 15
SD_I_00	Input	INT	Send data INT 00
SD_I_01	Input	INT	Send data INT 01
DP_DP_ID	Input	INT	Network-wide unique value for the address relationship between a SENDDP and RCVDP instruction
TIMEOUT	Input	TIME	Monitoring time in ms for safety-related communication (see also Monitoring and response times (Page 523))
LADDR	Input	INT	Start address of address area/transfer area: <ul style="list-style-type: none"> • Of DP/DP coupler for safety-related master-master communication • For safety-related master-I-slave communication • For safety-related I-slave-I-slave communication • Of PN/PN coupler for safety-related IO Controller-IO Controller communication • For safety-related IO Controller-I-Device communication • For safety-related IO Controller-I-slave communication
ERROR	Output	BOOL	1=Communication error
SUBS_ON	Output	BOOL	1=RCVDP outputs fail-safe values
RET_DPRD	Output	WORD	Error code RET_VAL of the DPRD_DAT instruction (for a description of error codes, refer to the online help for DPRD_DAT instruction ("Extended instructions > Distributed I/O > Other".))
RET_DPWR	Output	WORD	Error code RET_VAL of the DPWR_DAT instruction (for a description of error codes, refer to the online help for DPWR_DAT instruction ("Extended instructions > Distributed I/O > Other".))
DIAG	Output	BYTE	Service information

Parameters of the RCVDP instruction

Parameter	Declaration	Data type	Description
ACK_REI	Input	BOOL	1=Acknowledgment for reintegration of send data following communication error
SUBBO_00	Input	BOOL	Fail-safe value for receive data BOOL 00
...			...
SUBBO_15	Input	BOOL	Fail-safe value for receive data BOOL 15
SUBI_00	Input	INT	Fail-safe value for receive data INT 00
SUBI_01	Input	INT	Fail-safe value for receive data INT 01
DP_DP_ID	Input	INT	Network-wide unique value for the address relationship between a SENDDP and RCVDP instruction
TIMEOUT	Input	TIME	Monitoring time in ms for safety-related communication (see also Monitoring and response times (Page 523))
LADDR	Input	INT	Start address of address area/transfer area: <ul style="list-style-type: none"> • Of DP/DP coupler for safety-related master-master communication • For safety-related master-I-slave communication • For safety-related I-slave-I-slave communication • Of PN/PN coupler for safety-related IO Controller-IO Controller communication • For safety-related IO Controller-I-Device communication • For safety-related IO Controller-I-slave communication
ERROR	Output	BOOL	1=Communication error
SUBS_ON	Output	BOOL	1=Fail-safe values are output
ACK_REQ	Output	BOOL	1=Acknowledgment for reintegration of send data required
SENDMODE	Output	BOOL	1=F-CPU with SENDDP instruction in deactivated safety mode
RD_BO_00	Output	BOOL	Receive data BOOL 00
...			...
RD_BO_15	Output	BOOL	Receive data BOOL 15
RD_I_00	Output	INT	Receive data INT 00
RD_I_01	Output	INT	Receive data INT 01
RET_DPRD	Output	WORD	Error code RET_VAL of the DPRD_DAT instruction (for a description of error codes, refer to the online help for DPRD_DAT instruction ("Extended instructions > Distributed I/O > Other".))
RET_DPWR	Output	WORD	Error code RET_VAL of the DPWR_DAT instruction (for a description of error codes, refer to the online help for DPWR_DAT instruction ("Extended instructions > Distributed I/O > Other".))
DIAG	Output	BYTE	Service information

Placement

The RCVDP instruction must be inserted at the start of the main safety block and the SENDDP instruction at the end.

Startup characteristics

After the sending and receiving F-systems are started up, communication must be established between the connection partners for the first time (SENDDP and RCVDP instructions). During this time, the receiver (RCVDP instruction) outputs the fail-safe values present at its inputs SUBBO_xx and SUBBI_xx.

The SENDDP and RCVDP instructions signal this at output SUBS_ON with 1. Output SENDMODE has a default of 0 and is not updated, as long as output SUBS_ON = 1.

Behavior in the event of communication errors

If a communication error occurs, for example, due to a signature error (CRC) or when monitoring time TIMEOUT expires, outputs ERROR and SUBS_ON = 1 are set. The receiver (RCVDP instruction) then outputs the fail-safe values assigned at its SUBBO_xx inputs. Output SENDMODE is not updated while output SUBS_ON = 1.

The send data of the SENDDP instruction present at inputs SD_BO_xx and SUBI_xx are only output again when communication errors are no longer detected (ACK_REQ = 1) and you acknowledge (Page 99) the RCVDP instruction with a positive edge at input ACK_REI.



WARNING

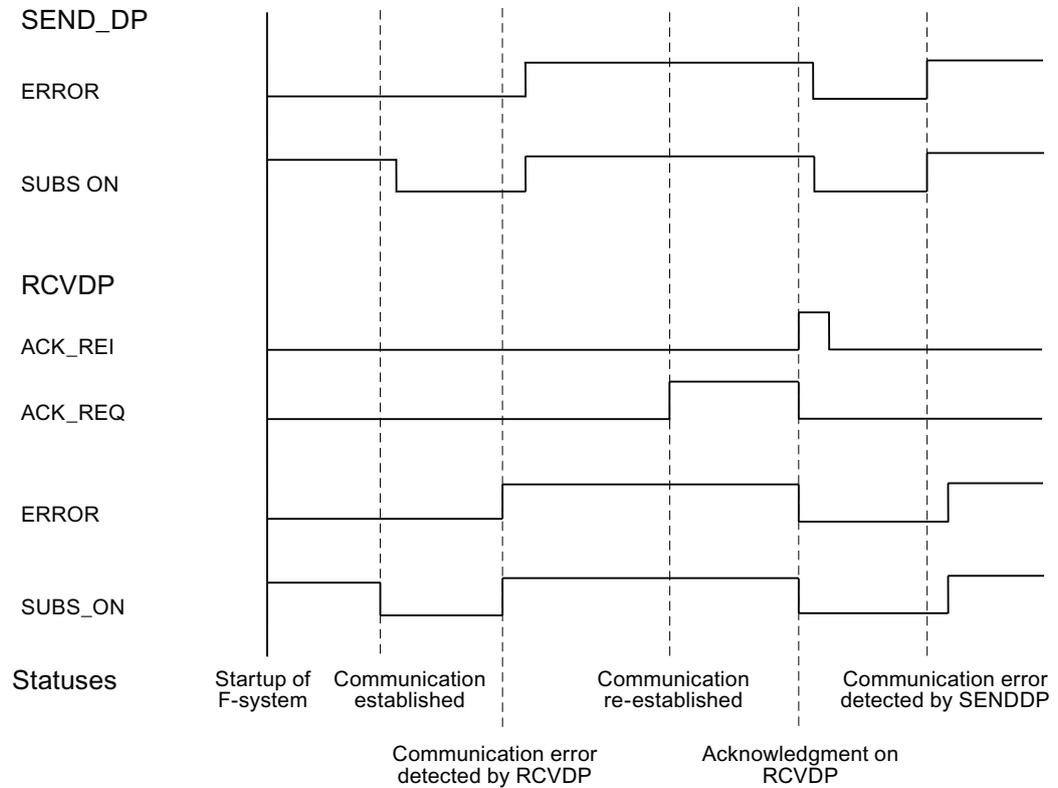
For the user acknowledgment, you must interconnect input ACK_REI with a signal generated by the operator input.

An interconnection with an automatically generated signal is not permitted. (S040)

Note that output ERROR (1=communication error) for a communication error will not be set unless communication between the connection partners (SENDDP and RCVDP instructions) has been previously established. If communication cannot be established after startup of the sending and receiving F-systems, check the configuration of the safety-related CPU-CPU communication, the parameter assignment of the SENDDP and RCVDP instructions, and the bus connection. You also obtain information on possible error causes by evaluating outputs RET_DPRD and RETDP_WR.

In general, always evaluate RET_DPRD and RETDP_WR, since it is possible that only one of the two outputs will contain error information.

Timing diagrams SENDDP/RCVDP



Output DIAG

In addition, non-fail-safe information about the type of communication errors that occurred is provided at output DIAG of the SENDDP and RCVDP instructions for service purposes.

You can read out this information by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program. DIAG bits are saved until you acknowledge the errors at input ACK_REI of the RCVDP instruction.

Structure of DIAG of the SENDDP/RCVDP instructions

Bit no.	Assignment	Possible error causes	Remedies
Bit 0	Reserved	—	—
Bit 1	Reserved	—	—
Bit 2	Reserved	—	—
Bit 3	Reserved	—	—
Bit 4	Timeout of SENDDP/RCVDP detected	Interference in bus connection to partner F-CPU.	Check bus connection and ensure that no external interference sources are present.
		Monitoring time setting for F-CPU and partner F-CPU is too low.	Check assigned monitoring time TIMEOUT for SENDDP and RCVDP of both F-CPU's. If necessary, set a higher value. Recompile safety program
		DP/DP coupler or PN/PN coupler configuration is invalid.	Check DP/DP coupler or PN/PN coupler configuration.
		Internal fault of DP/DP coupler or PN/PN coupler	Replace DP/DP coupler or PN/PN coupler
		CP in STOP mode, or internal fault in CP	Switch CP to RUN mode, check diagnostic buffer of CP, and replace CP, if necessary
		F-CPU/partner F-CPU in STOP mode, or internal fault in F-CPU/partner F-CPU	Switch F-CPU's to RUN mode, check diagnostic buffer of F-CPU's, and replace F-CPU's, if necessary
Bit 5	Sequence number error of SENDDP/RCVDP detected	See description for bit 4	See description for bit 4
Bit 6	CRC error of SENDDP/RCVDP detected	See description for bit 4	See description for bit 4
Bit 7	Reserved	—	—

See also

Configuring and Programming Communication (Page 111)

Safety-related IO controller-IO controller communication (Page 114)

Safety-related master-master communication (Page 122)

Safety-related communication between I/O-controller and I-device (Page 130)

Safety-related master-I-slave communication (Page 136)

Safety-related IO Controller-I-slave communication (Page 154)

Communication with S7 Distributed Safety via PN/PN coupler (IO Controller-IO Controller communication) (Page 162)

Communication with S7 Distributed Safety via DP/DP coupler (master-master communication) (Page 163)

13.2.15.2 S7 communication

SENDS7 and RCVS7: Communication via S7 connections (STEP 7 Safety Advanced V11)

Introduction

You use the SENDS7 and RCVS7 instructions for fail-safe sending and receiving of data using S7 connections.

Note

In *STEP 7 Safety Advanced V11*, S7 connections are generally permitted over Industrial Ethernet only.

Safety-related communication via S7 connections is possible from and to F-CPU with PROFINET interface or S7-400 F-CPU with PROFINET-capable CPs. See also Safety-related communication via S7 connections (Page 155).

Description

The SENDS7 instruction sends the send data contained in an F-communication DB to the F-communication DB of the associated RCVS7 instruction of another F-CPU in a fail-safe manner using an S7 connection.

Every call of this instruction must be assigned a data area in which the instruction data are stored. The "Call options" dialog is automatically opened for this reason when the instruction is inserted in the program; in it you can create a data block (single instance) (e.g., SENDS7_DB_1) or a multi-instance (e.g., SENDS7_Instance_1) for this instruction. Once you have created the data block, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Information on the F-communication DB is contained in "Programming safety-related communication via S7 connections (Page 158)".

An F-communication DB is an F-DB for safety-related CPU-CPU communication with special properties. You must specify the numbers of the F-communication DBs at inputs SEND_DB and RCV_DB of instructions SENDS7 and RCVS7.

The operating mode of the F-CPU with the SENDS7 instruction is provided at output SENDMODE of the RCVS7 instruction. If the F-CPU with the SENDS7 instruction is in deactivated safety mode, then output SENDMODE = 1.

To reduce the bus load, you can temporarily shut down communication between the F-CPU at input EN_SEND of the SENDS7 instruction. To do so, supply input EN_SEND with "0" (default = "1"). In this case, send data are no longer sent to the F-communication DB of the associated RCVS7 instruction, and the receiver provides fail-safe values for this period of time (initial values in its F-communication DB). If communication was already established between the partners, a communication error is detected.

You must specify the local ID - from the perspective of the F-CPU - of the S7 connection (from the connection table in the network view) at input ID of the SENDS7 instruction.

Communication between F-CPU's takes place in the background by means of a special safety protocol. You must define a communication relationship between an SENDS7 instruction in one F-CPU and a communication relationship between an RCVS7 instruction and the other F-CPU by assigning an odd number at input R_ID (of the SENDS7 and RCVS7 instructions). Associated SENDS7 and RCVS7 instructions receive the same value for R_ID.

 **WARNING**

The value for each address relationship (input parameter R_ID; data type: DWORD) is user-defined; however, it must be an odd number and be unique from all other safety-related communication connections in the network. The value R_ID + 1 is internally assigned and must not be used.

You must supply inputs ID and R_ID with constant values when calling the instruction. Direct read or write access to the associated instance DB is not permitted in the safety program! (S020)

Note

A separate instance DP must be used for each call of the SENDS7 and RCVS7 instructions within a safety program. You must not declare and call these instructions as multi-instances.

The input and output parameters of the RCVS7 instruction must not be initialized with temporary or static local data of the main safety block.

The input parameters of the RCVS7 instruction must not be initialized with output parameters (using fully qualified DB accesses) of a RCVS7 or RCVDP instruction called in a preceding network.

You must not use an actual parameter for an output parameter of an RCVS7 instruction, if it is already being used for an input parameter of the same or another RCVS7 or RCVDP instruction. The F-CPU can go to STOP if this is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"
- "Safety program: internal CPU fault; internal error information: 404"

Note

You must not program any SENDS7 instruction between a JMP or JMPN instruction and the associated destination network of the JMP or JMPN instruction.

You must not program a RET instruction prior to a SENDS7 instruction.

Parameters of the SENDS7 instruction

Parameter	Declaration	Data type	Description
SEND_DB	Input	BLOCK_DB	Number of F-communication DB
TIMEOUT	Input	TIME	Monitoring time in ms for safety-related communication (see also Monitoring and response times (Page 523))
EN_SEND	Input	BOOL	1= Send enable
ID	Input	WORD	Local ID of the S7 connection
R_ID	Input	DWORD	Network-wide unique value for an address relationship between a SENDS7 instruction and a RCVS7 instruction
ERROR	Output	BOOL	1=Communication error
SUBS_ON	Output	BOOL	1=Receiving block outputs fail-safe values
STAT_RCV	Output	WORD	Status parameter STATUS of the URCV instruction (You can find a description of error codes in the online help for the URCV instruction ("Communication > S7 Communication"))
STAT_SND	Output	WORD	Status parameter STATUS of the USEND instruction (You can find a description of error codes in the online help for the USEND instruction ("Communication > S7 Communication"))
DIAG	Output	BYTE	Service information

Parameters of the RCVS7 instruction

Parameter	Declaration	Data type	Description
ACK_REI	Input	BOOL	Acknowledgment for reintegration of send data after communication error
RCV_DB	Input	BLOCK_DB	Number of F-communication DB
TIMEOUT	Input	TIME	Monitoring time in ms for safety-related communication (see also Monitoring and response times (Page 523))
ID	Input	WORD	Local ID of the S7 connection
R_ID	Input	DWORD	Network-wide unique value for an address relationship between a SENDS7 instruction and a RCVS7 instruction
ERROR	Output	BOOL	1=Communication error
SUBS_ON	Output	BOOL	1=Fail-safe values are output
ACK_REQ	Output	BOOL	1=Acknowledgment for reintegration of send data required
SENDMODE	Output	BOOL	1=F-CPU with the SENDS7 instruction in deactivated safety mode
STAT_RCV	Output	WORD	Status parameter STATUS of the URCV instruction (You can find a description of error codes in the online help for the URCV instruction ("Communication > S7 Communication"))
STAT_SND	Output	WORD	Status parameter STATUS of the USEND instruction (You can find a description of error codes in the online help for the USEND instruction ("Communication > S7 Communication"))
DIAG	Output	BYTE	Service information

Startup characteristics

After the sending and receiving F-systems are started up, communication must be established between the connection partners for the first time (SENDS7 and RCVS7 instructions). The receiver (RCVS7 instruction) provides fail-safe values for this time period (initial values in its F-communication DB).

The SENDS7 and RCVS7 instructions signal this at output SUBS_ON with 1. Output SENDMODE (RCVS7 instruction) has a default of 0 and is not updated, as long as output SUBS_ON = 1.

Behavior in the event of communication errors

If a communication error occurs, for example, due to a signature error (CRC) or when monitoring time TIMEOUT expires, outputs ERROR and SUBS_ON = 1 are set. The receiver (RCVS7 instruction) then provides fail-safe values (initial values in its F-communication DB). Output SENDMODE is not updated while output SUBS_ON = 1.

The send data present in the F-communication DB (SENDS7 instruction) are not output before the communication error is no longer detected ((ACK_REQ = 1)) and you acknowledge (Page 99) with a positive edge at input ACK_REI of the RCVS7 instruction.



WARNING

For the user acknowledgment, you must interconnect input ACK_REI with a signal generated by the operator input.

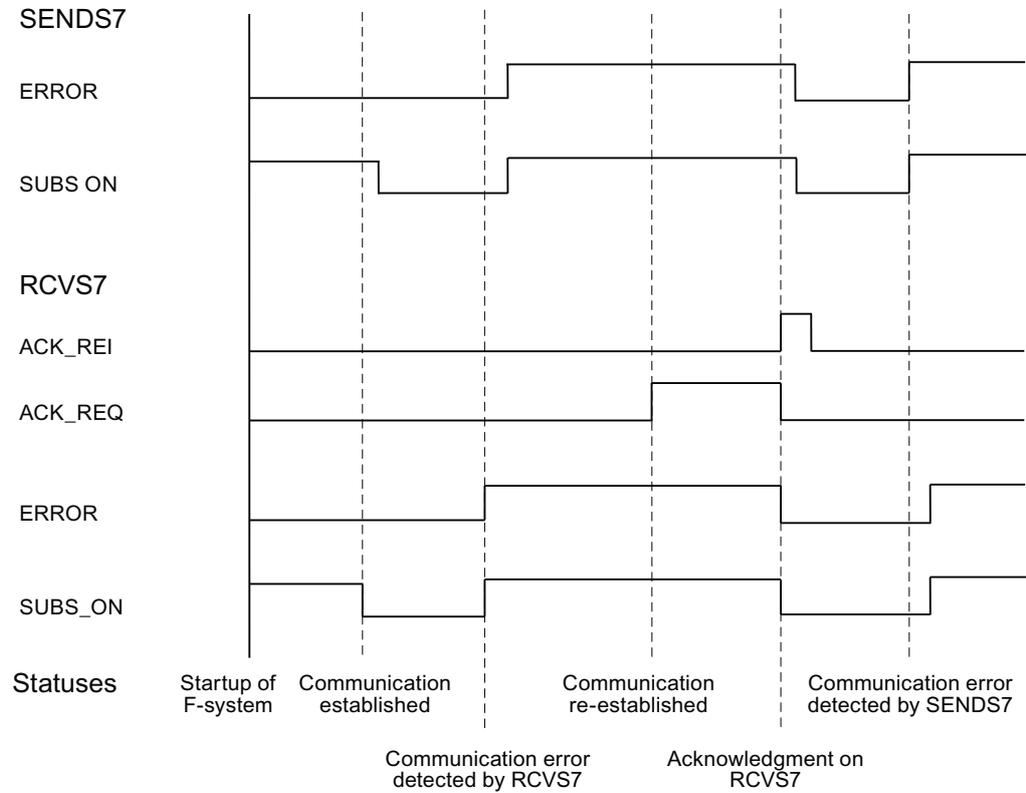
An interconnection with an automatically generated signal is not permitted. (S040)

Note that output ERROR (1=communication error) will be set for the first time on a communication error if communication has already been established between the connection partners (SENDS7 and RCVS7 instructions). If communication cannot be established after startup of the sending and receiving F-Systems, check the configuration of the safety-related CPU-CPU communication, parameter assignment of the SENDS7 and RCVS7 instructions, and the bus connection. You can also receive information on possible error causes by evaluating the STAT_RCV and STAT_SND outputs.

In general, always evaluate STAT_RCV and STAT_SND, since it is possible that only one of the two outputs will contain error information.

If one of the DIAG bits is set at output DIAG, also check whether the length and structure of the associated F-communication DB on both the sending and receiving ends match.

Timing diagrams SENDS7 and RCVS7



Output DIAG

Non-fail-safe information on the type of communication errors that have occurred is made available at output DIAG for service purposes. You can read out this information by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program. DIAG bits are saved until you acknowledge them at input ACK_REI of the associated RCVS7 instruction.

Structure of DIAG

Bit no.	Assignment of SENDS7 and RCVS7	Possible error causes	Remedies
Bit 0	Reserved	—	—
Bit 1	Reserved	—	—
Bit 2	Reserved	—	—
Bit 3	Reserved	—	—
Bit 4	Timeout detected by SENDS7 and RCVS7	Fault in bus connection to partner F-CPU	Check bus connection and ensure that no external fault sources are present.
		Monitoring time setting for F-CPU and partner F-CPU is too low	Check assigned monitoring time TIMEOUT for SENDS7 and RCVS7 of both F-CPU. If possible, set a higher value. Recompile safety program
		CPs in STOP mode, or internal fault in CPs	<ul style="list-style-type: none"> • Switch CPs to RUN mode • Check diagnostic buffer of CPs • Replace CPs, if necessary
		F-CPU/partner F-CPU in STOP mode, or internal fault in F-CPU/partner F-CPU	<ul style="list-style-type: none"> • Switch F-CPU to RUN mode • Check diagnostic buffer of F-CPU • Replace F-CPU, if necessary
		Communication was shut down with EN_SEND = 0.	Enable communication again at the associated SENDS7 with EN_SEND = 1
		S7 connection has changed, the IP address of the CP has changed, for example	Recompile the safety programs and download them to the F-CPU
Bit 5	Sequence number error detected by SENDS7 and RCVS7	See description for bit 4	See description for bit 4
Bit 6	CRC error detected by SENDS7 and RCVS7	See description for bit 4	See description for bit 4
Bit 7	RCVS7: Communication cannot be established	Configuration of the safety-related CPU-CPU communication is incorrect, parameter assignment of the SENDS7 and RCVS7 instructions is incorrect See also description for Bit 4	Check configuration of the safety-related CPU-CPU communication, parameter assignment of the SENDS7 and RCVS7 instructions is incorrect See also description for Bit 4
	SENDS7: Reserved	—	—

See also

Configuration (Page 25)

13.3 Instructions - FBD

13.3.1 General

13.3.1.1 New network (STEP 7 Safety Advanced V11)

Requirement

An F-block is open.

Procedure

To insert a new network, follow these steps:

1. Select the network after which you want to insert a new network.
2. Select the "Insert network" command in the shortcut menu.

Note

If you insert an element into the last empty network of the F-block in an FBD program, a new empty network is automatically inserted below it.

Result

A new empty network is inserted into the F-block.

13.3.1.2 Empty box (STEP 7 Safety Advanced V11)

Requirement

A network is available.

Procedure

To insert FBD elements into a network using an empty box, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "Basic instructions > General > Empty box".
3. Use a drag-and-drop operation to move the "Empty box" element to the desired place in the network.
4. Hover the cursor over the yellow triangle in the top right corner of the empty box.
A drop-down list is displayed.
5. Select the desired FBD element from the drop-down list.

If the instruction acts as a function block (FB) internally, the "Call options" dialog opens. In this dialog, you can create an instance data block for the function block, either as a single instance or multi-instance, in which data of the inserted instruction are stored. After its creation, the new instance data block can be found in the "Program resources" folder in the project tree under "Program blocks > System blocks". If you have selected "multi-instance", you will find it in the block interface in the "Static" section.

Result

The empty box is changed to the respective FBD element. Placeholders are inserted for the parameters.

13.3.1.3 Open branch (STEP 7 Safety Advanced V11)

Description

You use branches to program parallel connections with the Function Block Diagram (FBD) programming language. For this purpose, you use branches that you insert between the boxes. You can insert additional boxes in the branch, thereby programming complex function block diagrams.

Requirement

A network is available.

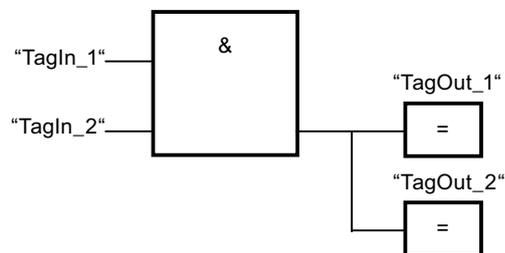
Procedure

To insert a new branch in a network, follow these steps:

1. Open the "Instructions" task card.
2. Navigate to "Basic instructions > General > Branch".
3. Use a drag-and-drop operation to move the element to the desired place in the network.

Example

The following figure provides an example of how to use branches:



13.3.1.4 Insert binary input (STEP 7 Safety Advanced V11)

Description

Use the "Insert binary input" instruction to expand the box of one of the following instructions by a binary input:

- "AND logic operation"
- "OR logic operation"
- "EXCLUSIVE OR logic operation"

You can query the signal state of several operands by expanding the instruction box.

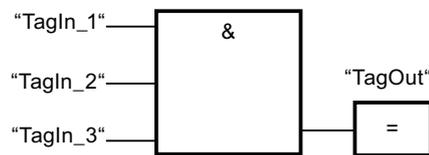
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand>	Input	BOOL	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:



The box of instruction "AND logic operation" has been expanded by an additional binary input at which the signal state of operand "TagIn_3" is queried. Output "TagOut" is set when the signal state of operands "TagIn_1", "TagIn_2", and "TagIn_3" is "1".

See also

AND logic operation (STEP 7 Safety Advanced V11) (Page 374)

OR logic operation (STEP 7 Safety Advanced V11) (Page 376)

X: EXCLUSIVE OR logic operation (STEP 7 Safety Advanced V11) (Page 377)

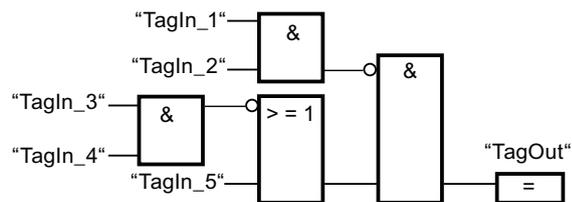
13.3.1.5 Invert RLO (STEP 7 Safety Advanced V11)

Description

You can use the "Invert RLO" instruction to invert the result of logic operation (RLO).

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- Input "TagIn_1" or "TagIn_2" has signal state "0".
- Input "TagIn_3" or "TagIn_4" has the signal state "0" or input "TagIn_5" has the signal state "1".

13.3.2 Bit logic operations

13.3.2.1 AND logic operation (STEP 7 Safety Advanced V11)

Description

You can use the "AND logic operation" instruction to query the signal states of two or more specified operands and evaluate them according to the AND truth table.

If the signal state of all the operands is "1", then the conditions are fulfilled and the instruction returns the result "1". If the signal state of one of the operands is "0", then the conditions are not fulfilled and the instruction generates the result "0".

If the "AND logic operation" instruction is the first instruction in a logic string, it saves the result of its signal state query in the RLO bit.

Each "AND logic operation" instruction that is not the first instruction in the logic string logically combines the result of its signal state query with the value saved in the RLO bit. This logical combination is performed according to the AND truth table.

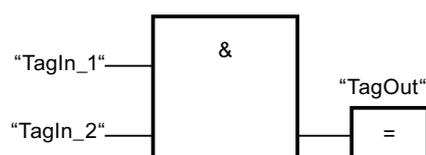
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand>	Input	BOOL	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the signal state of operands "TagIn_1" and "TagIn_2" is "1".

AND truth table

The following table shows the results when linking two operands by an AND logic operation:

Signal state of the first operand	Signal state of the second operand	Result of logic operation
1	1	1
0	1	0
1	0	0
0	0	0

See also

Insert binary input (STEP 7 Safety Advanced V11) (Page 372)

13.3.2.2 OR logic operation (STEP 7 Safety Advanced V11)

Description

You can use the "OR logic operation" instruction to get the signal states of two or more specified operands and evaluate them according to the OR truth table.

If the signal state of at least one of the operands is "1", then the conditions are fulfilled and the instruction returns the result "1". If the signal state of all of the operands is "0", then the conditions are not fulfilled and the instruction generates the result "0".

If the "OR logic operation" instruction is the first instruction in a logic string, it saves the result of its signal state query in the RLO bit.

Each "OR logic operation" instruction that is not the first instruction in the logic string, logically combines the result of its signal state query with the value saved in the RLO bit. This logical combination is performed according to the OR truth table.

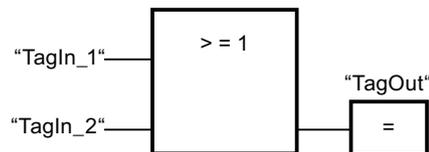
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand>	Input	BOOL	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the signal state of operand "TagIn_1" or "TagIn_2" is "1".

OR truth table

The following table shows the results when linking two operands by an OR logic operation:

Signal state of the first operand	Signal state of the second operand	Result of logic operation
1	0	1
0	1	1
1	1	1
0	0	0

See also

Insert binary input (STEP 7 Safety Advanced V11) (Page 372)

13.3.2.3 X: EXCLUSIVE OR logic operation (STEP 7 Safety Advanced V11)

Description

You can use the "EXCLUSIVE OR logic operation" instruction to get the result of a signal state query according to the the EXCLUSIVE OR truth table.

With an "EXCLUSIVE OR logic operation" instruction, the signal state is "1" when the signal state of one of the two specified operands is "1". When more than two operands are queried, the overall result is "1" if an odd-numbered quantity of queried operands returns the result "1".

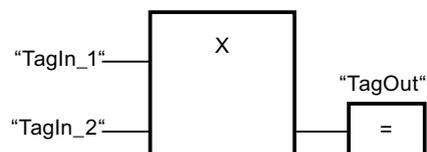
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand>	Input	BOOL	The operand indicates the bit whose signal state will be queried.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the signal state of one of the two operands "TagIn_1" and "TagIn_2" is "1". When both operands have signal state "1" or "0" then output "TagOut" is reset.

EXCLUSIVE OR truth table

The following table shows the results when two operands are linked by an EXCLUSIVE OR:

Signal state of the first operand	Signal state of the second operand	Result of logic operation
1	0	1
0	1	1
1	1	0
0	0	0

The following table shows the results when three operands are linked by an EXCLUSIVE OR:

Signal state of the first operand	Signal state of the second operand	Signal state of the third operand	Result of logic operation
1	0	0	1
0	1	1	0
0	1	0	1
1	0	1	0
0	0	1	1
1	1	0	0
1	1	1	1
0	0	0	0

See also

Insert binary input (STEP 7 Safety Advanced V11) (Page 372)

13.3.2.4 =: Assignment (STEP 7 Safety Advanced V11)

Description

You can use the "Assignment" instruction to set the bit of a specified operand. If the result of logic operation (RLO) at the box input has signal state "1", the specified operand is set to signal state "1". If the signal state at the box input is "0", the bit of the specified operand is reset to "0".

The instruction does not influence the RLO. The RLO at the box input is assigned directly to the operand located above the Assign box.

The "Assign" instruction can be placed at any position in the logic operation sequence.

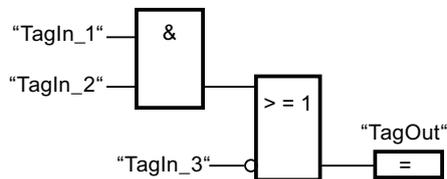
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand>	Input	BOOL	Operand to which the RLO is assigned.

Example

The following example shows how the instruction works:



Operand "TagOut" at the output of the "Assign" instruction is set when one of the following conditions is fulfilled:

- Inputs "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state at input "TagIn_3" is "0".

13.3.2.5 R: Reset output (STEP 7 Safety Advanced V11)

Description

You can use the "Reset output" instruction to reset the signal state of a specified operand to "0".

The instruction is only executed if the result of logic operation (RLO) is "1" at the box input. If the box input has signal state "1", the specified operand is reset to "0". If there is a result of logic operation of "0" at the box input, the signal state of the specified operand remains unchanged.

The instruction does not influence the RLO. The RLO at the box input is transferred directly to the box output.

Note

If you want to use a formal parameter of an F-FB/F-FC for the operand of the instruction then this must be declared as an in/out parameter.

The F-CPU can go to STOP if this caution is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
 - "Data corruption in the safety program prior to output to partner F-CPU"
 - "Safety program: internal CPU fault; internal error information: 404"
-

Note

If the operand area "local data" is used for the operand of the instruction then the local data bit used must be initialized beforehand.

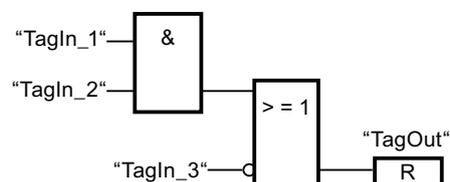
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand>	Output	BOOL	Operand that is reset with RLO = "1".

Example

The following example shows how the instruction works:



Operand "TagOut" is reset when one of the following conditions is fulfilled:

- Operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of operand "TagIn_3" is "0".

13.3.2.6 S: Set output (STEP 7 Safety Advanced V11)

Description

You can use the "Set output" instruction to set the signal state of a specified operand to "1".

The instruction is only executed if the result of logic operation (RLO) is "1" at the box input. If the box input has signal state "1", the specified operand is set to "1". If there is a result of logic operation of "0" at the box input, the signal state of the specified operand remains unchanged.

The instruction does not influence the RLO. The RLO at the box input is transferred directly to the box output.

Note

The instruction is not executed if it is applied to an output of an F-I/O that is passivated (e.g., during startup of the F-system). Therefore, it is preferable to access outputs of the F-I/O using only the "Assignment" instruction.

You can evaluate whether an F-I/O or channels of an F-I/O are passivated in the associated F-I/O DB.

Note

If you want to use a formal parameter of an F-FB/F-FC for the operand of the instruction then this must be declared as an in/out parameter.

The F-CPU can go to STOP if this caution is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
 - "Data corruption in the safety program prior to output to partner F-CPU"
 - "Safety program: internal CPU fault; internal error information: 404"
-

Note

If the operand area "local data" is used for the operand of the instruction then the local data bit used must be initialized beforehand.

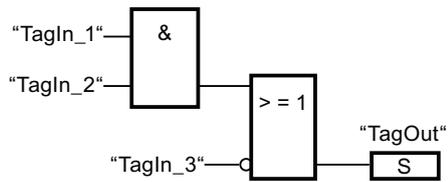
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand>	Output	BOOL	Operand that is set with RLO = "1".

Example

The following example shows how the instruction works:



Operand "TagOut" is set when one of the following conditions is fulfilled:

- Operands "TagIn_1" and "TagIn_2" have signal state "1".
- The signal state of operand "TagIn_3" is "0".

13.3.2.7 SR: Set/reset flip-flop (STEP 7 Safety Advanced V11)

Description

You can use the "Set/reset flip-flop" instruction to set or reset the bit of the specified operand based on the signal state at inputs S and R1. If the signal state at input S is "1" and the signal state at input R1 is "0", the specified operand is set to "1". If the signal state at input S is "0" and is "1" at input R1, the specified operand is reset to "0".

Input R1 takes priority over input S. If the signal state is "1" at the two inputs S and R1, the signal state of the specified operand is reset to "0".

The instruction is not executed if the signal state at the two inputs R and S1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

Note

If you want to use a formal parameter of an F-FB/F-FC for the operand of the instruction then this must be declared as an in/out parameter.

The F-CPU can go to STOP if this caution is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
 - "Data corruption in the safety program prior to output to partner F-CPU"
 - "Safety program: internal CPU fault; internal error information: 404"
-

Note

You cannot use the "process image input", "process image output", or "bit memory" operand areas for the operands of the instruction.

If the operand area "local data" is used for the edge bit memory of the instruction then the local data bit used must be initialized beforehand.

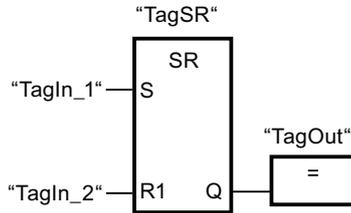
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
S	Input	BOOL	Enable setting
R1	Input	BOOL	Enable resetting
<Operand>	Output	BOOL	Operand that is set or reset.
Q	Output	BOOL	Signal state of the operand

Example

The following example shows how the instruction works:



Operands "TagSR" and "TagOut" are set when the following conditions are fulfilled:

- Operand "TagIn_1" has signal state "1".
- Operand "TagIn_2" has signal state "0".

Operands "TagSR" and "TagOut" are reset when the following conditions are fulfilled:

- Operand "TagIn_1" has signal state "0" and operand "TagIn_2" has signal state "1".
- Both operands "TagIn_1" and "TagIn_2" have signal state "1".

13.3.2.8 RS: Reset/set flip-flop (STEP 7 Safety Advanced V11)

Description

You can use the "Reset/set flip-flop" instruction to reset or set the bit of the specified operand based on the signal state of inputs R and S1. When the signal state is "1" at input R and "0" at input S1, the specified operand is reset to "0". When the signal state is "0" at input R and "1" at input S1, the specified operand is set to "1".

Input S1 takes priority over input R. If the signal state is "1" at the two inputs R and S1, the signal state of the specified operand is set to "1".

The instruction is not executed if the signal state at the two inputs R and S1 is "0". The signal state of the operand then remains unchanged.

The current signal state of the operand is transferred to output Q and can be queried there.

Note

If you want to use a formal parameter of an F-FB/F-FC for the operand of the instruction then this must be declared as an in/out parameter.

The F-CPU can go to STOP if this caution is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
 - "Data corruption in the safety program prior to output to partner F-CPU"
 - "Safety program: internal CPU fault; internal error information: 404"
-

Note

You cannot use the "process image input", "process image output", or "bit memory" operand areas for the operands of the instruction.

If the operand area "local data" is used for the edge bit memory of the instruction then the local data bit used must be initialized beforehand.

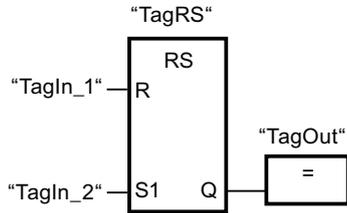
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
R	Input	BOOL	Enable resetting
S1	Input	BOOL	Enable setting
<Operand>	Output	BOOL	Operand that is reset or set.
Q	Output	BOOL	Signal state of the operand

Example

The following example shows how the instruction works:



Operands "TagRS" and "TagOut" are reset when the following conditions are fulfilled:

- Operand "TagIn_1" has signal state "1".
- Operand "TagIn_2" has signal state "0".

Operands "TagRS" and "TagOut" are set when the following conditions are fulfilled:

- Operand "TagIn_1" has signal state "0" and operand "TagIn_2" has signal state "1".
- Operands "TagIn_1" and "TagIn_2" have signal state "1".

13.3.2.9 P: Scan operand for positive signal edge (STEP 7 Safety Advanced V11)

Description

You can use the "Scan operand for positive signal edge" instruction to determine if there is a change from "0" to "1" in the signal state of a specified operand (<Operand1>). The instruction compares the current signal state of <Operand1> with the signal state of the previous query saved in <Operand2>. If the instruction detects a change in the result of logic operation from "0" to "1", there is a positive, rising edge.

If a rising edge is detected, the output of the instruction has signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Enter the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Enter the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the edge memory bit would be overwritten. This would influence edge evaluation and the result would no longer be unequivocal.

Note

If you want to use a formal parameter of an F-FB/F-FC for the edge memory bit <Operand2> of the instruction then this must be declared as an in/out parameter.

The F-CPU can go to STOP if this caution is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
 - "Data corruption in the safety program prior to output to partner F-CPU"
 - "Safety program: internal CPU fault; internal error information: 404"
-

Note

You cannot use the "process image input", "process image output", or "bit memory" operand areas for the edge memory bit <Operand2> of the instruction.

If operand area "local data" is used for the edge memory bit <Operand2> of the instruction then the local data bit used must be initialized beforehand.

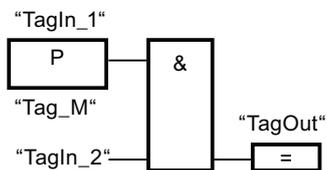
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand1>	Input	BOOL	Signal to be queried
<Operand2>	InOut	BOOL	Edge memory bit in which the signal state of the previous query is saved.

Example

The following example shows how the instruction works:



"TagOut" is set when the following conditions are fulfilled:

- There is a rising edge at input "TagIn_1".
- The signal state of operand "TagIn_2" is "1".

13.3.2.10 N: Scan operand for negative signal edge (STEP 7 Safety Advanced V11)**Description**

You can use the "Scan operand for negative signal edge" instruction to determine if there is a change from "1" to "0" in the signal state of a specified operand. The instruction compares the current signal state of <Operand1> with the signal state of the previous query saved in <Operand2>. If the instruction detects a change in the result of logic operation from "1" to "0", there is a negative, falling edge.

If a falling edge is detected, the output of the instruction has signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Enter the operand to be queried (<Operand1>) in the operand placeholder above the instruction. Enter the edge memory bit (<Operand2>) in the operand placeholder below the instruction.

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the edge memory bit would be overwritten. This would influence edge evaluation and the result would no longer be unequivocal.

Note

If you want to use a formal parameter of an F-FB/F-FC for the edge memory bit <Operand2> of the instruction then this must be declared as an in/out parameter.

The F-CPU can go to STOP if this caution is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"
- "Safety program: internal CPU fault; internal error information: 404"

Note

You cannot use the "process image input", "process image output", or "bit memory" operand areas for the edge memory bit <Operand2> of the instruction.

If operand area "local data" is used for the edge memory bit <Operand2> of the instruction then the local data bit used must be initialized beforehand.

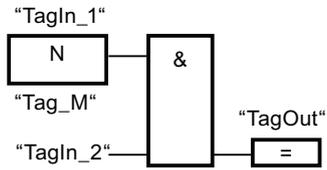
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Operand1>	Input	BOOL	Signal to be queried
<Operand2>	InOut	BOOL	Edge memory bit in which the signal state of the previous query is saved.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- There is a falling edge at input "TagIn_1".
- The signal state of operand "TagIn_2" is "1".

13.3.2.11 P_TRIG: Scan RLO for positive signal edge (STEP 7 Safety Advanced V11)

Description

You can use the "Scan RLO for positive signal edge" instruction to query a change in the signal state of the result of logic operation from "0" to "1". The instruction compares the current signal state of the result of logic operation with the signal state from the previous query, which is saved in the edge memory bit (<Operand>). If the instruction detects a change in the RLO from "0" to "1", there is a positive, rising edge.

If a rising edge is detected, the output of the instruction has signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the edge memory bit would be overwritten. This would influence edge evaluation and the result would no longer be unequivocal.

Note

If you want to use a formal parameter of an F-FB/F-FC for the edge memory bit <Operand> of the instruction then this must be declared as an in/out parameter.

The F-CPU can go to STOP if this caution is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
 - "Data corruption in the safety program prior to output to partner F-CPU"
 - "Safety program: internal CPU fault; internal error information: 404"
-

Note

You cannot use the "process image input", "process image output", or "bit memory" operand areas for the edge memory bit <Operand> of the instruction.

If operand area "local data" is used for the edge memory bit <Operand> of the instruction then the local data bit used must be initialized beforehand.

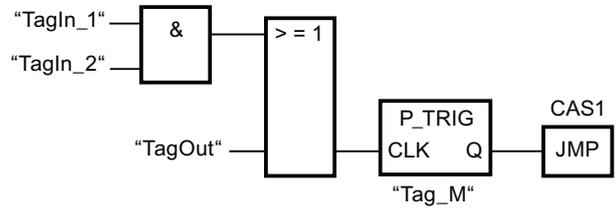
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
CLK	Input	BOOL	Current RLO
<Operand>	InOut	BOOL	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO from the previous bit logic operation is saved in edge memory bit "Tag_M". If a change in the RLO signal state from "0" to "1" is detected, the program jumps to jump label CAS1.

13.3.2.12 N_TRIG: Scan RLO for negative signal edge (STEP 7 Safety Advanced V11)

Description

You can use the "Scan RLO for negative signal edge" instruction to query a change in the signal state of the result of logic operation from "1" to "0". The instruction compares the current signal state of the result of logic operation with the signal state from the previous query, which is saved in the edge memory bit (<Operand>). If the instruction detects a change in the RLO from "1" to "0", there is a negative, falling edge.

If a falling edge is detected, the output of the instruction has signal state "1". In all other cases, the signal state at the output of the instruction is "0".

Note

The address of the edge memory bit must not be used more than once in the program, otherwise the edge memory bit would be overwritten. This would influence edge evaluation and the result would no longer be unequivocal.

Note

If you want to use a formal parameter of an F-FB/F-FC for the edge memory bit <Operand> of the instruction then this must be declared as an in/out parameter.

The F-CPU can go to STOP if this caution is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
 - "Data corruption in the safety program prior to output to partner F-CPU"
 - "Safety program: internal CPU fault; internal error information: 404"
-

Note

You cannot use the "process image input", "process image output", or "bit memory" operand areas for the edge memory bit <Operand> of the instruction.

If operand area "local data" is used for the edge memory bit <Operand> of the instruction then the local data bit used must be initialized beforehand.

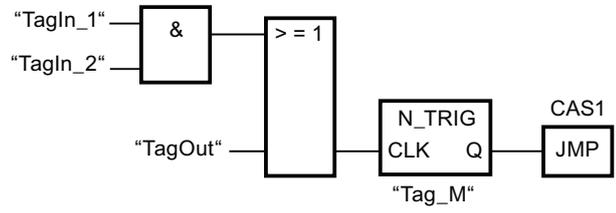
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
CLK	Input	BOOL	Current RLO
<Operand>	InOut	BOOL	Edge memory bit in which the RLO of the previous query is saved.
Q	Output	BOOL	Result of edge evaluation

Example

The following example shows how the instruction works:



The RLO of the previous bit logic operation is saved in edge bit memory "Tag_M". If a change in the RLO signal state from "1" to "0" is detected, the program jumps to jump label CAS1.

13.3.3 Safety functions

13.3.3.1 ESTOP1: Emergency Stop up to Stop Category 1 (STEP 7 Safety Advanced V11)

Description

This instruction implements an emergency STOP shutdown with acknowledgment for Stop Categories 0 and 1.

Enable signal Q is reset to 0, as soon as input E_STOP takes a signal state of 0 (Stop category 0). Enable signal Q_DELAY is reset to 0 after the time delay set at input TIME_DEL (Stop Category 1).

Enable signal Q is reset to 1 not before input E_STOP takes a signal state of 1 and an acknowledgment occurs. The acknowledgment for the enable takes place according to the parameter assignment at input ACK_NEC:

- If ACK_NEC = 0, the acknowledgment is automatic.
- If ACK_NEC = 1, you must use a rising edge at input ACK for acknowledging the enable.

Output ACK_REQ is used to signal that a user acknowledgment is required at input ACK for the acknowledgment. The instruction sets output ACK_REQ to 1, as soon as input E_STOP = 1.

Following an acknowledgment, the instruction resets ACK_REQ to 0.

Every call of the "Emergency STOP up to Stop Category 1" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., ESTOP1_DB_1) or a multi-instance (e.g., ESTOP1_Instance_1) for the "Emergency STOP up to Stop Category 1" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

WARNING

Tag ACK_NEC must not be assigned a value of 0 unless an automatic restart of the affected process is otherwise excluded. (*S033*)

⚠ WARNING

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction")
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision. (S034)

Note: Only one emergency STOP signal (E_STOP) can be evaluated for the instruction. With suitable configuration (type of sensor interconnection: 2-channel equivalent), Discrepancy monitoring of the two NC contacts (when two channels are involved) in accordance with Categories 3 and 4 as defined in EN 954-1 is performed directly by the F-I/O with inputs. 2-channel equivalent) directly through the F-I/O with inputs. In order to keep the discrepancy time from influencing the response time, during the configuration of discrepancy behavior, you must parameterize "provide 0 value".

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
E_STOP	Input	BOOL	Emergency STOP
ACK_NEC	Input	BOOL	1=Acknowledgment necessary
ACK	Input	BOOL	1=Acknowledgment
TIME_DEL	Input	TIME	Time delay
Q	Output	BOOL	1=Enable
Q_DELAY	Output	BOOL	Enable is OFF delayed
ACK_REQ	Output	BOOL	1=Acknowledgment necessary
DIAG	Output	BYTE	Service information

Instruction versions

Two versions are available for this instruction:

- Version 1.0

When projects that were created with *S7 Distributed Safety V5.4 SP5* are migrated, Version 1.0 of the instruction is used automatically.

Version 1.0 requires that the F_TOF block with the number FB 186 is available in the project tree in the "Program blocks/System blocks/STEP 7 Safety" folder.

If you want to compile a migrated safety program with *STEP 7 Safety Advanced V11* for the first time, we recommend that you update the version of the ESTOP1 instruction to Version 1.1 beforehand. You will then avoid number conflicts.

- Version 1.1

When a new project is created with *STEP 7 Safety Advanced V11*, Version V1.1 is preset automatically. This version is functionally identical to Version V1.0, but does not require the F_TOF block to have a particular number.

For more information on the use of instruction versions, refer to the help on *STEP 7 Professional* under "Using instruction versions".

Startup characteristics

After an F-system startup, when ACK_NEC = 1, you must acknowledge the instruction using a rising edge at input ACK.

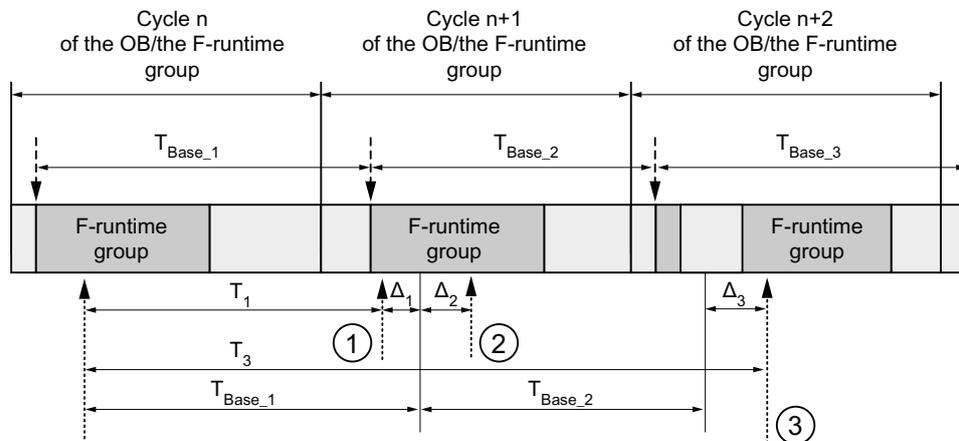
Output DIAG

The DIAG output provides non-fail-safe information on errors for service purposes. You can read out this information by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program. DIAG bits 1 to 5 are saved until acknowledgment at input ACK.

Structure of DIAG

Bit no.	Assignment	Possible error causes	Remedies
Bit 0	Incorrect TIM_DEL setting	Time delay setting < 0	Set time delay > 0
Bit 1	Reserved	—	—
Bit 2	Reserved	—	—
Bit 3	Reserved	—	—
Bit 4	Acknowledgment not possible because emergency STOP is still active	Emergency STOP switch is locked	Release interlocking of emergency STOP switch
		F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON of F-I/O of emergency STOP switch	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O access (Page 79)
		Emergency STOP switch is defective	Check emergency STOP switch
		Wiring fault	Check wiring of emergency STOP switch
Bit 5	If enable is missing: input ACK has a permanent signal state of 1	Acknowledgment button defective	Check acknowledgment button
		Wiring fault	Check wiring of acknowledgment button
Bit 6	Acknowledgement required (= state of ACK_REQ)	—	—
Bit 7	State of output Q	—	—

Timing imprecision resulting from the update time of the time base used in the instruction:



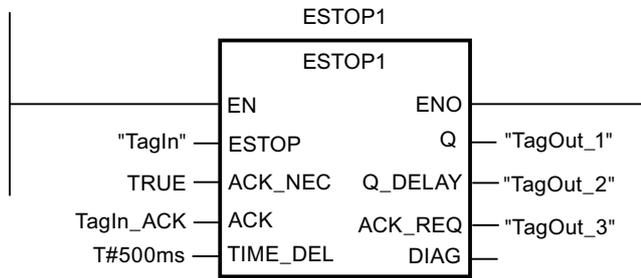
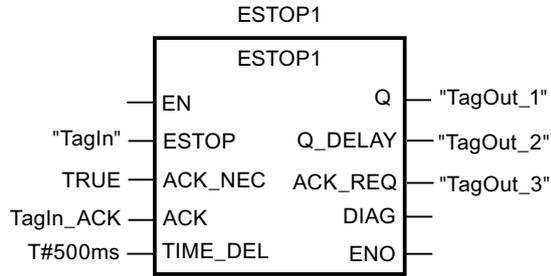
-----> = Time base update

.....> = Call time of an instruction with time processing

- ① For the first call in cycle n+1, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle n+1 are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle n+1. This does not involve another time update (by Δ_2).
- ③ For the call in cycle n+2, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle n+2. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle n+1.

Example

The following example shows how the instruction works:



13.3.3.2 TWO_HAND: Two-hand monitoring (STEP 7 Safety Advanced V11)

Description

This instruction implements two-hand monitoring.

If pushbuttons IN1 and IN2 are activated within the permitted discrepancy time $DISCTIME \leq 500$ ms ($IN1/IN2 = 1$) (synchronous activation), output signal Q is set to 1. If the time difference between activation of pushbutton IN1 and pushbutton IN2 is greater than $DISCTIME$, then the pushbuttons must be released and reactivated.

Q is reset to 0 as soon as one of the pushbuttons is released ($IN1/IN2 = 0$). Enable signal Q can be reset to 1 only if the other pushbutton has been released, and if both pushbuttons are then reactivated within the discrepancy time. Enable signal Q can never be set to 1 if the discrepancy time is set to values less than 0 or greater than 500 ms.

Every call of the "Two-hand monitoring" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., `TWO_HAND_DB_1`) or a multi-instance (e.g., `TWO_HAND_Instance_1`) for the "Two-hand monitoring" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

The instruction supports the requirements in accordance with EN 574:1996 + A1:2008.

Note: Only one signal per pushbutton can be evaluated in the instruction. Discrepancy monitoring of the NC and NO contacts of pushbuttons IN1 and IN2 is performed directly during suitable configuration (type of sensor interconnection: 2-channel equivalent) directly through the F-I/O with inputs. The normally open contact must be wired in such a way that it supplies the useful signal (see manual for the F-I/O you are using). In order to keep the discrepancy time from influencing the response time, you must assign "Provide value 0" for the behavior of discrepancy during configuration. If a discrepancy is detected, a fail-safe value of 0 is entered in the process image input (PII) for the pushbutton and QBAD or `QBAD_I_xx = 1` is set in the relevant F-I/O DB. (See also F-I/O access (Page 79))

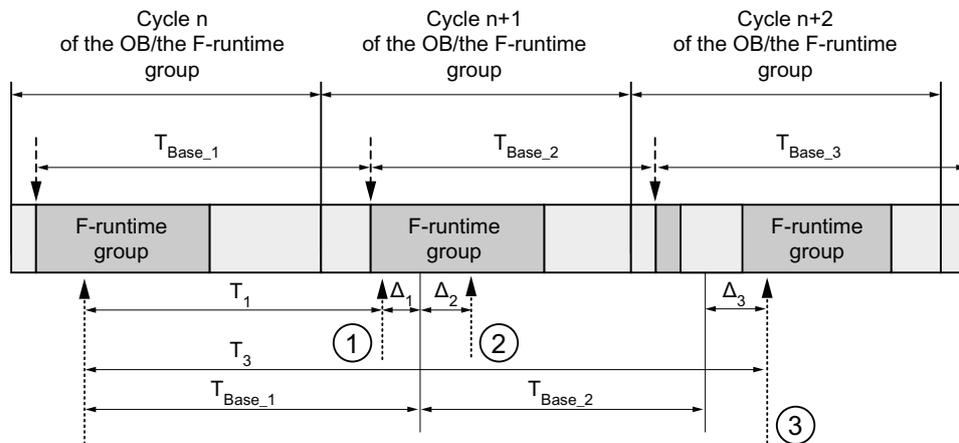
 WARNING
<p>When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:</p> <ul style="list-style-type: none"> • Known timing imprecision (based on standard systems) resulting from cyclic processing • Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction") • Tolerance of internal time monitoring in the F-CPU <ul style="list-style-type: none"> – For time values up to 100 ms, a maximum of 20% of the (assigned) time value – For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value <p>You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision. (S034)</p>

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN1	Input	BOOL	Pushbutton 1
IN2	Input	BOOL	Pushbutton 2
DISCTIME	Input	TIME	Discrepancy time (0 to 500 ms)
Q	Output	BOOL	1=Enable

Timing imprecision resulting from the update time of the time base used in the instruction:



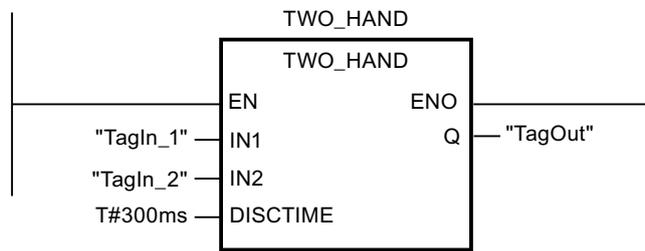
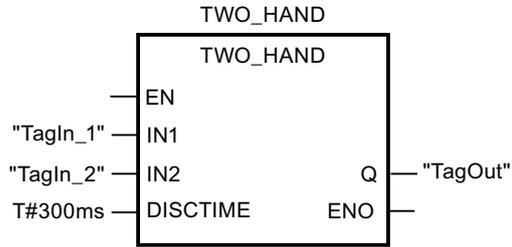
-----> = Time base update

.....> = Call time of an instruction with time processing

- ① For the first call in cycle n+1, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle n+1 are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle n+1. This does not involve another time update (by Δ_2).
- ③ For the call in cycle n+2, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle n+2. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle n+1.

Example

The following example shows how the instruction works:



13.3.3.3 TWO_H_EN: Two-hand monitoring with enable (STEP 7 Safety Advanced V11)

Description

This instruction implements two-hand monitoring with enable.

If pushbuttons IN1 and IN2 are activated within the permitted discrepancy time $DISCTIME \leq 500$ ms ($IN1/IN2 = 1$) (synchronous activation), output signal Q is set to 1 when existing $ENABLE = 1$. If the time difference between activation of pushbutton IN1 and pushbutton IN2 is greater than $DISCTIME$, then the pushbuttons must be released and reactivated.

Q is reset to 0 as soon as one of the pushbuttons is released ($IN1/IN2 = 0$) or $ENABLE = 0$. Enable signal Q can be reset to 1 only if the other pushbutton has been released, and if both pushbuttons are then reactivated within the discrepancy time when existing $ENABLE = 1$.

Every call of the "Two-hand monitoring with enable" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., `TWO_H_EN_DB_1`) or a multi-instance (e.g., `TWO_H_EN_Instance_1`) for the "Two-hand monitoring with enable" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

The instruction supports the requirements in accordance with EN 574:1996 + A1:2008.

Note: Only one signal per pushbutton can be evaluated in the instruction. Discrepancy monitoring of the NC and NO contacts of pushbuttons IN1 and IN2 is performed directly during suitable configuration (type of sensor interconnection: 2-channel equivalent) directly through the F-I/O with inputs. The normally open contact must be wired in such a way that it supplies the useful signal (see manual for the F-I/O you are using). In order to keep the discrepancy time from influencing the response time, during the configuration of discrepancy behavior, you must parameterize "provide 0 value".

If a discrepancy is detected, a fail-safe value of 0 is entered in the process image input (PII) for the pushbutton and $QBAD$ or $QBAD_I_xx = 1$ is set in the relevant F-I/O DB.

⚠ WARNING

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction")
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision. (S034)

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN1	Input	BOOL	Pushbutton 1
IN2	Input	BOOL	Pushbutton 2
ENABLE	Input	BOOL	Enable input
DISCTIME	Input	TIME	Discrepancy time (0 to 500 ms)
Q	Output	BOOL	1=Enable
DIAG	Output	BYTE	Service information

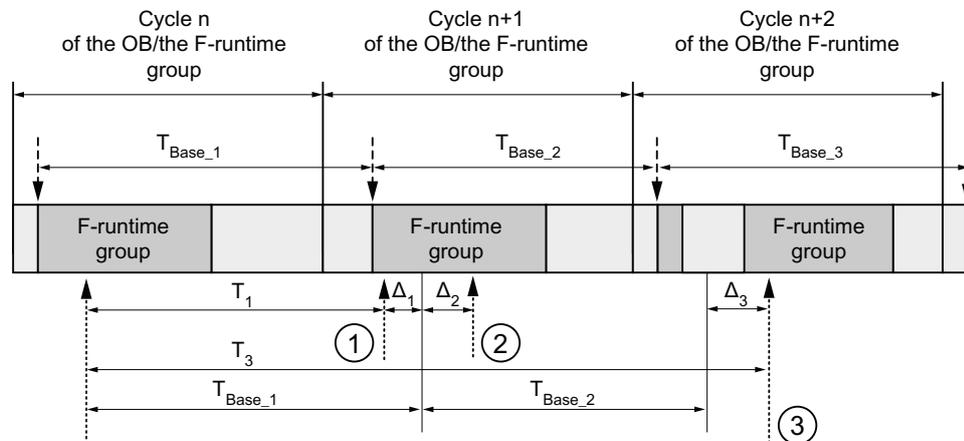
Output DIAG

The DIAG output provides non-fail-safe information on errors for service purposes. You can read out this information by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program. DIAG bits 0 to 5 are saved until the cause of the error has been eliminated.

Structure of DIAG

Bit no.	Assignment	Possible error causes	Remedies
Bit 0	Incorrect discrepancy time DISCTIME setting	Discrepancy time setting is <0 or > 500 ms	Set discrepancy time in range of 0 to 500 ms
Bit 1	Discrepancy time elapsed	Discrepancy time setting is too low	If necessary, set a higher discrepancy time
		Pushbuttons were not activated within the discrepancy time	Release pushbuttons and activate them within the discrepancy time
		Wiring fault	Check wiring of pushbuttons
		Pushbuttons defective	Check pushbuttons
		Pushbuttons are wired to different F-I/O, and F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON on an F-I/O	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
Bit 2	Reserved	—	—
Bit 3	Reserved	—	—
Bit 4	Incorrect activation sequence	One pushbutton was not released	Release pushbuttons and activate them within the discrepancy time
		Pushbuttons defective	Check pushbuttons
Bit 5	ENABLE does not exist	ENABLE = 0	Set ENABLE = 1, release pushbutton and activate it within the discrepancy time
Bit 6	Reserved	—	—
Bit 7	State of output Q	—	—

Timing imprecision resulting from the update time of the time base used in the instruction:



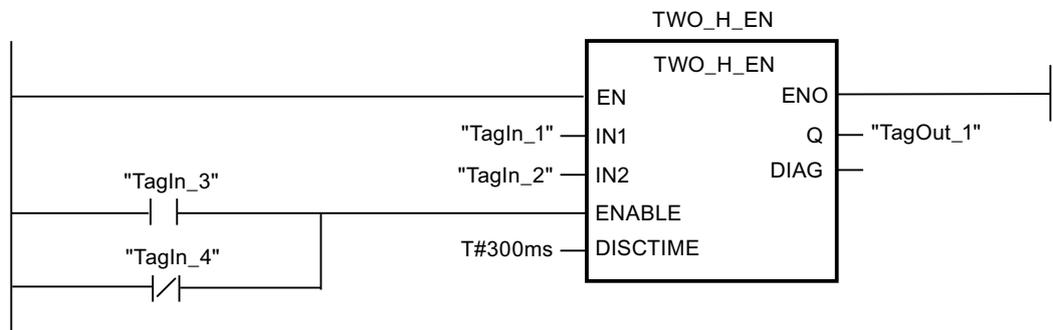
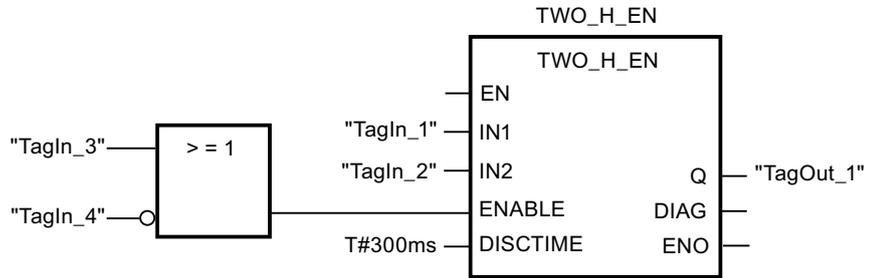
-----▶ = Time base update

.....▶ = Call time of an instruction with time processing

- ① For the first call in cycle n+1, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle n+1 are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle n+1. This does not involve another time update (by Δ_2).
- ③ For the call in cycle n+2, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle n+2. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle n+1.

Example

The following example shows how the instruction works:



13.3.3.4 MUTING: Muting (STEP 7 Safety Advanced V11)

Description

This instruction performs parallel muting with two or four muting sensors.

Muting is a defined suppression of the protective function of light curtains. Light curtain muting can be used to introduce goods or objects into the danger area monitored by the light curtain without causing the machine to stop.

To utilize the muting function, at least two independently wired muting sensors must be present. The use of two or four muting sensors and correct integration into the production sequence must ensure that no persons enter the danger area while the light curtain is muted.

Every call of the "Muting" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., MUTING_DB_1) or a multi-instance (e.g., MUTING_Instance_1) for the "Muting" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

WARNING

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction")
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

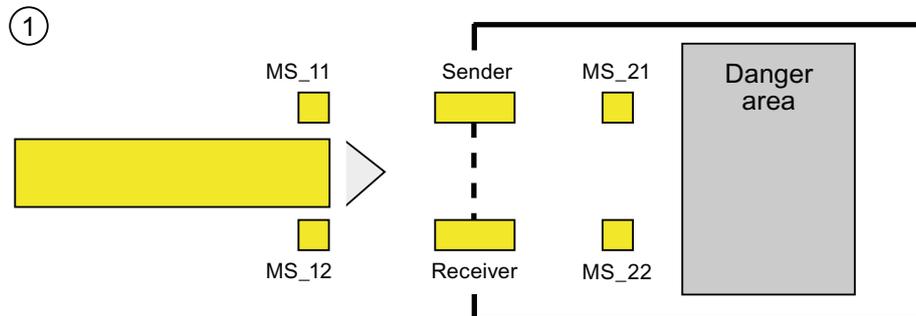
You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision. (S034)

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
MS_11	Input	BOOL	Muting sensor 1 of sensor pair 1
MS_12	Input	BOOL	Muting sensor 2 of sensor pair 1
MS_21	Input	BOOL	Muting sensor 1 of sensor pair 2
MS_22	Input	BOOL	Muting sensor 2 of sensor pair 2
STOP	Input	BOOL	1=Conveyor system stopped
FREE	Input	BOOL	1=Light curtain uninterrupted
QBAD_MUT	Input	BOOL	QBAD or QBAD_O_xx signal of F-I/O/channel of muting lamp (F-I/O DB)
DISCTIM1	Input	TIME	Discrepancy time of sensor pair 1 (0 to 3 s)
DISCTIM2	Input	TIME	Discrepancy time of sensor pair 2 (0 to 3 s)
TIME_MAX	Input	TIME	Maximum muting time (0 to 10 min)
ACK	Input	BOOL	Acknowledgment of restart inhibit
Q	Output	BOOL	1= Enable, not off
MUTING	Output	BOOL	Display of muting is active
ACK_REQ	Output	BOOL	Acknowledgment necessary
FAULT	Output	BOOL	Group error
DIAG	Output	BYTE	Service information

Schematic sequence of error-free muting procedure with 4 muting sensors (MS_11, MS_12, MS_21, MS_22)

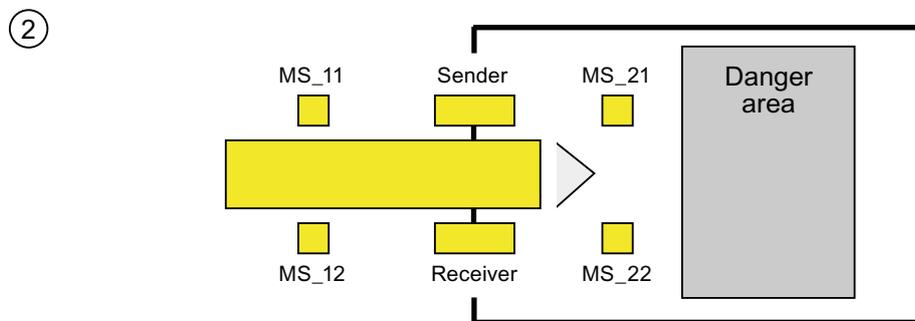


- If both muting sensors MS_11 and MS_12 are activated by the product within DISCTIM1 (apply signal state = 1), the instruction starts the MUTING function. Enable signal Q remains 1, even when input FREE = 0 (light curtain interrupted by product). The MUTING output for setting the muting lamp switches to 1.

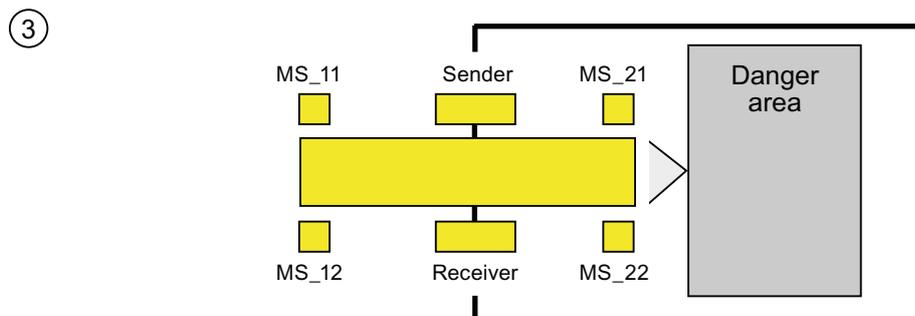
Note

The muting lamp can be monitored using the QBAD_MUT input. To do this, you must wire the muting lamp to an output with wire break monitoring of an F-I/O and supply the QBAD_MUT input with the QBAD or QBAD_O_xx signal of the associated F-I/O or channel. If QBAD_MUT = 1, muting is terminated by the instruction. If monitoring of the muting lamp is not necessary, you do not have to supply input QBAD_MUT.

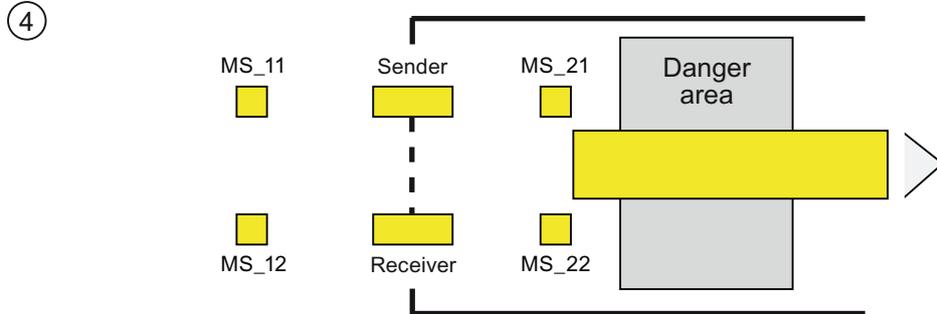
F-I/O that can promptly detect a wire break after activation of the muting operation must be used (*see manual for specific F-I/O*).



- As long as both muting sensors MS_11 and MS_12 continue to be activated, the MUTING function of the instruction causes Q to remain 1 and MUTING to remain 1 (so that the product can pass through the light curtain without causing the machine to stop).



- The two muting sensors MS_21 and MS_22 must be activated (within DISCTIM2) before muting sensors MS_11 and MS_12 are switched to inactive (apply signal state 0). In this way, the instruction retains the MUTING function. (Q = 1, MUTING = 1).

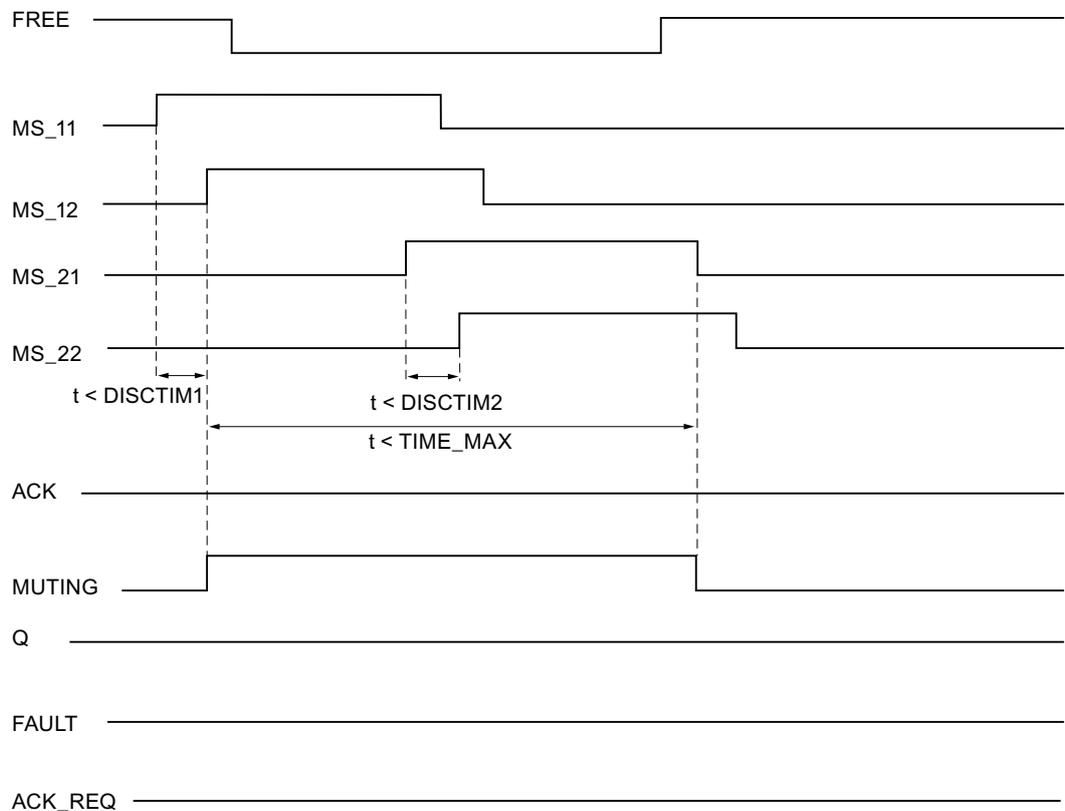


- Only if one of the two muting sensors MS_21 and MS_22 is switched to inactive (product enables sensors) is the MUTING function terminated (Q = 1, MUTING = 0). The maximum activation time for the MUTING function is the time set at input TIME_MAX.

Note

The MUTING function is also started if the product passes the light curtain in the reverse direction and the muting sensors are thus activated by the product in reverse order.

Timing diagrams for error-free muting procedure with 4 muting sensors

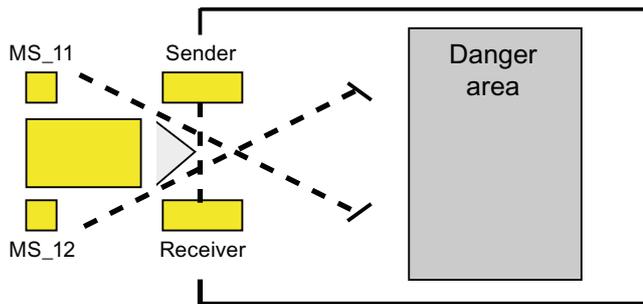


Schematic sequence of muting procedure with reflection light barriers

If reflection light barriers are used as muting sensors, they are generally arranged diagonally.

In general, this arrangement of reflection light barriers as muting sensors requires only two light barriers, and only MS_11 and MS_12 are interconnected.

The sequence is similar to that of the muting procedure with 4 muting sensors. Step 3 is omitted. In step 4, replace MS_21 and MS_22 with MS_11 and MS_12, respectively.



Restart inhibit upon interruption of light curtain (if MUTING is not active), when errors occur, and during F-system startup

Enable signal Q cannot be set to 1 or becomes 0, if:

- Light curtain is interrupted (e.g., by a person or material transport) while the MUTING function is not active
- The muting lamp monitoring function responds at input QBAD_MUT
- Sensor pair 1 (MS_11 and MS_12) or sensor pair 2 (MS_21 and MS_22) is not activated or deactivated during discrepancy time DISCTIM1 or DISCTIM2, respectively
- The MUTING function is active longer than the maximum muting time TIME_MAX
- Discrepancy times DISCTIM1 and DISCTIM2 have been set to values < 0 or > 3 s
- Maximum muting time TIME_MAX has been set to a value < 0 or > 10 min

In the identified cases, output FAULT (group error) is set to 1 (restart inhibit). If the MUTING function is started, it will be terminated and the Muting output becomes 0.

WARNING

When a valid combination of muting sensors is immediately detected at startup of the F-system (for example, because the muting sensors are interconnected to inputs of a standard I/O that immediately provide process values during the F-system startup), the MUTING function is immediately started and the MUTING output and enable signal Q are set to 1. The FAULT output (group error) is not set to 1 (no restart inhibit!). (S035)

Acknowledgment of restart inhibit

Enable signal Q becomes 1 again, if:

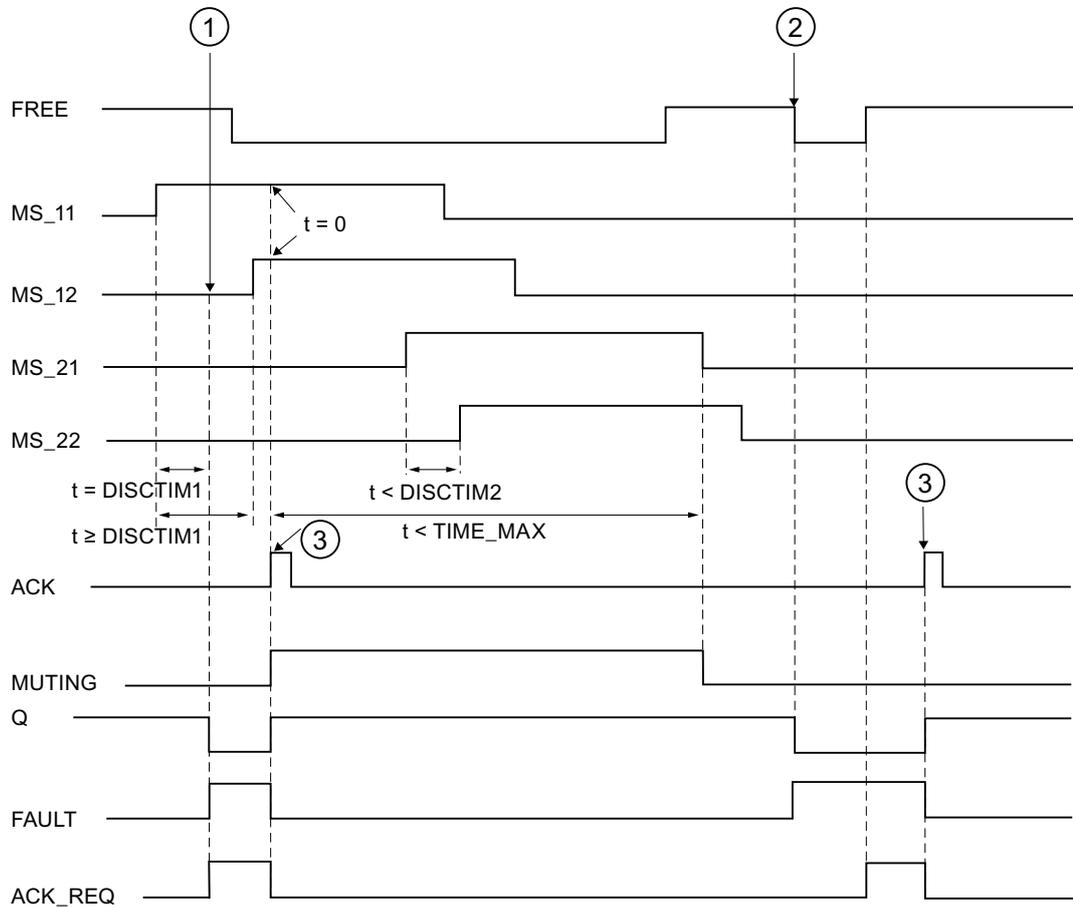
- The light curtain is no longer interrupted
 - Errors, if present, are eliminated (see output DIAG)
- and
- A user acknowledgement with positive edge is occurs at input ACK (see also Implementation of user acknowledgment (Page 99)).

The FAULT output is set to 0. Output ACK_REQ = 1 signals that user acknowledgment at input ACK is required to eliminate the restart inhibit. The instruction sets ACK-REQ = 1 as soon as the light curtain is no longer interrupted or errors have been eliminated. Once acknowledgment has occurred, the instruction resets ACK_REQ to 0.

Note

Following discrepancy errors and once the maximum muting time has been exceeded, ACK_REQ is immediately set to 1. As soon as a user acknowledgment has taken place at input ACK, discrepancy times DISCTIM1 and DISCTIM2 and maximum muting time TIME_MAX are reset.

Timing diagrams for discrepancy errors at sensor pair 1 or interruption of the light curtain (if MUTING is not active)



- ① Sensor pair 1 (MS_11 and MS_12) is not activated within discrepancy time DISCTIM1.
- ② The light curtain is interrupted even though the MUTING function is not active.
- ③ Acknowledgment

Behavior with stopped conveyor equipment

If, while the conveyor equipment has stopped, the monitoring for one of the following reasons is to be deactivated:

- To comply with discrepancy time DISCTIM1 or DISCTIM2
- To comply with maximum muting time TIME_MAX

you must supply input STOP with a "1" signal for as long as the conveyor equipment is stopped. As soon as the conveyor equipment is running again (STOP = 0), discrepancy times DISCTIM1 and DISCTIM2 and maximum muting time TIME_MAX are reset.

WARNING

When STOP = 1, the discrepancy monitoring is shut down. During this time, if inputs MSx1/MSx2 of a sensor pair both take a signal state of 1 due to an undetected error, e.g., because both muting sensors fail to 1, the error is not detected and the MUTING function can be started unintentionally. (S036)

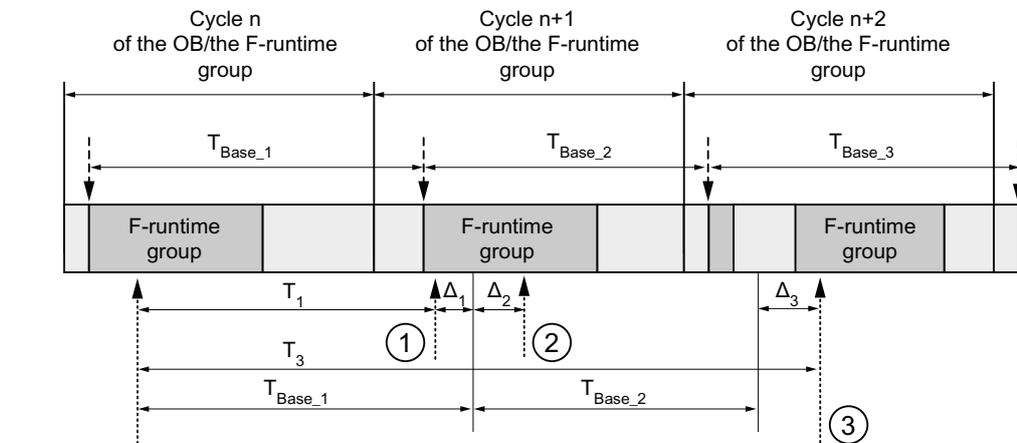
Output DIAG

The DIAG output provides non-fail-safe information on errors for service purposes. You can read out this information by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program. DIAG bits are saved until acknowledgment at input ACK.

Structure of DIAG

Bit no.	Assignment	Possible error causes	Remedies
Bit 0	Discrepancy error or incorrect discrepancy time DISCTIM 1 setting for sensor pair 1	Malfunction in production sequence	Malfunction in production sequence eliminated
		Sensor defective	Check sensors
		Wiring fault	Check wiring of sensors
		Sensors are wired to different F-I/O, and F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON on an F-I/O	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
		Discrepancy time setting is too low	If necessary, set a higher discrepancy time
		Discrepancy time setting is < 0 s or > 3 s	Set discrepancy time in range between 0 s and 3 s
Bit 1	Discrepancy error or incorrect discrepancy time DISCTIM 2 setting for sensor pair 2	Same as Bit 0	Same as Bit 0
Bit 2	Maximum muting time exceeded or incorrect muting time TIME_MAX setting	Malfunction in production sequence	Malfunction in production sequence eliminated
		Maximum muting time setting is too low	If necessary, set a higher maximum muting time
		Muting time setting is < 0 s or > 10 min	Set muting time in range from 0 s to 10 min
Bit 3	Light curtain interrupted and muting not active	Light curtain is defective	Check light curtain
		Wiring fault	Check wiring of light curtain (FREE input)
		F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON of F-I/O of light curtain (FREE input)	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
		See other DIAG bits	
Bit 4	Muting lamp is defective or cannot be set	Muting lamp is defective	Replace muting lamp
		Wiring fault	Check wiring of muting lamp
		F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON of F-I/O of muting lamp	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
Bit 5	Reserved	—	—
Bit 6	Reserved	—	—
Bit 7	Reserved	—	—

Timing imprecision resulting from the update time of the time base used in the instruction:



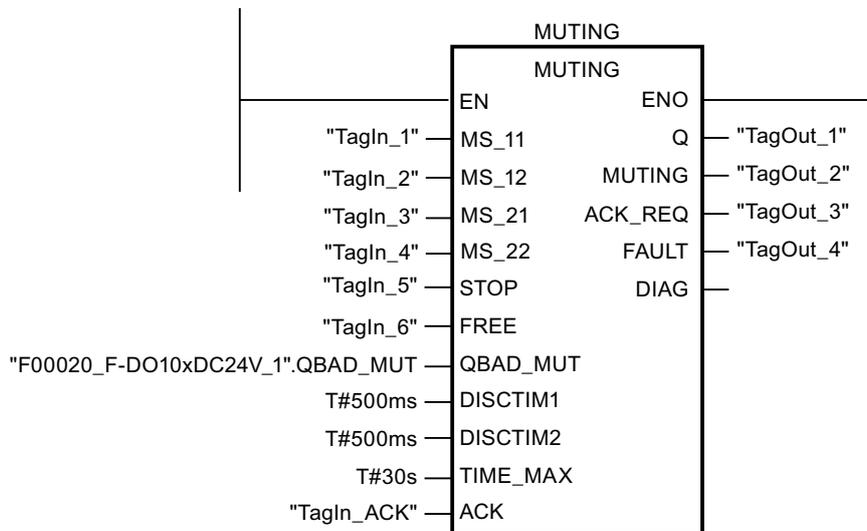
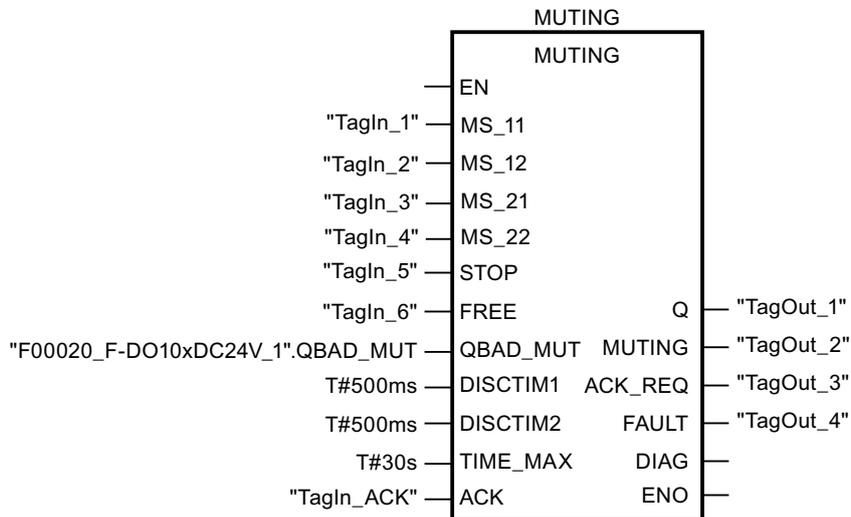
-----> = Time base update

.....> = Call time of an instruction with time processing

- ① For the first call in cycle n+1, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle n+1 are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle n+1. This does not involve another time update (by Δ_2).
- ③ For the call in cycle n+2, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle n+2. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle n+1.

Example

The following example shows how the instruction works:



13.3.3.5 MUT_P: Parallel muting (STEP 7 Safety Advanced V11)

Description

This instruction performs parallel muting with two or four muting sensors.

Muting is a defined suppression of the protective function of light curtains. Light curtain muting can be used to introduce goods or objects into the danger area monitored by the light curtain without causing the machine to stop.

To utilize the muting function, at least two independently wired muting sensors must be present. The use of two or four muting sensors and correct integration into the production sequence must ensure that no persons enter the danger area while the light curtain is muted.

Every call of the "Parallel muting" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., MUT_P_DB_1) or a multi-instance (e.g., MUT_P_Instance_1) for the "Parallel muting" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

WARNING

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction")
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

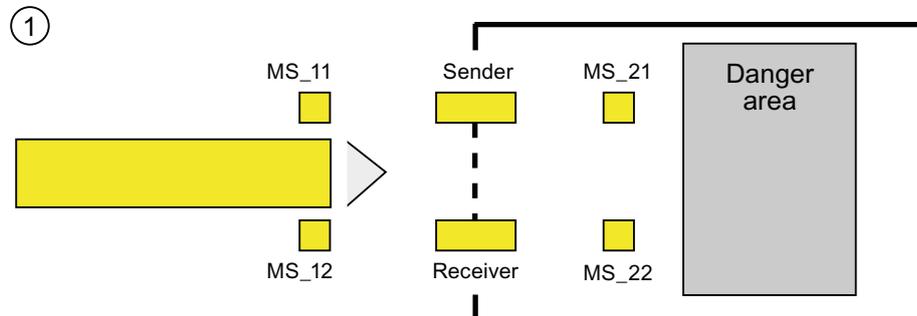
You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision. (S034)

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
MS_11	Input	BOOL	Muting sensor 11
MS_12	Input	BOOL	Muting sensor 12
MS_21	Input	BOOL	Muting sensor 21
MS_22	Input	BOOL	Muting sensor 22
STOP	Input	BOOL	1=Conveyor system stopped
FREE	Input	BOOL	1=Light curtain uninterrupted
ENABLE	Input	BOOL	1=Enable MUTING
QBAD_MUT	Input	BOOL	QBAD or QBAD_O_xx signal of F-I/O/channel of muting lamp (F-I/O DB)
ACK	Input	BOOL	Acknowledgment of restart inhibit
DISCTIM1	Input	TIME	Discrepancy time of sensor pair 1 (0 to 3 s)
DISCTIM2	Input	TIME	Discrepancy time of sensor pair 2 (0 to 3 s)
TIME_MAX	Input	TIME	Maximum muting time (0 to 10 min)
Q	Output	BOOL	1= Enable, not off
MUTING	Output	BOOL	Display of muting is active
ACK_REQ	Output	BOOL	Acknowledgment necessary
FAULT	Output	BOOL	Group error
DIAG	Output	WORD	Service information

Schematic sequence of error-free muting procedure with 4 muting sensors (MS_11, MS_12, MS_21, MS_22)

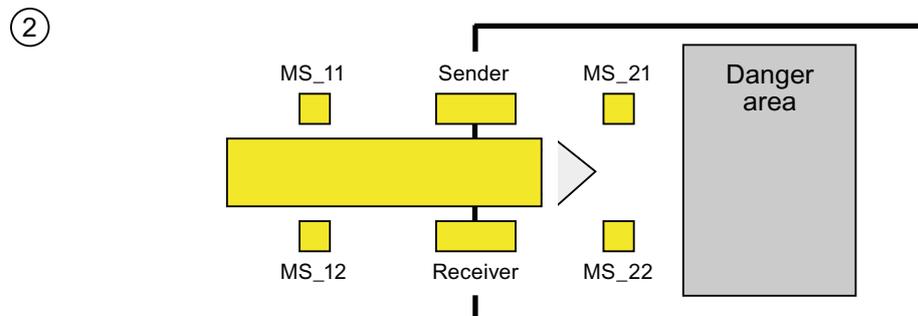


- If muting sensors MS_11 and MS_12 are both activated by the product within DISCTIM1 (apply signal state = 1) and MUTING is enabled by setting the ENABLE input to 1, the instruction starts the MUTING function. Enable signal Q remains 1, even when input FREE = 0 (light curtain interrupted by product). The MUTING output for setting the muting lamp switches to 1.

Note

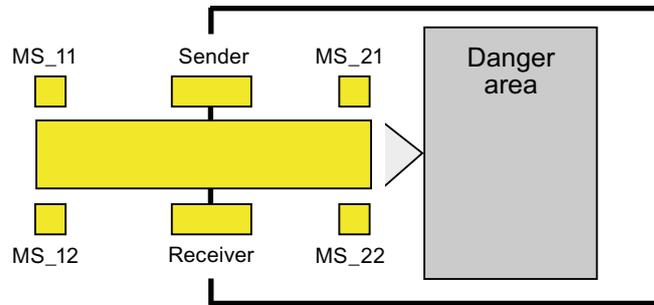
The muting lamp can be monitored using the QBAD_MUT input. To do this, you must wire the muting lamp to an output with wire break monitoring of an F-I/O and supply the QBAD_MUT input with the QBAD or QBAD_O_xx signal of the associated F-I/O or channel. If QBAD_MUT = 1, muting is terminated by the instruction. If monitoring of the muting lamp is not necessary, you do not have to supply input QBAD_MUT.

F-I/O that can promptly detect a wire break after activation of the muting operation must be used (*see manual for specific F-I/O*).



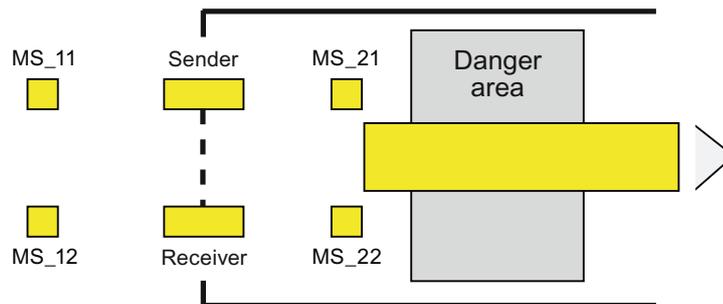
- As long as both muting sensors MS_11 and MS_12 continue to be activated, the MUTING function of the instruction causes Q to remain 1 and MUTING to remain 1 (so that the product can pass through the light curtain without causing the machine to stop). Each of the two muting sensors MS_11 and MS_12 may be switched to inactive ($t < DISCTIM1$) for a short time (apply signal state 0).

③



- Muting sensors MS_21 and MS_22 must both be activated (within DISCTIM2) before muting sensors MS_11 and MS_12 are switched to inactive (apply signal state 0). In this way, the instruction retains the MUTING function. ($Q = 1$, $MUTING = 1$).

④

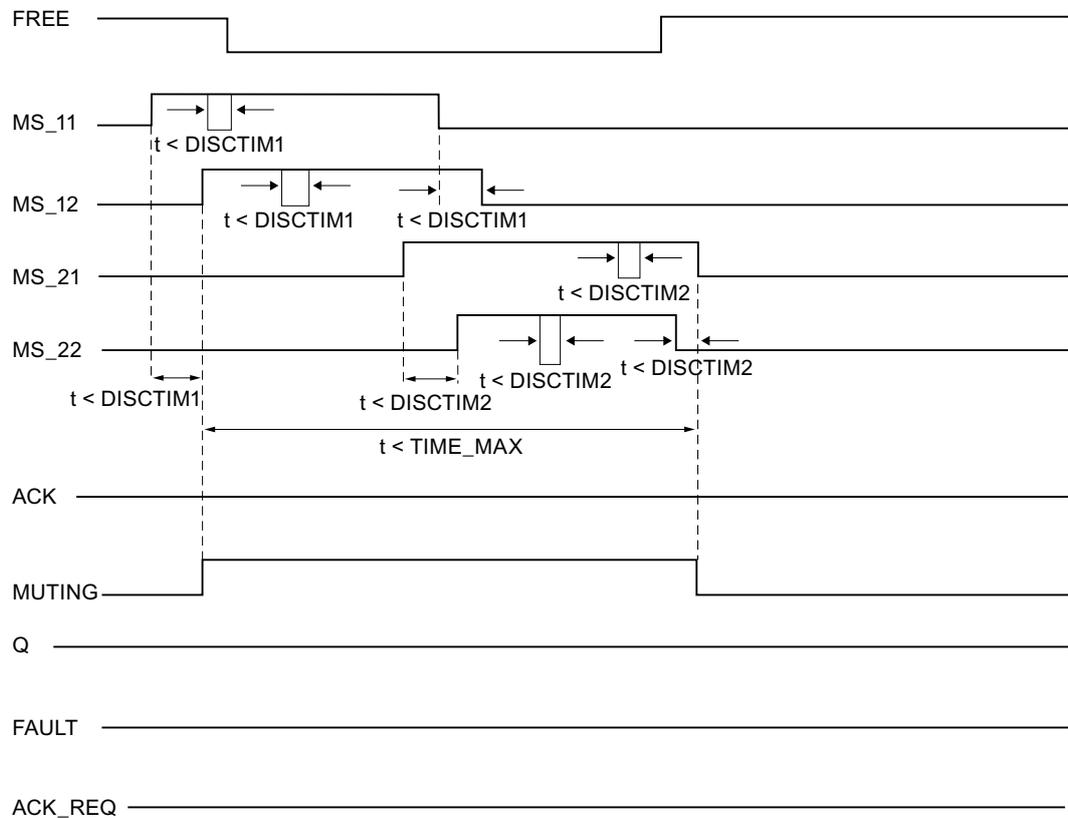


Only if muting sensors MS_21 and MS_22 are both switched to inactive (product enables sensors) is the MUTING function terminated ($Q = 1$, $MUTING = 0$). The maximum activation time for the MUTING function is the time set at input TIME_MAX.

Note

The MUTING function is also started if the product passes the light curtain in the reverse direction and the muting sensors are thus activated by the product in reverse order.

Timing diagrams for error-free muting procedure with 4 muting sensors

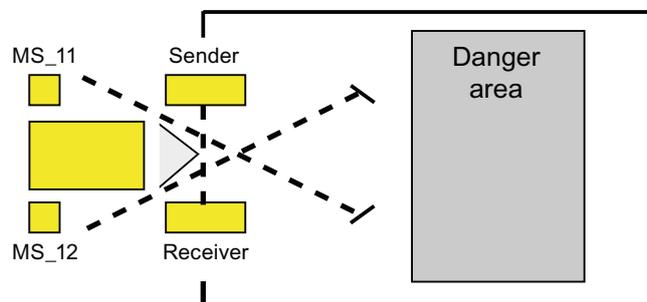


Schematic sequence of muting procedure with reflection light barriers

If reflection light barriers are used as muting sensors, they are generally arranged diagonally.

In general, this arrangement of reflection light barriers as muting sensors requires only two light barriers, and only MS_11 and MS_12 are interconnected.

The sequence is similar to that of the muting procedure with 4 muting sensors. Step 3 is omitted. In step 4, replace MS_21 and MS_22 with MS_11 and MS_12, respectively.



Restart inhibit upon interruption of light curtain (MUTING is not active), as well as when errors occur and during F-system startup

Enable signal Q cannot be set to 1 or becomes 0, if:

- Light curtain is interrupted (e.g., by a person or material transport) while the MUTING function is not active
- Light curtain is (being) interrupted and the muting lamp monitoring at input QBAD_MUT is set to 1
- Light curtain is (being) interrupted and the MUTING function is not enabled by setting input ENABLE to 1
- Sensor pair 1 (MS_11 and MS_12) or sensor pair 2 (MS_21 and MS_22) is not activated or deactivated during discrepancy time DISCTIM1 or DISCTIM2, respectively
- The MUTING function is active longer than the maximum muting time TIME_MAX
- Discrepancy times DISCTIM1 and DISCTIM2 have been set to values < 0 or > 3 s
- Maximum muting time TIME_MAX has been set to a value < 0 or > 10 min
- The F-system starts up (regardless of whether or not the light curtain is interrupted, because the F-I/O is passivated after F-system startup and, thus, the FREE input is initially supplied with 0)

In the identified cases, output FAULT (group error) is set to 1 (restart inhibit). If the MUTING function is started, it will be terminated and the Muting output becomes 0.

User acknowledgment of restart inhibit (no muting sensor is activated or ENABLE = 0)

Enable signal Q becomes 1 again, if:

- The light curtain is no longer interrupted
 - Errors, if present, are eliminated (see output DIAG)
- and
- A user acknowledgement with positive edge occurs at input ACK (see also Implementation of user acknowledgment (Page 99)).

The FAULT output is set to 0. Output ACK_REQ = 1 signals that user acknowledgment at input ACK is required to eliminate the restart inhibit. The instruction sets ACK_REQ = 1 as soon as the light curtain is no longer interrupted or the errors have been eliminated. Once acknowledgment has occurred, the instruction resets ACK_REQ to 0.

User Acknowledgment of restart inhibit (at least one muting sensor is activated and ENABLE = 1)

Enable signal Q becomes 1 again, if:

- Errors, if present, are eliminated (see output DIAG)
- FREE occurs until a valid combination of muting sensors is detected

The FAULT output is set to 0. The MUTING function is restarted, if necessary, and the MUTING output becomes 1 if a valid combination of muting sensors is detected. When ENABLE = 1, output ACK_REQ = 1 signals that FREE is necessary for error elimination and for removal of the restart inhibit. Once FREE has occurred, the instruction resets ACK_REQ to 0.

Note

Once the maximum muting time is exceeded, TIME_MAX is reset as soon as the MUTING function is restarted.

FREE function

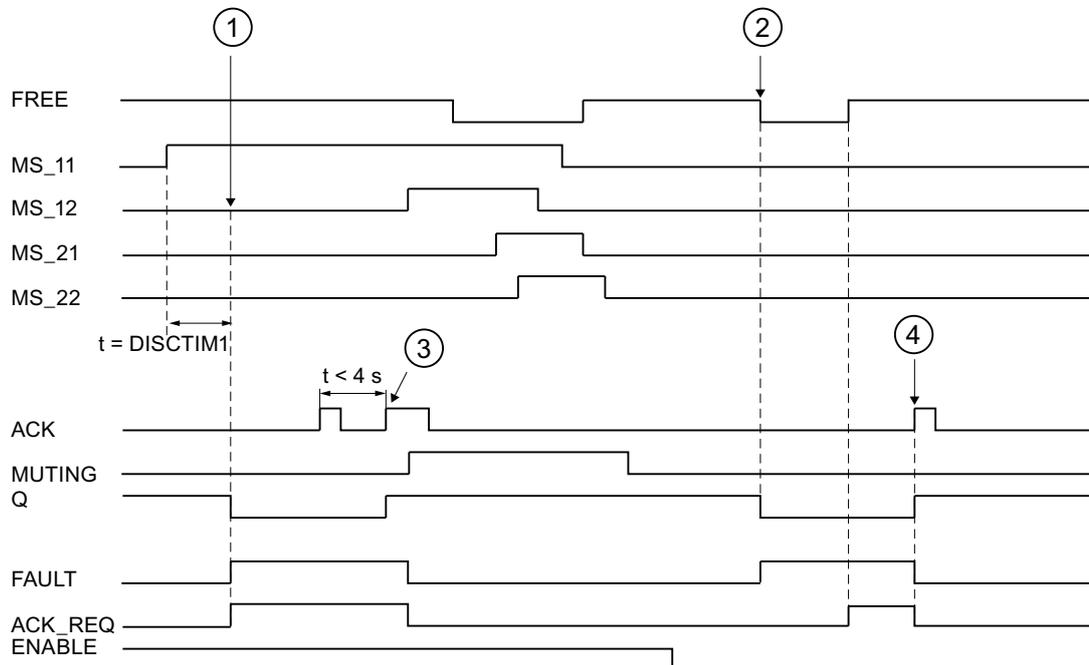
If an error cannot be corrected immediately, the FREE function can be used to free the muting range. Enable signal Q and output MUTING = 1 temporarily. The FREE function can be used if:

- ENABLE = 1
- At least one muting sensor is activated
- A user acknowledgment with rising edge at input ACK occurs twice within 4 s, and the second user acknowledgment at input ACK remains at a signal state of 1 (acknowledgment button remains activated)

 WARNING

When using the FREE function, the action must be observed. A dangerous situation must be able to be interrupted at any time by releasing the acknowledgment button. The acknowledgment button must be mounted in such a way the entire danger area can be observed. (S037)

Timing diagrams for discrepancy errors at sensor pair 1 or interruption of the light curtain (MUTING is not active)



- ① Sensor pair 1 (MS_11 and MS_22) is not activated within discrepancy time DISCTIM1.
- ② The light curtain is interrupted even though there is no enable (ENABLE=0)
- ③ FREE function
- ④ Acknowledgment

Behavior with stopped conveyor equipment

If, while the conveyor equipment has stopped, the monitoring for one of the following reasons is to be deactivated:

- To comply with discrepancy time DISCTIM1 or DISCTIM2
- To comply with maximum muting time TIME_MAX

you must supply input STOP with a "1" signal for as long as the conveyor equipment is stopped. As soon as the conveyor equipment is running again (STOP = 0), discrepancy times DISCTIM1 and DISCTIM2 and maximum muting time TIME_MAX are reset.

 WARNING
<p>When STOP = 1 or ENABLE = 0, discrepancy monitoring is shut down. During this time, if inputs MSx1/MSx2 of a sensor pair both take a signal state of 1 due to an undetected error, e.g., because both muting sensors fail to 1, the fault is not detected and the MUTING function can be started unintentionally (when ENABLE = 1). (S038)</p>

Output DIAG

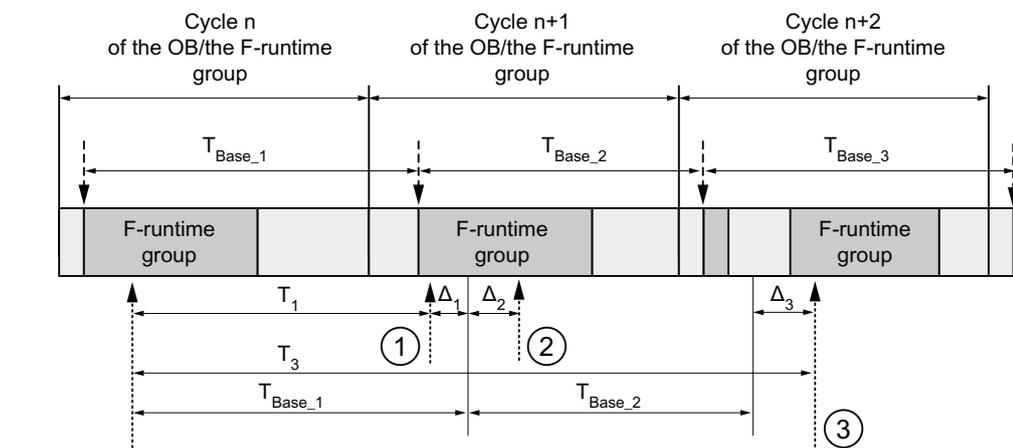
The DIAG output provides non-fail-safe information on errors for service purposes. You can read out this information by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program. DIAG bits 0 to 6 are saved until acknowledgment at input ACK.

Structure of DIAG

Bit no.	Assignment	Possible error causes	Remedies
Bit 0	Discrepancy error or incorrect discrepancy time DISCTIM 1 setting for sensor pair 1	Malfunction in production sequence	Malfunction in production sequence eliminated
		Sensor defective	Check sensors
		Wiring fault	Check wiring of sensors
		Sensors are wired to different F-I/O, and F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON on an F-I/O	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
		Discrepancy time setting is too low	If necessary, set a higher discrepancy time
		Discrepancy time setting is < 0 s or > 3 s	Set discrepancy time in range between 0 s and 3 s
Bit 1	Discrepancy error or incorrect discrepancy time DISCTIM 2 setting for sensor pair 2	Same as Bit 0	Same as Bit 0
Bit 2	Maximum muting time exceeded or incorrect muting time TIME_MAX setting	Malfunction in production sequence	Malfunction in production sequence eliminated
		Maximum muting time setting is too low	If necessary, set a higher maximum muting time
		Muting time setting is < 0 s or > 10 min	Set muting time in range from 0 s to 10 min
Bit 3	Light curtain interrupted and muting not active	ENABLE = 0	Set ENABLE = 1
		Light curtain is defective	Check light curtain
		Wiring fault	Check wiring of light curtain (FREE input)
		I/O fault, channel fault, or communication error, or passivation by means of PASS_ON of F-I/O of light curtain (FREE input)	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
		Startup of F-system	For FREE, see DIAG Bit 5
		See other DIAG bits	
Bit 4	Muting lamp is defective or cannot be set	Muting lamp is defective	Replace muting lamp
		Wiring fault	Check wiring of muting lamp
		F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON of F-I/O of muting lamp	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)

Bit no.	Assignment	Possible error causes	Remedies
Bit 5	FREE is necessary	See other DIAG bits	Two rising edges at ACK within 4 s, and activate acknowledgment button until ACK_REQ = 0
Bit 6	Acknowledgment necessary	—	—
Bit 7	State of output Q	—	—
Bit 8	State of output MUTING	—	—
Bit 9	FREE active	—	—
Bit 10	Reserved	—	—
...			
Bit 15	Reserved	—	—

Timing imprecision resulting from the update time of the time base used in the instruction:



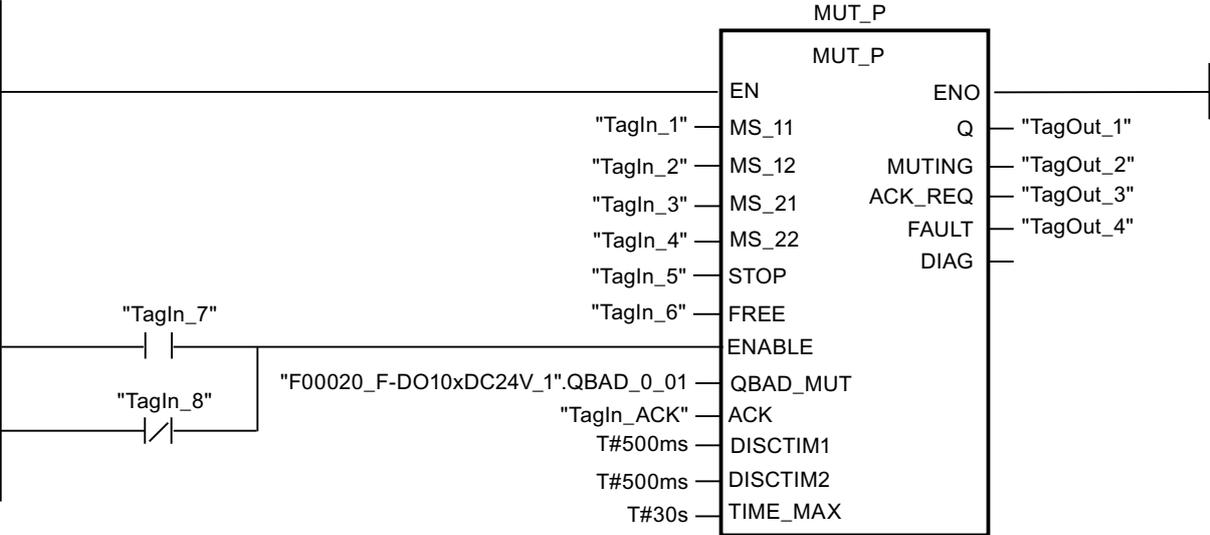
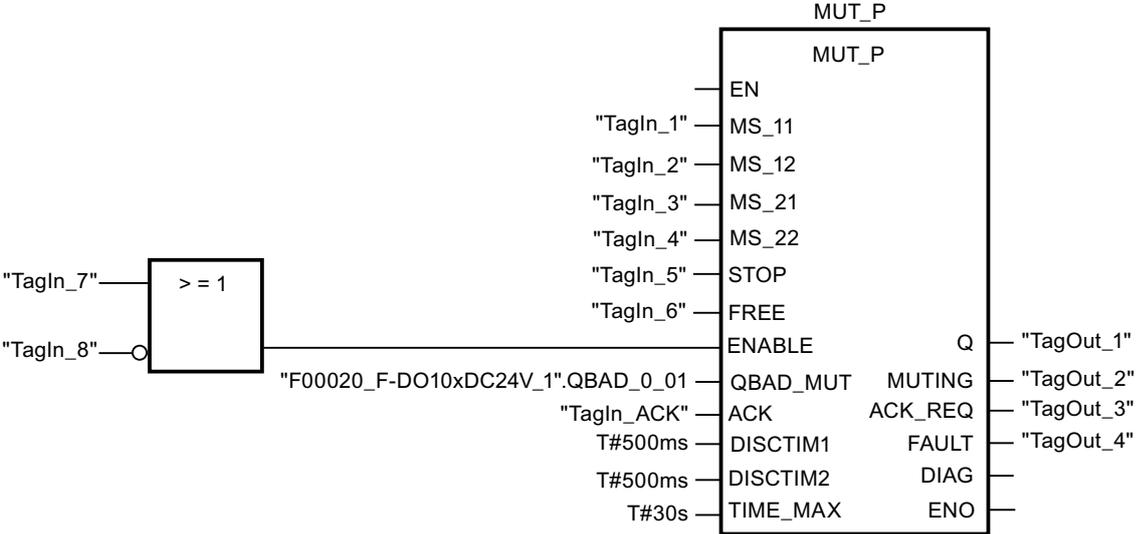
-----> = Time base update

.....> = Call time of an instruction with time processing

- ① For the first call in cycle n+1, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle n+1 are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle n+1. This does not involve another time update (by Δ_2).
- ③ For the call in cycle n+2, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle n+2. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle n+1.

Example

The following example shows how the instruction works:



13.3.3.6 EV1oo2DI: 1oo2 evaluation with discrepancy analysis (STEP 7 Safety Advanced V11)

Description

This instruction implements a 1oo2 evaluation of two single-channel sensors combined with a discrepancy analysis.

Output Q is set to 1, if the signal states of inputs IN1 and IN2 both equal 1 and no discrepancy error DISC_FLT is stored. If the signal state of one or both inputs is 0, output Q is set to 0.

As soon as the signal states of inputs IN1 and IN2 are different, the discrepancy time DISCTIME is started. If the signal states of the two inputs are still different once the discrepancy time expires, a discrepancy error is detected and DISC_FLT is set to 1 (restart inhibit).

If the discrepancy between inputs IN1 and IN2 is no longer detected, the discrepancy error is acknowledged according to the parameter assignment of ACK_NEC:

- If ACK_NEC = 0, the acknowledgment is automatic.
- If ACK_NEC = 1, you must use a rising edge at input ACK to acknowledge the discrepancy error.

ACK_REQ = 1 signals that a user acknowledgment at input ACK is necessary to acknowledge the discrepancy error (cancel the restart inhibit). The instruction sets ACK_REQ = 1 as soon as discrepancy is no longer detected. After acknowledgment or if, prior to acknowledgment, there is once again a discrepancy between inputs IN1 and IN2, the instruction resets ACK_REQ to 0.

Output Q can never be set to 1 if the discrepancy time setting is < 0 or > 60 s. In this case, output DISC_FLT is also set to 1 (restart inhibit). The call interval of the safety program (e.g., OB35) must be less than the discrepancy time setting.

Every call of the "1oo2 evaluation with discrepancy analysis" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., EV1oo2DI_DB_1) or a multi-instance (e.g., EV1oo2DI_Instance_1) for the "1oo2 evaluation with discrepancy analysis" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

WARNING

The ACK_NEC tag must not be assigned a value of 0 unless an automatic restart of the affected process is otherwise excluded. (S033)

⚠ WARNING

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction")
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision. (S034)

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN1	Input	BOOL	Sensor 1
IN2	Input	BOOL	Sensor 2
DISCTIME	Input	TIME	Discrepancy time (0 to 60 s)
ACK_NEC	Input	BOOL	1 = acknowledgment necessary for discrepancy error
ACK	Input	BOOL	Acknowledgment of discrepancy error
Q	Output	BOOL	Output
ACK_REQ	Output	BOOL	1 = acknowledgement required
DISC_FLT	Output	BOOL	1 = discrepancy error
DIAG	Output	BYTE	Service information

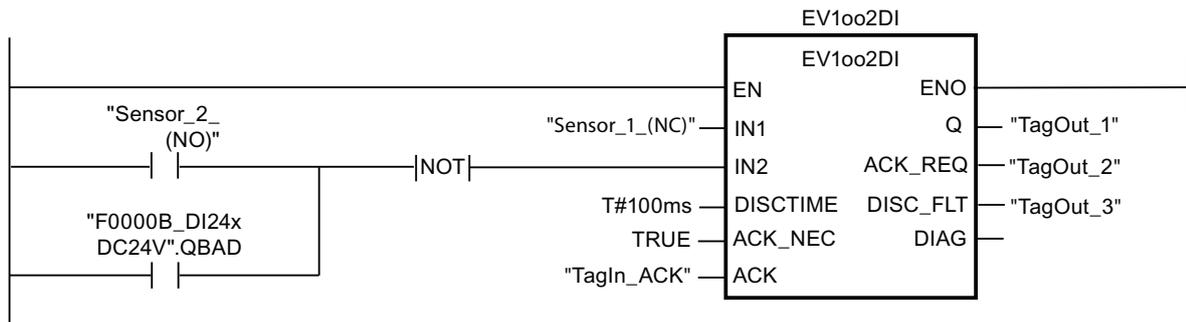
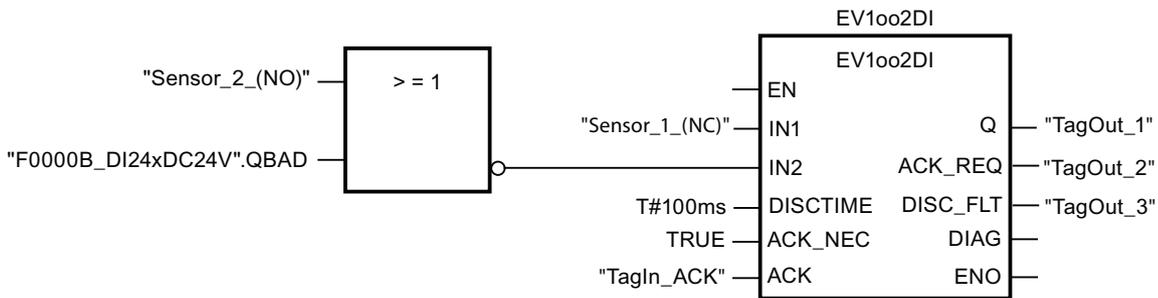
Activating inputs IN1 and IN2

Inputs IN1 and IN2 must both be activated in such a way that their safe state is 0.

Example

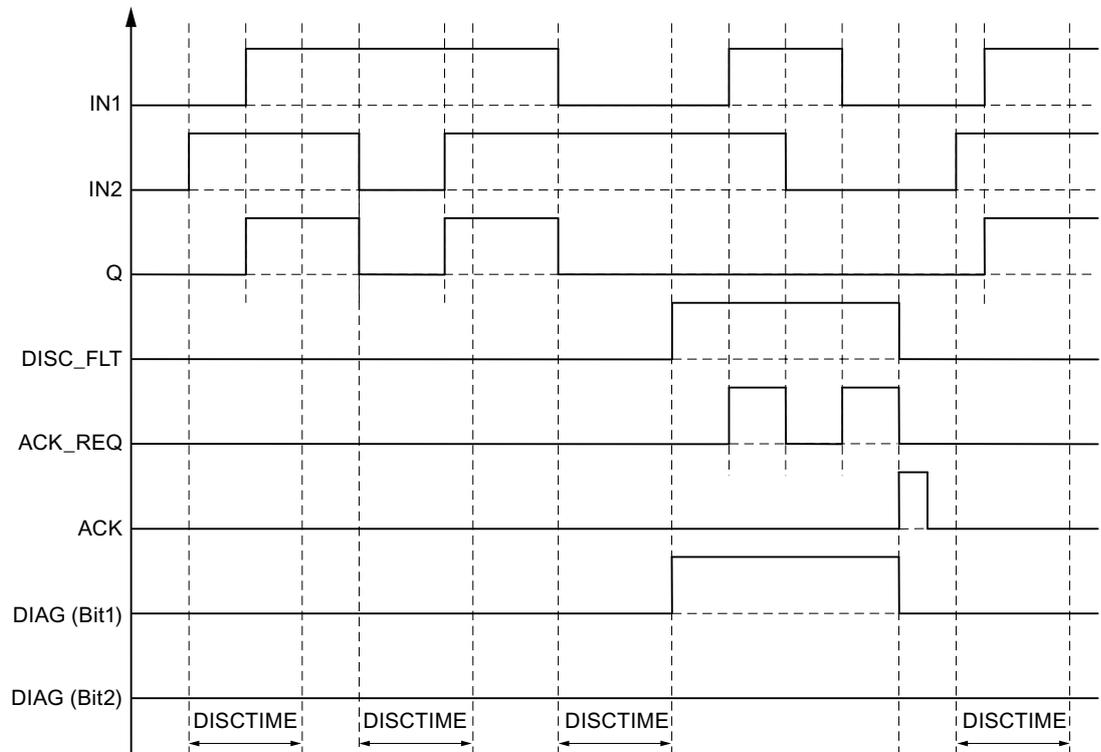
For nonequivalent signals, you have to invert the input (IN1 or IN2) to which you have assigned the sensor signal with a safe state of 1. You must also OR the sensor signal with the QBAD or QBAD_I_xx tag of the associated F-I/O DB or channel, so that a signal state of 0 is present at input IN1 or IN2 (after inversion) if fail-safe values are output.

Network1: EV1oo2DI with nonequivalent signals



Timing diagrams EV1oo2DI

If ACK_NEC = 1:



Startup characteristics

Note

If the sensors at inputs IN1 and IN2 are assigned to different F-I/O, it is possible that the fail-safe values are output for different lengths of time following startup of the F-system due to different startup characteristics of the F-I/O. If the signal states of inputs IN1 and IN2 remain different after the discrepancy time DISCTIME has expired, a discrepancy error is detected after the F-system starts up.

If ACK_NEC = 1 you must acknowledge the discrepancy error with a rising edge at input ACK.

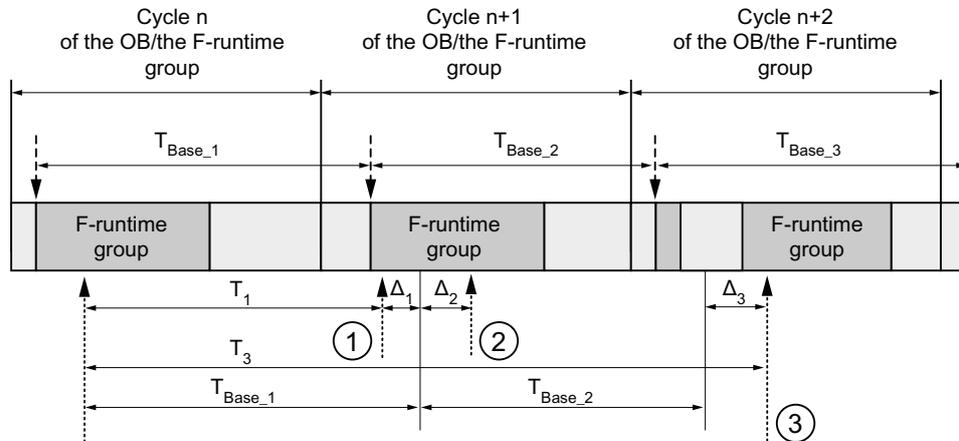
Output DIAG

The DIAG output provides non-fail-safe information on errors for service purposes. You can read out this information by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program. DIAG bits are saved until acknowledgment at input ACK.

Structure of DIAG

Bit No.	Assignment	Possible error causes	Remedies
Bit 0	Discrepancy error or incorrect discrepancy time setting (= status of DISC_FLT)	Sensor defective	Check sensors
		Wiring fault	Check wiring of sensors
		Sensors are wired to different F-I/O, and F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON on an F-I/O	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
		Discrepancy time setting is too low	If necessary, set a higher discrepancy time
		Discrepancy time setting is < 0 s or > 60 s	Set discrepancy time in range between 0 s and 60 s
Bit 1	For discrepancy errors: last signal state change was at input IN1	—	—
Bit 2	For discrepancy errors: last signal state change was at input IN2	—	—
Bit 3	Reserved	—	—
Bit 4	Reserved	—	—
Bit 5	For discrepancy errors: input ACK has a permanent signal state of 1	Acknowledgment button defective	Replace acknowledgment button
		Wiring fault	Check wiring of acknowledgment button
Bit 6	Acknowledgment necessary	—	—
Bit 7	State of output Q	—	—

Timing imprecision resulting from the update time of the time base used in the instruction:



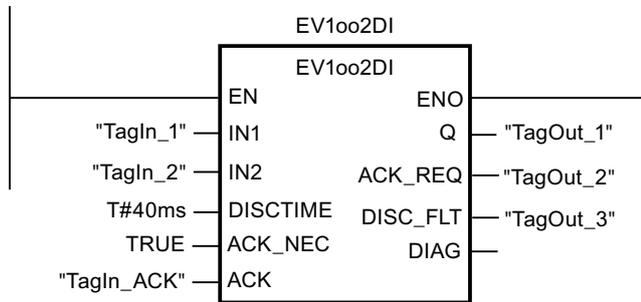
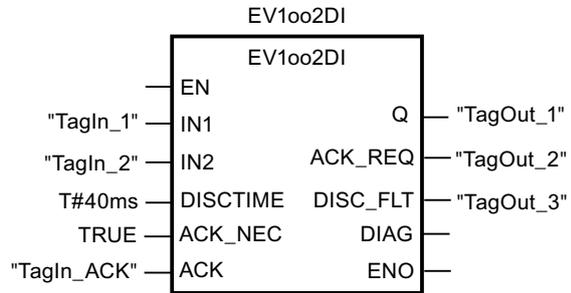
-----> = Time base update

.....> = Call time of an instruction with time processing

- ① For the first call in cycle n+1, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle n+1 are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle n+1. This does not involve another time update (by Δ_2).
- ③ For the call in cycle n+2, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle n+2. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle n+1.

Example

The following example shows how the instruction works:



13.3.3.7 FDBACK: Feedback monitoring (STEP 7 Safety Advanced V11)

Description

This instruction implements feedback monitoring.

To do this, the signal state of the output Q is checked for equality with the inverse signal state of the feedback input FEEDBACK.

Output Q is set to 1 as soon as input ON = 1. Requirement for this is that the feedback input FEEDBACK = 1 and no feedback error is saved.

Output Q is reset to 0, as soon as input ON = 0 or if a feedback error is detected.

A feedback error ERROR = 1 is detected if the inverse signal state of the feedback input FEEDBACK (to input Q) does not follow the signal state of output Q within the maximum tolerable feedback time. The feedback error is saved.

If a discrepancy is detected between the feedback input FEEDBACK and the output Q after a feedback error, the feedback error is acknowledged in accordance with the parameter assignment of ACK_NEC:

- If ACK_NEC = 0, the acknowledgment is automatic.
- If ACK_NEC = 1, you must acknowledge the feedback error with a rising edge at input ACK.

The ACK_REQ = 1 output then signals that a user acknowledgment is necessary at input ACK to acknowledge the feedback error. Following an acknowledgment, the instruction resets ACK_REQ to 0.

To avoid a feedback error from being detected and an acknowledgment from being required when the F-I/O controlled by output Q are passivated, you must supply input QBAD_FIO with the QBAD or QBAD_O_xx tag of the associated F-I/O.

Every call of the "Feedback monitoring" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., FDBACK_DB_1) or a multi-instance (e.g., FDBACK_Instance_1) for the "Feedback monitoring" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

WARNING

The ACK_NEC tag must not be assigned a value of 0 unless an automatic restart of the affected process is otherwise excluded. (S033)

⚠ WARNING

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction")
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

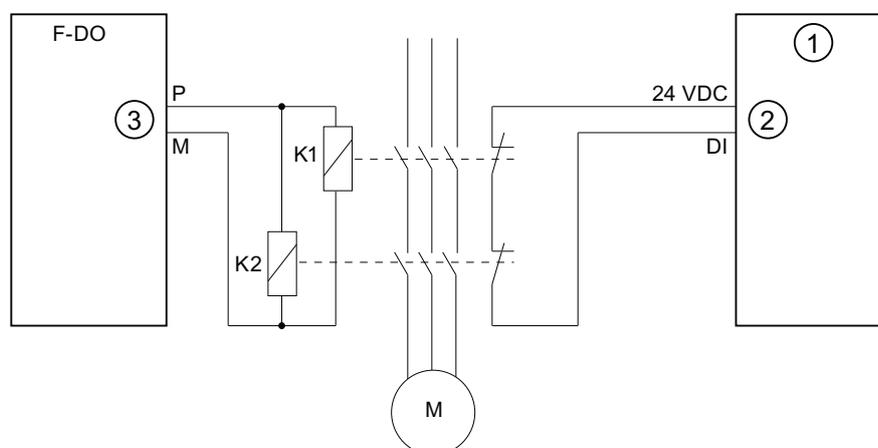
You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision (S034).

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
ON	Input	BOOL	1= Enable output
FEEDBACK	Input	BOOL	Feedback input
QBAD_FIO	Input	BOOL	QBAD or QBAD_O_xx signal of F-I/O/channel of output Q (F-I/O DB)
ACK_NEC	Input	BOOL	1=Acknowledgment necessary
ACK	Input	BOOL	Acknowledgment
FDB_TIME	Input	TIME	Feedback time
Q	Output	BOOL	Output
ERROR	Output	BOOL	Feedback error
ACK_REQ	Output	BOOL	Acknowledgment request
DIAG	Output	BYTE	Service information

Interconnection example



- ① Standard DI
- ② Input FEEDBACK
- ③ Output Q

The feedback contact is wired to a standard I/O module.

Instruction versions

Two versions are available for this instruction:

- Version 1.0

When projects that were created with *S7 Distributed Safety V5.4 SP5* are migrated, Version 1.0 of the instruction is used automatically.

Version 1.0 requires that the F_TOF block with the number FB 186 is available in the project tree in the "Program blocks/System blocks/STEP 7 Safety" folder.

If you want to compile a migrated safety program with *STEP 7 Safety Advanced V11* for the first time, we recommend that you update the version of the FBACK instruction to Version 1.1 beforehand. You will then avoid number conflicts.

- Version 1.1

When a new project is created with *STEP 7 Safety Advanced V11*, Version V1.1 is preset automatically. This version is functionally identical to Version V1.0, but does not require the F_TOF block to have a particular number.

For more information on the use of instruction versions, refer to the help on *STEP 7 Professional* under "Using instruction versions".

Startup characteristics

After an F-system startup, the instruction does not have to be acknowledged when no errors are present.

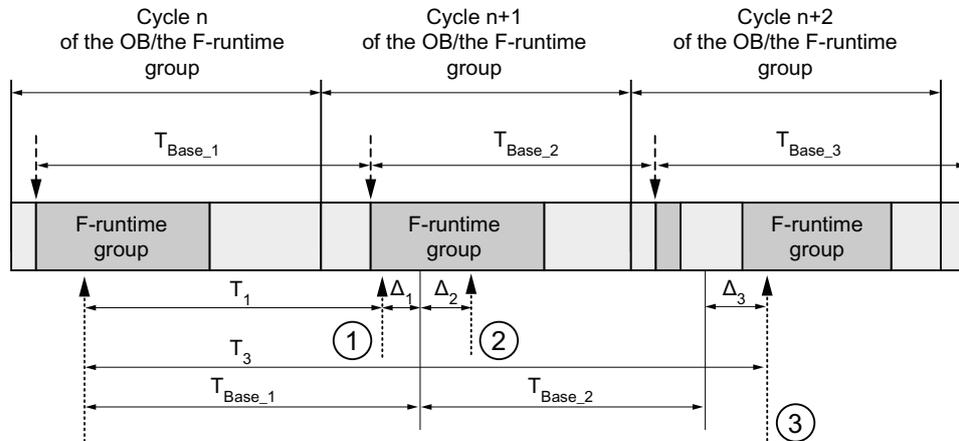
Output DIAG

The DIAG output provides non-fail-safe information on errors for service purposes. You can read out this information by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program. DIAG bits 0, 2, and 5 are saved until acknowledgment at input ACK.

Structure of DIAG

Bit No.	Assignment	Possible error causes	Remedies
Bit 0	Feedback error or incorrect feedback time setting (= state of ERROR)	Feedback time setting < 0	Set feedback time > 0
		Feedback time setting is too low	If necessary, set a higher feedback time
		Wiring fault	Check wiring of actuator and feedback contact
		Actuator or feedback contact is defective	Check actuator and feedback contact
		I/O fault or channel fault of feedback input	Check I/O
Bit 1	Passivation of F-I/O/channel controlled by output Q (= state of QBAD_FIO)	F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON of F-I/O	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
Bit 2	After feedback error: feedback input has permanent signal state of 0	I/O fault or channel fault of feedback input	Check I/O
		Feedback contact is defective	Check feedback contact
		F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON of F-I/O of feedback input	For remedy, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
Bit 3	Reserved	—	—
Bit 4	Reserved	—	—
Bit 5	For feedback error: input ACK has a permanent signal state of 1	Acknowledgment button defective	Check acknowledgment button
		Wiring fault	Check wiring of acknowledgment button
Bit 6	Acknowledgement required (= state of ACK_REQ)	—	—
Bit 7	State of output Q	—	—

Timing imprecision resulting from the update time of the time base used in the instruction:



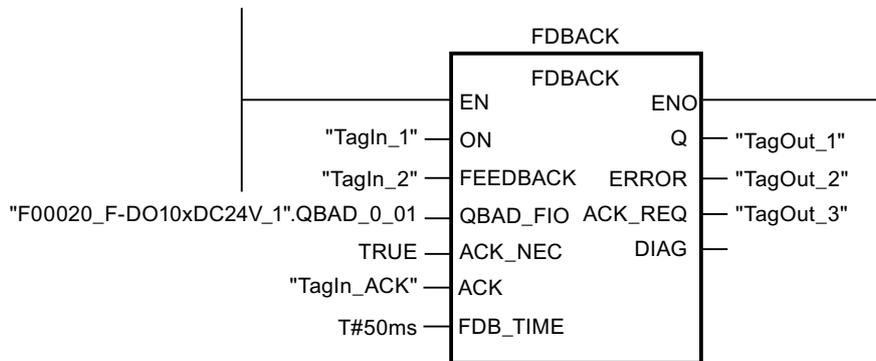
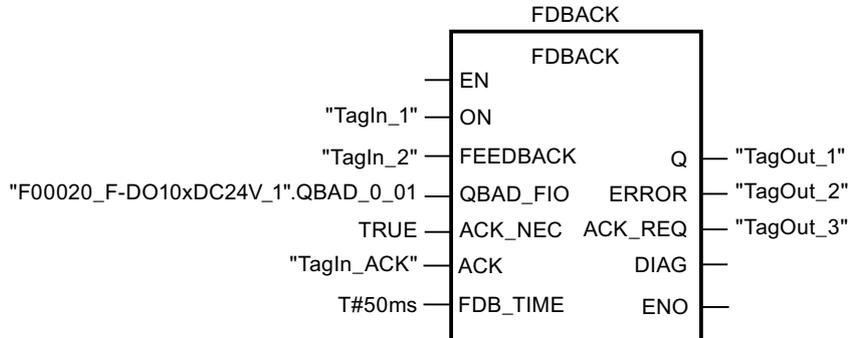
-----> = Time base update

.....> = Call time of an instruction with time processing

- ① For the first call in cycle n+1, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle n+1 are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle n+1. This does not involve another time update (by Δ_2).
- ③ For the call in cycle n+2, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle n+2. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle n+1.

Example

The following example shows how the instruction works:



13.3.3.8 SFDOOR: Safety door monitoring (STEP 7 Safety Advanced V11)

Description

This instruction implements safety door monitoring.

Enable signal Q is reset to 0 as soon as one of the inputs IN1 or IN2 take a signal state of 0 (safety door is opened). The enable signal can be reset to 1, only if:

- Inputs IN1 and IN2 both take a signal state of 0 prior to opening the door (safety door has been completely opened)
- Inputs IN1 and IN2 then both take a signal state of 1 (safety door is closed)
- An acknowledgment occurs

The acknowledgment for the enable takes place according to the parameter assignment at input ACK_NEC:

- If ACK_NEC = 0, the acknowledgment is automatic.
- If ACK_NEC = 1, you must use a rising edge at input ACK for acknowledging the enable.

Output ACK_REQ = 1 is used to signal that a user acknowledgment is required at input ACK for the acknowledgment. The instruction sets ACK_REQ = 1 as soon as the door is closed. Following an acknowledgment, the instruction resets ACK_REQ to 0.

In order for the instruction to recognize whether inputs IN1 and IN2 are 0 merely due to passivation of the associated F-I/O, you must supply inputs QBAD_IN1 or QBAD_IN2 with the QBAD or QBAD_I_xx tag of the associated F-I/O or channel. Among other things, this will prevent you from having to open the safety door completely prior to an acknowledgment in the event the F-I/O are passivated.

Every call of the "Safety door monitoring" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., SFDOOR_DB_1) or a multi-instance (e.g., SFDOOR_Instance_1) for the "Safety door monitoring" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

WARNING

The ACK_NEC tag must not be assigned a value of 0 unless an automatic restart of the affected process is otherwise excluded. (S033)

Parameters

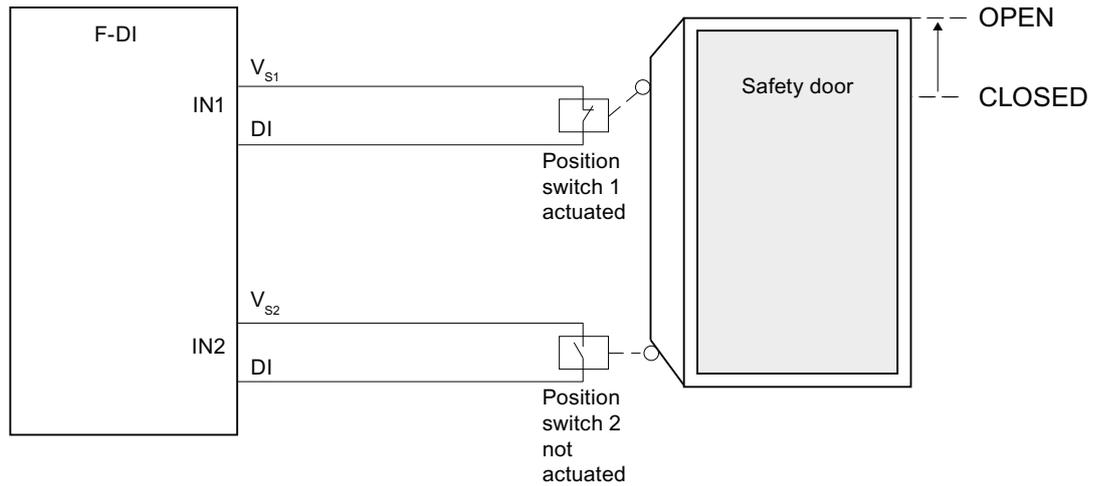
The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN1	Input	BOOL	Input 1
IN2	Input	BOOL	Input 2
QBAD_IN1	Input	BOOL	QBAD or QBAD_I_xx signal of F-I/O/channel of input IN1 (F-I/O)
QBAD_IN2	Input	BOOL	QBAD or QBAD_I_xx signal of F-I/O/channel of input IN2 (F-I/O)
OPEN_NEC	Input	BOOL	1= Open necessary at startup
ACK_NEC	Input	BOOL	1=Acknowledgment necessary
ACK	Input	BOOL	Acknowledgment
Q	Output	BOOL	1= Enable, safety door closed
ACK_REQ	Output	BOOL	Acknowledgment request
DIAG	Output	BYTE	Service information

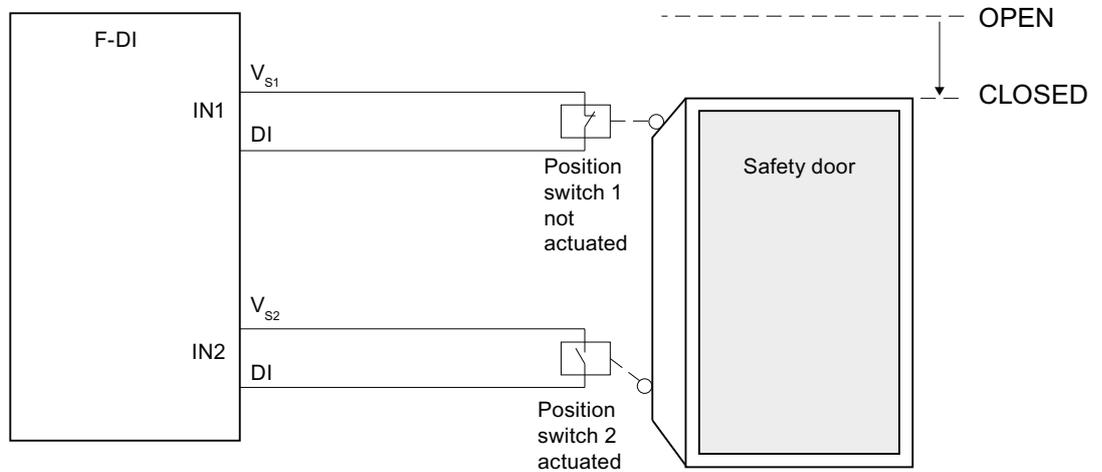
Interconnection example

You must interconnect the NC contact of position switch 1 of the safety door at input IN1 and the NO contact of position switch 2 at input IN2. Position switch 1 must be mounted in such a way that it is positively operated when the safety door is open. Position switch 2 must be mounted in such a way that it is operated when the safety door is closed.

Safety door open:



Safety door closed:



Startup characteristics

After an F-system startup, enable signal Q is reset to 0. The acknowledgment for the enable takes place according to the parameter assignment at inputs OPEN_NEC and ACK_NEC:

- When OPEN_NEC = 0, an automatic acknowledgement occurs **independently** of ACK_NEC, as soon as the two inputs IN1 and IN2 take signal state 1 for the first time following reintegration of the associated F-I/O (safety door is closed).
- When OPEN_NEC = 1 or if at least one of the IN1 and IN2 inputs still has a signal state of 0 after reintegration of the associated F-I/O, an automatic acknowledgment occurs **according** to ACK_NEC or you have to use a rising edge at input ACK for the enable. Prior to acknowledgment, inputs IN1 and IN2 both have to take a signal state of 0 (safety door has been completely opened) followed by a signal state of 1 (safety door is closed).

 **WARNING**

The OPEN_NEC tag must not be assigned a value of 0 unless an automatic restart of the affected process is otherwise excluded. (S039)

Output DIAG

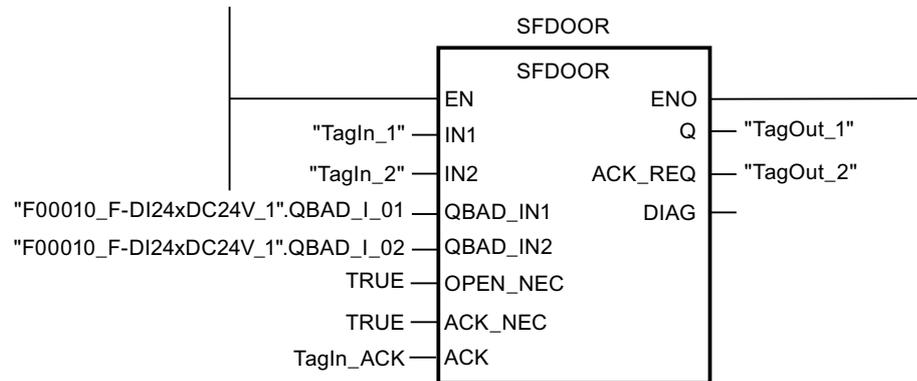
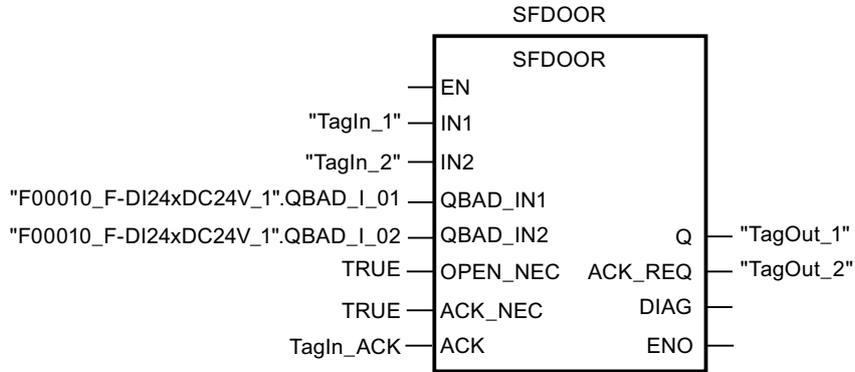
The DIAG output provides non-fail-safe information on errors for service purposes. You can read out this information by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program.

Structure of DIAG

Bit No.	Assignment	Possible error causes	Remedies
Bit 0	Reserved	—	—
Bit 1	Signal state 0 is missing at both IN1 and IN2 inputs	Safety door was not completely opened when OPEN_NEC = 1 after F-system startup	Open safety door completely
		Open safety door was not completely opened	Open safety door completely
		Wiring fault	Check wiring of position switch
		Position switch is defective	Check position switch
		Position switch is incorrectly adjusted	Adjust position switch properly
Bit 2	Signal state 1 is missing at both IN1 and IN2 inputs	Safety door was not closed	Close safety door
		Wiring fault	Check wiring of position switch
		Position switch is defective	Check position switch
		Position switch is incorrectly adjusted	Adjust position switch properly
Bit 3	QBAD_IN1 and/or QBAD_IN2 = 1	F-I/O fault, channel fault, or communication error, or passivation by means of PASS_ON of F-I/O or channel of IN1 and/or IN2	For a solution, see section "Structure of DIAG", bits 0 to 6 under F-I/O DB (Page 82)
Bit 4	Reserved	—	—
Bit 5	If enable is missing: input ACK has a permanent signal state of 1	Acknowledgment button defective	Check acknowledgment button
		Wiring fault	Check wiring of acknowledgment button
Bit 6	Acknowledgement required (= state of ACK_REQ)	—	—
Bit 7	State of output Q	—	—

Example

The following example shows how the instruction works:



13.3.3.9 ACK_GL: Global acknowledgment of all F-I/O in an F-runtime group (STEP 7 Safety Advanced V11)

Description

This instruction creates an acknowledgment for the simultaneous reintegration of all F-I/O or channels of the F-I/O of an F-runtime group after communication errors, F-I/O errors, or channel faults.

A user acknowledgment (Page 99) with a positive edge at input ACK_GLOB is required for reintegration. The acknowledgement occurs analogously to the user acknowledgment via the ACK_REI tag of the F-I/O DB (Page 79), but it acts simultaneously on all F-I/O of the F-runtime group in which the instruction is called.

If you use the instruction ACK_GL, you do not have to provide for a user acknowledgment for each F-I/O of the F-runtime group via the ACK_REI tag of the F-I/O DB.

Every call of the "Global acknowledgment of all F-I/O of a runtime group" instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., ACK_GL_DB_1) or a multi-instance (e.g., ACK_GL_Instance_1) for the "Global acknowledgment of all F-I/O of a runtime group" instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
ACK_GLOB	Input	BOOL	1=acknowledgment for reintegration

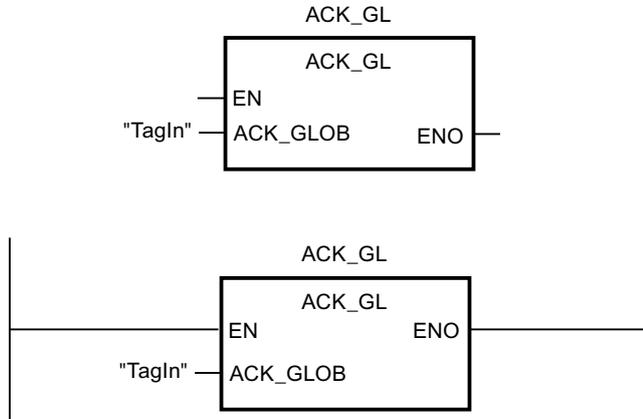
Note

An acknowledgment via the ACK_GL instruction is only possible if the tag ACK_REI of the F-I/O DB = 0. Accordingly, an acknowledgment via the tag ACK_REI of the F-I/O DB is only possible if the input ACK_GLOB of the instruction = 0.

The instruction is only allowed to be called once per F-runtime group.

Example

The following example shows how the instruction works:



13.3.4 Timer operations

13.3.4.1 TP: Generate pulse (STEP 7 Safety Advanced V11)

Description

You can use the "Generate pulse" instruction to set output Q for an assigned period. The instruction is started if the result of logic operation (RLO) changes from "0" to "1" (positive signal edge) at input IN. The assigned period PT starts running when the instruction starts. Output Q is set for period PT, regardless of the subsequent sequence of the input signal. Also the detection of a new positive signal edge does not influence the signal state at output Q as long as period PT runs.

The current time value can be queried at output ET. The time value begins at T#0s and ends when the value of period PT is reached. If period PT is reached and the signal state at input IN is "0", output ET is reset.

Every call of the "Generate pulse" instruction must be assigned a data area in which the instruction data are stored. The "Call options" dialog is automatically opened for this reason when the instruction is inserted in the program; in it you can create a data block (single instance) (e.g., F_IEC_Timer_DB_1) or a multi-instance (e.g., F_IEC_Timer_Instance_1) for the "Generate pulse" instruction. Once you have created the data block, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the *help on STEP 7 Professional*.

WARNING

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction").
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision. (S034)

The operating system resets the instances of the "Generate pulse" instruction on a startup of the F-system.

Note

The functionality of this instruction differs from the corresponding standard TP instruction in the following points:

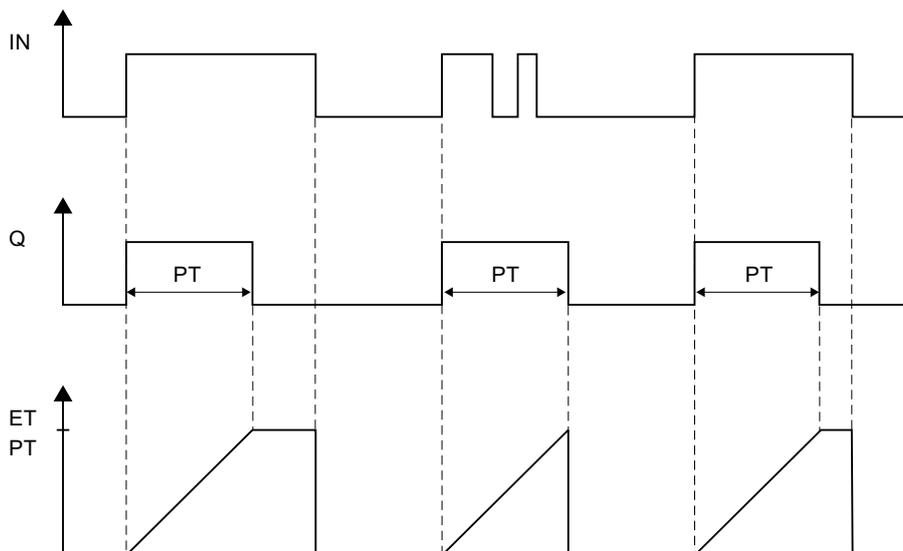
- When a call is made with PT = 0 ms, the TP instance is not reset completely (initialized). The instruction behaves in accordance with the timing diagrams: only outputs Q and ET are reset. To restart the pulse, a new rising signal edge at input IN is required once PT is greater than 0 again.
- A call with PT < 0 ms resets outputs Q and ET. To restart the pulse, a new rising signal edge at input IN is required once PT is greater than 0 again.

Parameters

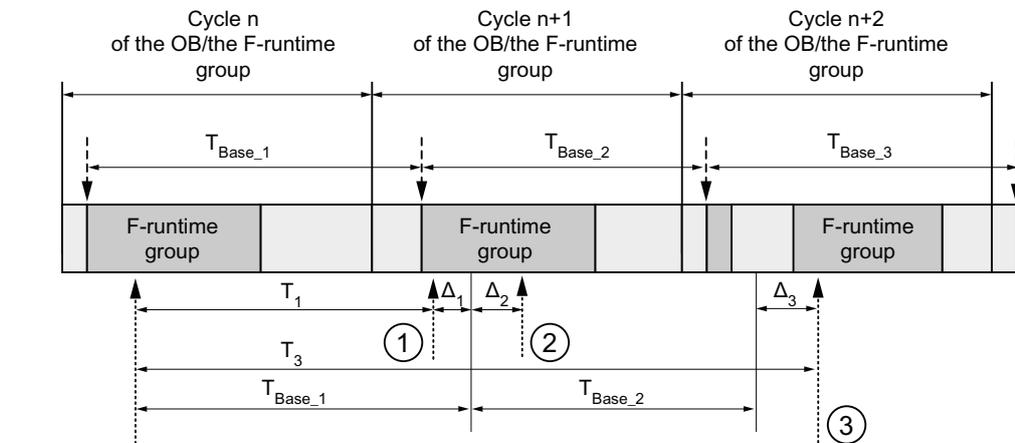
The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN	Input	BOOL	Start input
PT	Input	TIME	Duration of pulse; must be positive.
Q	Output	BOOL	Pulse output
ET	Output	TIME	Current time value

Timing diagrams TP



Timing imprecision resulting from the update time of the time base used in the instruction:

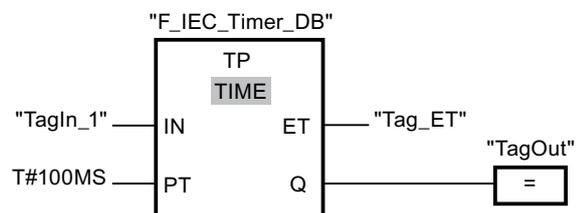


-----> = Time base update
> = Call time of an instruction with time processing

- ① For the first call in cycle $n+1$, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle $n+1$ are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle $n+1$. This does not involve another time update (by Δ_2).
- ③ For the call in cycle $n+2$, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle $n+2$. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle $n+1$.

Example

The following example shows how the instruction works:



If the signal state of operand "TagIn_1" changes from "0" to "1", the "Generate pulse" instruction is started and the period assigned at input PT (100 ms) runs, regardless of the further sequence of operand "TagIn_1".

Operand "TagOut" at output Q has signal state "1" as long as the period is running. Operand "Tag_ET" contains the current time value.

13.3.4.2 TON: Generate on-delay (STEP 7 Safety Advanced V11)

Description

You can use the "Generate on-delay" instruction to delay the setting of output Q by the assigned period PT. The "Generate on-delay" instruction is started when the result of logic operation (RLO) at input IN changes from "0" to "1" (positive signal edge). The assigned period PT starts running when the instruction starts. When period PT has expired, output Q is set to signal state "1". Output Q remains set as long as start input is set to signal state "1". When the signal state at the start input changes from "1" to "0", output Q is reset. The time function is restarted when a new positive signal edge is detected at the start input.

The current time value can be queried at output ET. The time value begins at T#0s and ends when the value of period PT is reached. Output ET is reset, as soon as the signal state at input IN changes to "0".

Every call of the "Generate on-delay" instruction must be assigned a data area in which the instruction data are stored. The "Call options" dialog is automatically opened for this reason when the instruction is inserted in the program; in it can create a data block (single instance) (e.g., F_IEC_Timer_DB_1) or a multi-instance (e.g., F_IEC_Timer_Instance_1) for the "Generate on-delay" instruction. Once you have created the data block, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the *help on STEP 7 Professional*.

WARNING

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction")
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision. (S034)

The operating system resets the instances of the "Generate on-delay" instruction on a startup of the F-system.

Note

The functionality of this instruction differs from the corresponding standard TON instruction in the following points:

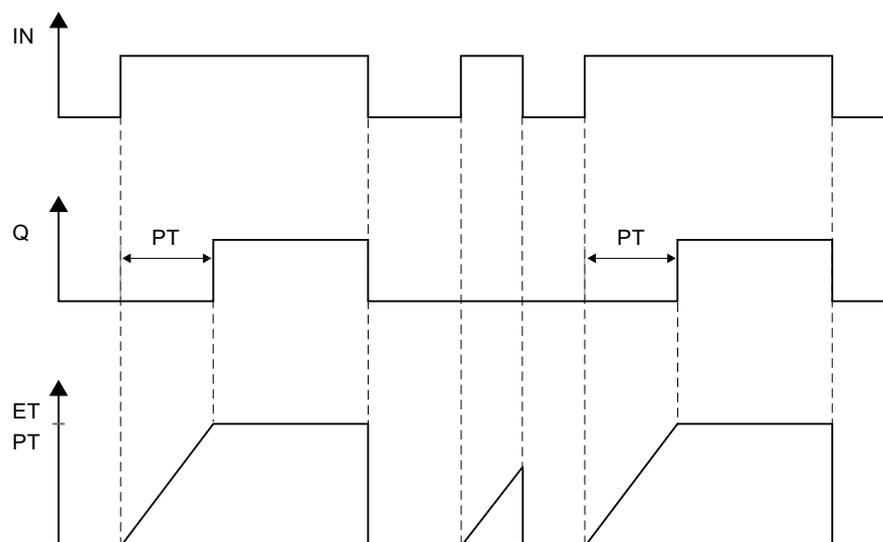
- When a call is made with $PT = 0$ ms, the instance of the TON is not reset completely (initialized). The instruction behaves in accordance with the timing diagrams: only output ET is reset. To restart the on-delay, a new rising signal edge at input IN is required once PT is greater than 0 again.
- A call with $PT < 0$ ms resets outputs Q and ET. To restart the on-delay, a new rising signal edge at input IN is required once PT is greater than 0 again.

Parameters

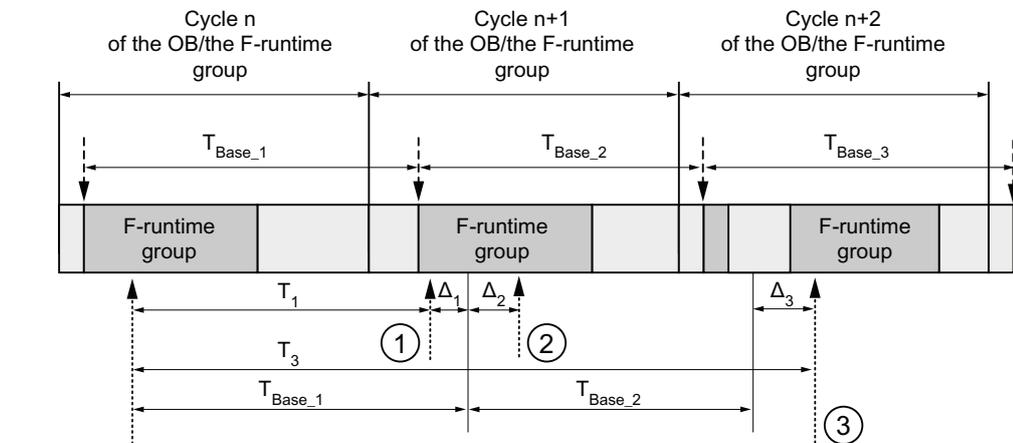
The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN	Input	BOOL	Start input
PT	Input	TIME	Duration of on-delay; must be positive.
Q	Output	BOOL	Output that is set after expiration of time PT.
ET	Output	TIME	Current time value

Pulse diagram



Timing imprecision resulting from the update time of the time base used in the instruction:



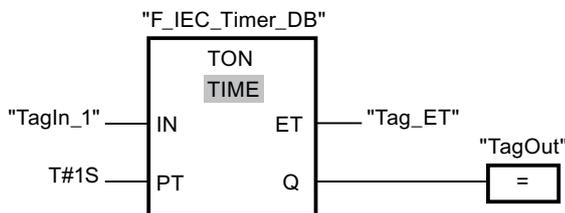
-----> = Time base update

.....> = Call time of an instruction with time processing

- ① For the first call in cycle n+1, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle n+1 are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle n+1. This does not involve another time update (by Δ_2).
- ③ For the call in cycle n+2, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle n+2. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle n+1.

Example

The following example shows how the instruction works:



When the signal state of operand "TagIn_1" changes from "0" to "1", the

"Generate on-delay" instruction is started and the period assigned at input PT (1 s) runs.

Operand "TagOut" at output Q is set to signal state "1" when the period has elapsed and remains set as long as operand "TagIn_1" still has signal state "1". Operand "Tag_ET" contains the current time value.

13.3.4.3 TOF: Generate off-delay (STEP 7 Safety Advanced V11)

Description

You can use the "Generate off-delay" instruction to delay resetting output Q by the assigned period PT. Output Q is set if the result of logic operation (RLO) changes from "0" to "1" (positive signal edge) at input IN. The assigned period PT starts when the signal state at input IN changes back to "0". Output Q remains set as long as period PT runs. After period PT expires, output Q is reset. If the signal state at input IN changes to "1" before period PT has expired, then the time is reset. The signal state at output Q remains at "1".

The current time value can be queried at output ET. The time value begins at T#0s and ends when the value of period PT is reached.

Every call of the "Generate off-delay" instruction must be assigned a data area in which the instruction data are stored. the "Call options" dialog is automatically opened for this reason when the instruction is inserted in the program; in it you can create a data block (single instance) (e.g., F_IEC_Timer_DB_1) or a multi-instance (e.g., F_IEC_Timer_Instance_1) for the "Generate off-delay" instruction. Once you have created the data block, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the *help on STEP 7 Professional*.

WARNING

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction")
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision. (S034)

The operating system resets the instances of the "Generate off-delay" instruction on a startup of the F-system.

Note

The functionality of this instruction differs from the corresponding standard TOF instruction in the following points:

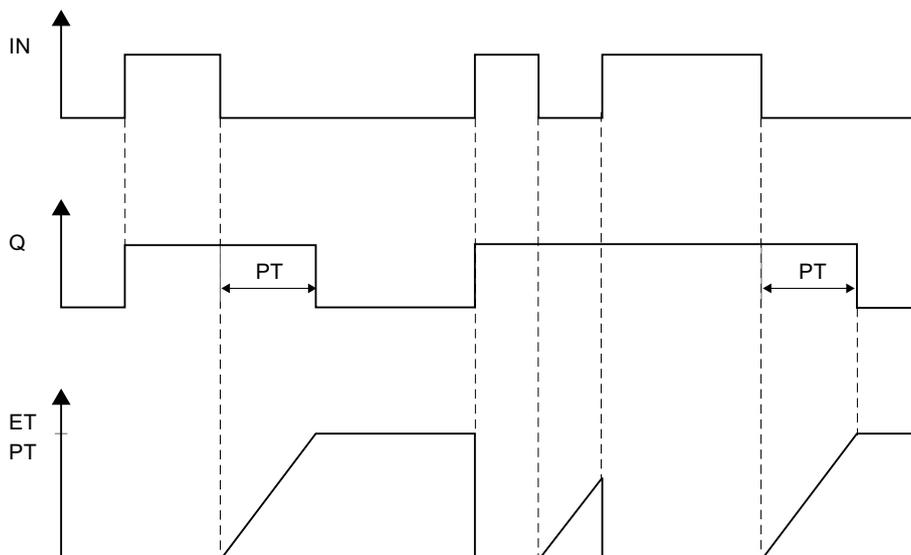
- When a call is made with PT = 0 ms, the instance of the TOF is not reset completely (initialized). The instruction behaves in accordance with the timing diagrams: only outputs Q and ET are reset. To restart the off-delay, another falling signal edge at input IN is required once PT is greater than 0 again.
- A call with PT < 0 ms resets outputs Q and ET. To restart the off-delay, another falling signal edge at input IN is required once PT is greater than 0 again.

Parameters

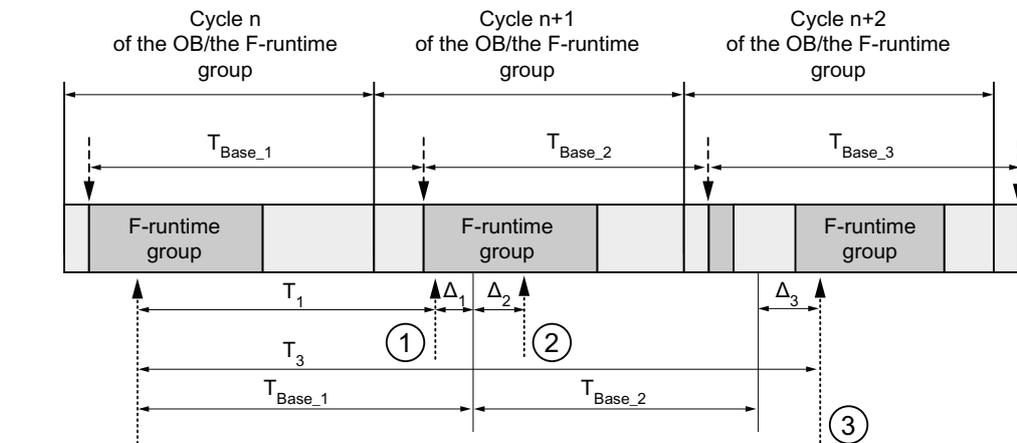
The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN	Input	BOOL	Start input
PT	Input	TIME	Duration of off-delay; PT must be positive.
Q	Output	BOOL	Output that is reset after expiration of time PT.
ET	Output	TIME	Current time value

Pulse diagram



Timing imprecision resulting from the update time of the time base used in the instruction:



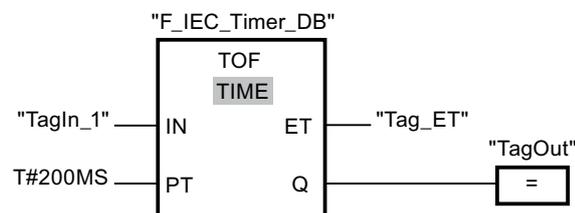
-----> = Time base update

.....> = Call time of an instruction with time processing

- ① For the first call in cycle n+1, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle n+1 are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle n+1. This does not involve another time update (by Δ_2).
- ③ For the call in cycle n+2, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle n+2. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle n+1.

Example

The following example shows how the instruction works:



If the signal state of operand "TagIn_1" changes from "0" to "1", the signal state of operand "TagOut" at output Q is set to "1".

If the signal state of operand "TagIn_1" changes back to "0", the period assigned at input PT (200 ms) runs.

The "TagOut" operand at output Q is set back to "0" when the period expires. Operand "Tag_ET" contains the current time value.

13.3.5 Counter operations

13.3.5.1 CTU: Count up (STEP 7 Safety Advanced V11)

Description

You can use the "Count up" instruction to increment the value at output CV. When the signal state at input CU changes from "0" to "1" (positive signal edge), the instruction is executed and the current count value at output CV is increased by one. The count value is increased on each detection of a positive signal edge until it reaches the high limit of the data type specified at output CV. When the high limit is reached, the signal state at input CU no longer affects the instruction.

The counter status can be queried at output Q. The signal state at output Q is determined by parameter PV. When the current count value is greater than or equal to the value of parameter PV, output Q is set to signal state "1". In all other cases, the signal state at output Q is "0".

The value at output CV is reset to zero when the signal state at input R changes to "1". As long as signal state "1" exists at input R, the signal state at input CU has no effect on the instruction.

Every call of the "Count up" instruction must be assigned a data area in which the instruction data are stored. The "Call options" dialog is automatically opened for this reason when the instruction is inserted in the program; in it you can create a data block (single instance) (e.g., F_IEC_Counter_DB_1) or a multi-instance (e.g., F_IEC_Counter_Instance_1) for the "Count up" instruction. Once you have created the data block, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the *help on STEP 7 Professional*.

The operating system resets the instances of the "Count up" instruction on a startup of the F-system.

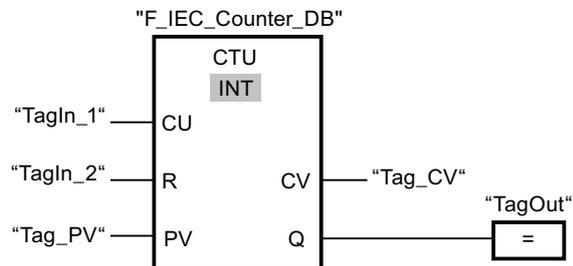
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
CU	Input	BOOL	Counter input
R	Input	BOOL	Reset input
PV	Input	INT	Value at which output Q is set
Q	Output	BOOL	Counter status
CV	Output	INT	Current count value

Example

The following example shows how the instruction works:



When the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count up" instruction is executed and the current count value of the "Tag_CV" operand is increased by one. The count value is increased on every additional positive signal edge until the high limit of the specified data type (32767) is reached.

The value at parameter PV is used as the limit for the determination of output "TagOut". Output "TagOut" has the signal state "1" as long as the current count value is greater than or equal to the value of operand "Tag_PV". In all other cases, output TagOut has signal state "0".

13.3.5.2 CTD: Count down (STEP 7 Safety Advanced V11)

Description

You can use the "Count down" instruction to decrement the value at output CV. When the signal state at input CD changes from "0" to "1" (positive signal edge), the instruction is executed and the current count value at output CV is decreased by one. The count value is decreased on each detection of a positive signal edge until it reaches the low limit of the specified data type. When the low limit is reached, the signal state at input CD no longer affects the instruction.

The counter status can be queried at output Q. When the current count value is less than or equal to zero, output Q is set to signal state "1". In all other cases, the signal state at output Q is "0".

The value at output CV is set to the value of parameter "PV" when the signal state at input LD changes to "1". As long as signal state "1" exists at input LD, the signal state at input CD has no effect on the instruction.

Every call of the "Count down" instruction must be assigned a data area in which the instruction data are stored. The "Call options" dialog is automatically opened for this reason when the instruction is inserted in the program; in it you can generate a data block (single instance) (e.g., F_IEC_Counter_DB_1) or a multi-instance (e.g., F_IEC_Counter_Instance_1) for the "Count down" instruction. Once you have created the data block, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the *help on STEP 7 Professional*.

The operating system resets the instances of the "Count down" instruction on a startup of the F-system.

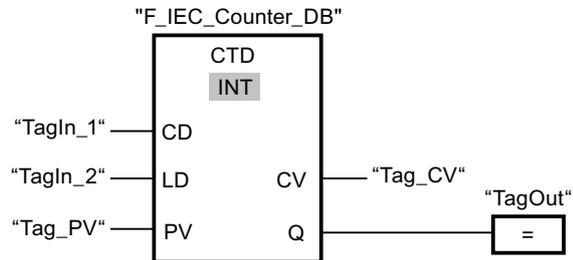
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
CD	Input	BOOL	Counter input
LD	Input	BOOL	Load input
PV	Input	INT	Value for which output Q is set
Q	Output	BOOL	Counter status
CV	Output	INT	Current count value

Example

The following example shows how the instruction works:



If the signal state of the "TagIn_1" operand changes from "0" to "1", the "Count down" instruction is executed and the current count value at output "Tag_CV" is decreased by one. The count value is decreased on each additional positive signal edge until the low limit of the specified data type (-32768) is reached.

Output "TagOut" has the signal state "1" as long as the current count value is less than or equal to zero. In all other cases, output TagOut has signal state "0".

13.3.5.3 CTUD: Count up and down (STEP 7 Safety Advanced V11)

Description

You can use the "Count up and down" instruction to increment and decrement the count value at output CV. If the signal state at input CU changes from "0" to "1" (positive signal edge), the current count value at output CV is increased by one. If the signal state at input CD changes from "0" to "1" (positive signal edge), the count value at output CV is decreased by one. If a positive signal edge is present at inputs CU and CD in one program cycle, the current count value at output CV remains unchanged.

The count value can be increased until it reaches the high limit of the data type specified at output CV. When the high limit is reached, the count value is no longer incremented on a positive signal edge. When the low limit of the specified data type is reached, the count value is no longer decremented.

When the signal state at input LD changes to "1", the count value at output CV is set to the value of parameter PV. As long as signal state "1" exists at input LD, the signal state at inputs CU and CD has no effect on the instruction.

The count value is set to zero, when the signal state at input R changes to "1". As long as signal state "1" exists at input R, the signal state at inputs CU, CD, and LD has no effect on the "Count up and down" instruction.

The status of the up counter can be queried at output QU. When the current count value is greater than or equal to the value of parameter PV, output QU has signal state "1". In all other cases, the signal state at output QU is "0".

The status of the down counter can be queried at output QD. When the current count value is lesser than or equal to zero, output QD delivers signal state "1". In all other cases, the signal state at output QD is "0".

Every call of the "Count up and down" instruction must be assigned a data area in which the instruction data are stored. The "Call options" dialog is automatically opened for this reason when the instruction is inserted in the program; in it you can create a data block (single instance) (e.g., F_IEC_Counter_DB_1) or a multi-instance (e.g., F_IEC_Counter_Instance_1) for the "Count up and down" instruction. Once you have created the data block, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the *help on STEP 7 Professional*.

The operating system resets the instances of the "Count up and down" instruction on a startup of the F-system.

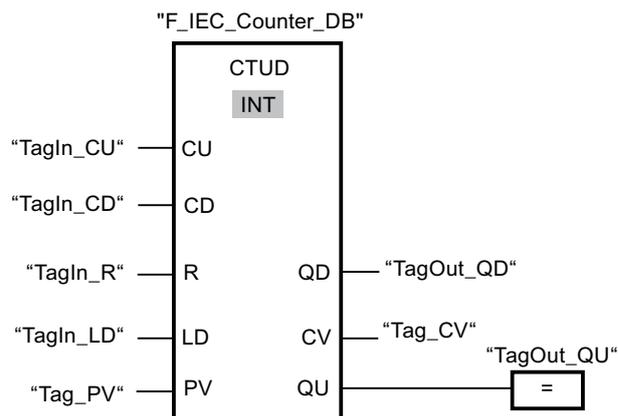
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
CU	Input	BOOL	Count up input
CD	Input	BOOL	Count down input
R	Input	BOOL	Reset input
LD	Input	BOOL	Load input
PV	Input	INT	Value for which output QU is set
QU	Output	BOOL	Status of up counter
QD	Output	BOOL	Status of down counter
CV	Output	INT	Current count value

Example

The following example shows how the instruction works:



If the signal state at input "TagIn_CU" or at input "TagIn_CD" changes from "0" to "1" (positive signal edge), the "Count up and down" instruction is executed. When a positive signal edge at input "TagIn_CU" is detected, the current count value of the "Tag_CV" operand is increased by one. When a positive signal edge is detected at input "TagIn_CD", the current count value at output "Tag_CV" is decreased by one. The count value is increased on each positive signal edge at input CU until it reaches the high limit of 32767. The count value is decreased on each positive signal edge at input CD until it reaches the low limit of -32768.

Output "TagOut_GU" delivers the signal state "1" as long as the current count value is greater than or equal to the value at input "Tag_PV". In all other cases, output TagOut_QU has signal state "0".

Output "TagOut_QD" delivers the signal state "1" as long as the current count value is less than or equal to zero. In all other cases, output TagOut_QD has signal state "0".

13.3.6 Comparator operations

13.3.6.1 CMP ==: Equal (STEP 7 Safety Advanced V11)

Description

Using the "Equal" instruction you can query whether the value at input IN1 is equal to the value at input IN2.

If the condition of the comparison is fulfilled, the instruction returns result of logic operation (RLO) "1". If the condition is not fulfilled, the instruction returns RLO "0".

Parameters

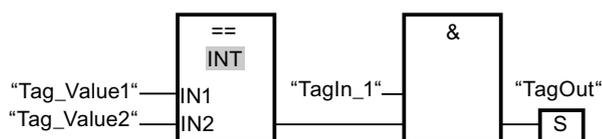
The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN1	Input	INT, DINT, TIME, WORD, DWORD	First value to compare
IN2	Input	INT, DINT, TIME, WORD, DWORD	Second value to compare

You can select the data type of the instruction in the "<???"> drop-down list in the instruction box.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- Operand "TagIn_1" has signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" = "Tag_Value2").

13.3.6.2 CMP <>: Not equal (STEP 7 Safety Advanced V11)

Description

Using the "Not equal" instruction you can query whether the value at input IN1 is not equal to the value at input IN2.

If the condition of the comparison is fulfilled, the instruction returns result of logic operation (RLO) "1". If the condition is not fulfilled, the instruction returns RLO "0".

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN1	Input	INT, DINT, TIME, WORD, DWORD	First value to compare
IN2	Input	INT, DINT, TIME, WORD, DWORD	Second value to compare

You can select the data type of the instruction in the "<???" drop-down list in the instruction box.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- Operand "TagIn_1" has signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" <> "Tag_Value2").

13.3.6.3 CMP >=: Greater than or equal (STEP 7 Safety Advanced V11)

Description

Using the "Greater or equal" instruction you can query whether the value at input IN1 is greater than or equal to the value at input IN2. Both comparison values must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns result of logic operation (RLO) "1". If the condition of the comparison is not fulfilled, the instruction returns RLO "0".

Parameters

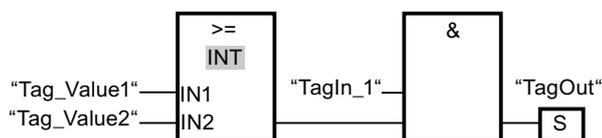
The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN1	Input	INT, DINT, TIME	First value to compare
IN2	Input	INT, DINT, TIME	Second value to compare

You can select the data type of the instruction in the "<???" drop-down list in the instruction box.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- Operand "TagIn_1" has signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" >= "Tag_Value2").

13.3.6.4 CMP <=: Less than or equal (STEP 7 Safety Advanced V11)

Description

Using the "Less or equal" instruction you can query whether the value at input IN1 is less than or equal to the value at input IN2. Both comparison values must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns result of logic operation (RLO) "1". If the condition of the comparison is not fulfilled, the instruction returns RLO "0".

Parameters

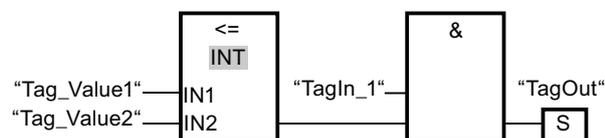
The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN1	Input	INT, DINT, TIME	First value to compare
IN2	Input	INT, DINT, TIME	Second value to compare

You can select the data type of the instruction in the "<???" drop-down list in the instruction box.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- Operand "TagIn_1" has signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" <= "Tag_Value2").

13.3.6.5 CMP >: Greater than (STEP 7 Safety Advanced V11)

Description

Using the "Greater than" instruction you can query whether the value at input IN1 is greater than the value at input IN2. Both comparison values must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns result of logic operation (RLO) "1". If the condition of the comparison is not fulfilled, the instruction returns RLO "0".

Parameters

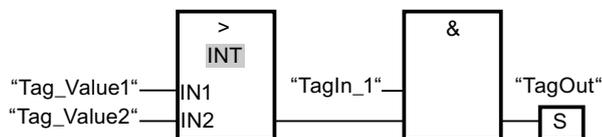
The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN1	Input	INT, DINT, TIME	First value to compare
IN2	Input	INT, DINT, TIME	Second value to compare

You can select the data type of the instruction in the "<???">" drop-down list in the instruction box.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- Operand "TagIn_1" has signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" > "Tag_Value2").

13.3.6.6 CMP <: Less than (STEP 7 Safety Advanced V11)

Description

Using the "Less than" instruction you can query whether the value at input IN1 is less than the value at input IN2. Both comparison values must be of the same data type.

If the condition of the comparison is fulfilled, the instruction returns result of logic operation (RLO) "1". If the condition of the comparison is not fulfilled, the instruction returns RLO "0".

Parameters

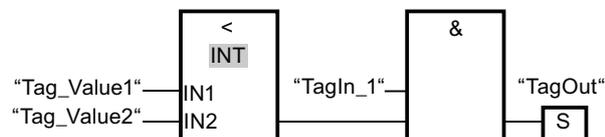
The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN1	Input	INT, DINT, TIME	First value to compare
IN2	Input	INT, DINT, TIME	Second value to compare

You can select the data type of the instruction in the "<???" drop-down list in the instruction box.

Example

The following example shows how the instruction works:



Output "TagOut" is set when the following conditions are fulfilled:

- Operand "TagIn_1" has signal state "1".
- The condition of the comparison instruction is fulfilled ("Tag_Value1" < "Tag_Value2").

13.3.7 Math functions

13.3.7.1 ADD: Add (STEP 7 Safety Advanced V11)

Description

You can use the "Add" instruction to add the value at input IN1 and the value at input IN2 and query the sum at the OUT output ($OUT = IN1 + IN2$).

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Note

If the result of the instruction is outside the range permitted for the data type, then the F-CPU goes to STOP when the result is sent to an output at an F-I/O or to a partner F-CPU via safety-related CPU-CPU communication.

One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"
- "Safety program: internal CPU fault; internal error information: 404"

You must therefore ensure that the permitted range for the data type is observed when creating the program!

If this is not possible, you can avoid a STOP of the F-CPU by inserting a "Get status bit OV" instruction in the next network, thereby programming overflow detection into it.

Note the following:

- The result of the instruction behaves like the analogous instruction in a standard block.
 - The network with the "Get status bit OV" instruction must not contain any jump labels.
 - The execution time of the instruction is extended (see also Excel file for response time calculation (<http://support.automation.siemens.com/WW/view/en/49368678/134200>)).
 - A warning is issued if you do not insert a "Get status bit OV" instruction.
-

Parameters

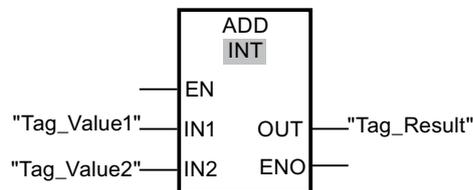
Parameter	Declaration	Data type	Description
IN1	Input	INT, DINT	First addend
IN2	Input	INT, DINT	Second addend
OUT	Output	INT, DINT	Sum

You can select the data type of the instruction in the "<???">" drop-down list in the instruction box.

Example

The following example shows how the instruction works:

Network 1:



Network 2:



The "Add" instruction is always run (regardless of the signal state at enable input EN).

The value of the "Tag_Value1" operand is added to value of the Tag_Value2 operand. The result of the addition is stored in the "Tag_Result" operand.

If an overflow occurs during execution of the "Add" instruction, the status bit OV is set to "1". In network 2, following the query of the status bit OV, the "Set output" (S) instruction is executed and the "TagOut" operand is set.

See also

OV: Get status bit OV (STEP 7 Safety Advanced V11) (Page 509)

13.3.7.2 SUB: Subtract (STEP 7 Safety Advanced V11)

Description

You can use the "Subtract" instruction to subtract the value at input IN2 from the value at input IN1 and query the difference at the OUT output ($OUT = IN1 - IN2$).

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Note

If the result of the instruction is outside the range permitted for the data type, then the F-CPU goes to STOP when the result is sent to an output at an F-I/O or to a partner F-CPU via safety-related CPU-CPU communication.

One of the following diagnostic events is then

entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"
- "Safety program: internal CPU fault; internal error information: 404"

You must therefore ensure that the permitted range for the data type is observed when creating the program!

If this is not possible, you can avoid a STOP of the F-CPU by inserting a "Get status bit OV" instruction in the next network, thereby programming overflow detection into it.

Note the following:

- The result of the instruction behaves like the analogous instruction in a standard block.
 - The network with the "Get status bit OV" instruction must not contain any jump labels.
 - The execution time of the instruction is extended (see also Excel file for response time calculation (<http://support.automation.siemens.com/WW/view/en/49368678/134200>)).
 - A warning is issued if you do not insert a "Get status bit OV" instruction.
-

Parameters

The following table shows the parameters of the instruction:

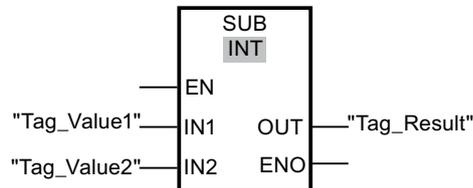
Parameter	Declaration	Data type	Description
IN1	Input	INT, DINT	Minuend
IN2	Input	INT, DINT	Subtrahend
OUT	Output	INT, DINT	Difference

You can select the data type of the instruction in the "<???">" drop-down list in the instruction box.

Example

The following example shows how the instruction works:

Network 1:



Network 2:



The "Subtract" instruction is always run (regardless of the signal state at enable input EN).

The value of operand "Tag_Value2" is subtracted from the value of operand "Tag_Value1". The result of the addition is stored in operand "Tag_Result".

If an overflow occurs during execution of the "Subtract" instruction, the status bit OV is set to "1". In network 2, following the query of the status bit OV, the "Set output" (S) instruction is executed and the "TagOut" operand is set.

See also

OV: Get status bit OV (STEP 7 Safety Advanced V11) (Page 509)

13.3.7.3 MUL: Multiply (STEP 7 Safety Advanced V11)

Description

You can use the "Multiply" instruction to multiply the value at input IN1 by the value at input IN2 and query the product at output OUT ($OUT = IN1 \times IN2$).

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Note

If the result of the instruction is outside the range permitted for the data type, then the F-CPU goes to STOP when the result is sent to an output at an F-I/O or to a partner F-CPU via safety-related CPU-CPU communication.

One of the following diagnostic events is then

entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"
- "Safety program: internal CPU fault; internal error information: 404"

You must therefore ensure that the permitted range for the data type is observed when creating the program!

If this is not possible, you can avoid a STOP of the F-CPU by inserting a "Get status bit OV" instruction in the next network, thereby programming overflow detection into it.

Note the following:

- The result of the instruction behaves like the analogous instruction in a standard block.
 - The network with the "Get status bit OV" instruction must not contain any jump labels.
 - The execution time of the instruction is extended (see also Excel file for response time calculation (<http://support.automation.siemens.com/WW/view/en/49368678/134200>)).
 - A warning is issued if you do not insert a "Get status bit OV" instruction.
-

Parameters

The following table shows the parameters of the instruction:

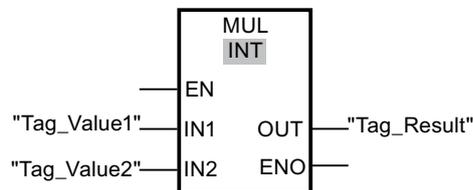
Parameter	Declaration	Data type	Description
IN1	Input	INT, DINT	Multiplier
IN2	Input	INT, DINT	Multiplicand
OUT	Output	INT, DINT	Product

You can select the data type of the instruction in the "<???">" drop-down list in the instruction box.

Example

The following example shows how the instruction works:

Network 1:



Network 2:



The "Multiply" instruction is always executed (regardless of the signal state at enable input EN).

The value of the "Tag_Value1" operand is multiplied by the value of the "Tag_Value2" operand. The result of the multiplication is stored in the "Tag_Result" operand.

If an overflow occurs during execution of the "Multiply" instruction, the status bit OV is set to "1". In network 2, following the query of the status bit OV, the "Set output" (S) instruction is executed and the "TagOut" operand is set.

See also

OV: Get status bit OV (STEP 7 Safety Advanced V11) (Page 509)

13.3.7.4 DIV: Divide (STEP 7 Safety Advanced V11)

Description

You can use the "Divide" instruction to divide the value at input IN1 by the value at input IN2 and query the quotient at the OUT output ($OUT = IN1 / IN2$).

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Note

If the result of the instruction is outside the range permitted for the data type, then the F-CPU goes to STOP when the result is sent to an output at an F-I/O or to a partner F-CPU via safety-related CPU-CPU communication.

One of the following diagnostic events is then

entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"
- "Safety program: internal CPU fault; internal error information: 404"

You must therefore ensure that the permitted range for the data type is observed when creating the program!

If this is not possible, you can avoid a STOP of the F-CPU by inserting a "Get status bit OV" instruction in the next network, thereby programming overflow detection into it.

Note the following:

- The result of the instruction behaves like the analogous instruction in a standard block.
 - The network with the "Get status bit OV" instruction must not contain any jump labels.
 - The execution time of the instruction is extended (see also Excel file for response time calculation (<http://support.automation.siemens.com/WW/view/en/49368678/134200>)).
 - A warning is issued if you do not insert a "Get status bit OV" instruction.
-

Note

If the divisor (input IN2) of a DIV instruction = 0, the quotient of the division (result of division at output OUT) = 0. The result behaves like the corresponding instruction in a standard block. The F-CPU does *not* go to STOP mode. The behavior occurs regardless of whether a "Get status bit OV" instruction has been inserted in the next network.

Parameters

The following table shows the parameters of the instruction:

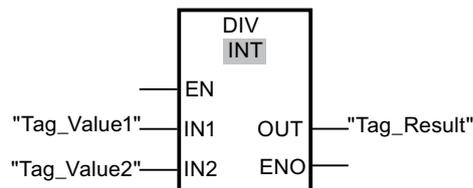
Parameter	Declaration	Data type	Description
IN1	Input	INT, DINT	Dividend
IN2	Input	INT, DINT	Divisor
OUT	Output	INT, DINT	Quotient

You can select the data type of the instruction in the "<???" drop-down list in the instruction box.

Example

The following example shows how the instruction works:

Network 1:



Network 2:



The "Divide" instruction is always executed (regardless of the signal state at enable input EN).

The value of operand "Tag_Value1" is divided by the value of operand "Tag_Value2". The result of the division is stored in operand "Tag_Result".

If an overflow occurs during execution of the "Divide" instruction, the status bit OV is set to "1". In network 2, following the query of the status bit OV, the "Set output" (S) instruction is executed and the "TagOut" operand is set.

See also

OV: Get status bit OV (STEP 7 Safety Advanced V11) (Page 509)

13.3.7.5 NEG: Create twos complement (STEP 7 Safety Advanced V11)

Description

You can use the "Create twos complement" instruction to change the sign of the value at input IN input and query the result at output OUT. If there is a positive value at input IN, for example, the negative equivalent of this value is sent to output OUT.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Note

If the result of the instruction is outside the range permitted for the data type, then the F-CPU goes to STOP when the result is sent to an output at an F-I/O or to a partner F-CPU via safety-related CPU-CPU communication.

One of the following diagnostic events is then

entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"
- "Safety program: internal CPU fault; internal error information: 404"

You must therefore ensure that the permitted range for the data type is observed when creating the program!

If this is not possible, you can avoid a STOP of the F-CPU by inserting a "Get status bit OV" instruction in the next network, thereby programming overflow detection into it.

Note the following:

- The result of the instruction behaves like the analogous instruction in a standard block.
 - The network with the "Get status bit OV" instruction must not contain any jump labels.
 - The execution time of the instruction is extended (see also Excel file for response time calculation (<http://support.automation.siemens.com/WW/view/en/49368678/134200>)).
 - A warning is issued if you do not insert a "Get status bit OV" instruction.
-

Parameters

The following table shows the parameters of the instruction:

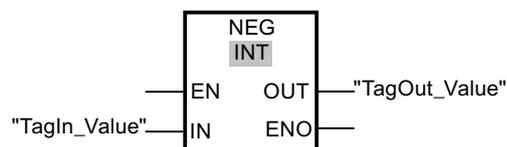
Parameter	Declaration	Data type	Description
IN	Input	INT, DINT	Input value
OUT	Output	INT, DINT	Twos complement of the input value

You can select the data type of the instruction in the "<???">" drop-down list in the instruction box.

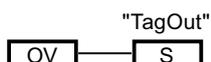
Example

The following example shows how the instruction works:

Network 1:



Network 2:



The "Create twos complement" instruction is always executed (regardless of the signal state at enable input EN).

The sign of the "TagIn_Value" operand is changed and the result is stored in the "TagOut_Value" operand.

If an overflow occurs during execution of the "Create twos complement" instruction, the status bit OV is set to "1". In network 2, following the query of the status bit OV, the "Set output" (S) instruction is executed and the "TagOut" operand is set.

See also

OV: Get status bit OV (STEP 7 Safety Advanced V11) (Page 509)

13.3.8 Move operations

13.3.8.1 MOVE: Move value (STEP 7 Safety Advanced V11)

Description

You can use the "Move value" instruction to transfer the content of the operand at input IN to the operand at output OUT1.

Only identical operand widths can be specified for input IN and output OUT1.

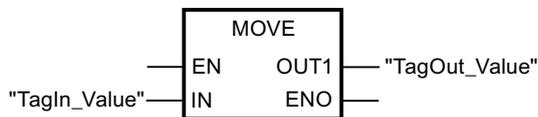
Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Parameters

Parameter	Declaration	Data type	Description
IN	Input	INT, DINT, WORD, DWORD, TIME	Source value
OUT1	Output	INT, DINT, WORD, DWORD, TIME	Destination address

Example

The following example shows how the instruction works:



The instruction is always executed regardless of the signal state at enable input "EN". The instruction copies the content of operand "TagIn_Value" to operand "TagOut_Value".

13.3.8.2 WR_FDB: Write value indirectly to an F-DB (STEP 7 Safety Advanced V11)

Description

This instruction writes the value specified in input IN to the tag addressed by INI_ADDR and OFFSET in an F-DB.

The address of the tags addressed using INI_ADDR and OFFSET must be within the address range defined by addresses INI_ADDR and END_ADDR.

If the F-CPU has gone to STOP mode with diagnostic event ID 75E2, check to determine if this condition is fulfilled.

The start address of the area in an F-DB to which the value at input IN is to be written is transferred using input INI_ADDR. The associated offset in this area is transferred using input OFFSET.

The addresses transferred in input INI_ADDR or END_ADDR must point to a tag of the selected data type in an F-DB. Only tags of the selected data type are permitted between the INI_ADDR and END_ADDR addresses. The INI_ADDR address must be smaller than the END_ADDR address.

As shown in the following example, the INI_ADDR and END_ADDR addresses must be transferred fully-qualified as "DBx".DBWy or in the corresponding symbolic representation. Transfers in other forms are not permitted.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Parameters

Parameter	Declaration	Data type	Description
IN	Input	INT, DINT	Value to be written to the F-DB
INI_ADDR	Input	POINTER	Start address of the area in an F-DB
END_ADDR	Input	POINTER	End address of the area in an F-DB
OFFSET	Input	INT	Offset

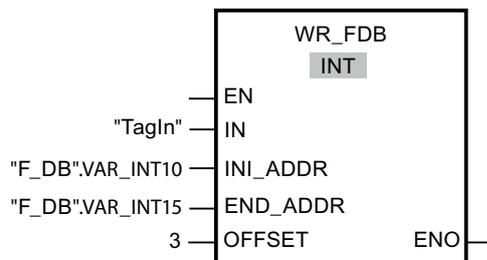
You can select the data type of the instruction in the "<???">" drop-down list in the instruction box.

Examples of parameter assignment of INI_ADDR, END_ADDR, and OFFSETS

Name	Data type	Initial value	Comment
Static			
VAR_BOOL10	Bool	false	
VAR_BOOL11	Bool	false	
VAR_BOOL12	Bool	false	
VAR_BOOL13	Bool	false	
VAR_TIME10	Time	T#0MS	
VAR_TIME11	Time	T#0MS	
VAR_INT10	Int	0	<- INI_ADDR = "F-DB".VAR_INT10 Example 1
VAR_INT11	Int	0	
VAR_INT12	Int	0	
VAR_INT13	Int	0	<- OFFSET = 3
VAR_INT14	Int	0	
VAR_INT15	Int	0	<- END_ADDR = "F-DB".VAR_INT15
VAR_BOOL20	Bool	false	
VAR_BOOL21	Bool	false	
VAR_BOOL22	Bool	false	
VAR_BOOL23	Bool	false	
VAR_INT20	Int	0	<- INI_ADDR = "F-DB".VAR_INT20 Example 2
VAR_INT21	Int	0	
VAR_INT22	Int	0	
VAR_INT23	Int	0	<- INI_END = "F-DB".VAR_INT23
VAR_INT30	Int	0	<- INI_ADDR = "F-DB".VAR_INT30 Example 3
VAR_INT31	Int	0	<- OFFSET = 1
VAR_INT32	Int	0	
VAR_INT33	Int	0	
VAR_INT34	Int	0	<- END_ADDR = "F-DB".VAR_INT34
VAR_TIME20	TIME	T#0MS	
VAR_DINT10	DInt	0	<- INI_ADDR = "F-DB".VAR_DINT10 Example 4
VAR_DINT11	DInt	0	
VAR_DINT12	DInt	0	<- OFFSET = 2
VAR_DINT13	DInt	0	<- END_ADDR = "F-DB".VAR_DINT13

Example

The following example shows how the instruction works:



13.3.8.3 RD_FDB: Read value indirectly from an F-DB (STEP 7 Safety Advanced V11)

Description

This instruction reads the tag addressed via INI_ADDR and OFFSET in an F-DB and provides it at output OUT.

The address of the tags addressed using INI_ADDR and OFFSET must be within the address range defined by addresses INI_ADDR and END_ADDR.

If the F-CPU has gone to STOP mode with diagnostic event ID 75E2, check to determine if this condition is fulfilled.

The start address of the area in an F-DB from which the tag is to be read is transferred using input INI_ADDR. The associated offset in this area is transferred using input OFFSET.

The addresses transferred in input INI_ADDR or END_ADDR must point to a tag of the selected data type in an F-DB. Only tags of the selected data type are permitted between the INI_ADDR and END_ADDR addresses. The INI_ADDR address must be smaller than the END_ADDR address.

The INI_ADDR and END_ADDR addresses must be transferred fully-qualified as "DBx".DBWy or in the corresponding symbolic representation. Transfers in other forms are not permitted. Examples of parameter assignment of INI_ADDR, END_ADDR, and OFFSET are contained in WR_FDB: Write value indirectly to an F-DB (STEP 7 Safety Advanced V11) (Page 485).

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

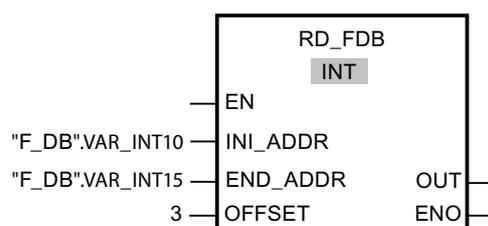
Parameters

Parameter	Declaration	Data type	Description
INI_ADDR	Input	POINTER	Start address of the area in an F-DB
END_ADDR	Input	POINTER	End address of the area in an F-DB
OFFSET	Input	INT	Offset
OUT	Output	INT, DINT	Value to be read from the F-DB

You can select the data type of the instruction in the "<???">" drop-down list in the instruction box.

Example

The following example shows how the instruction works:



13.3.9 Conversion operations

13.3.9.1 CONVERT: Convert value (STEP 7 Safety Advanced V11)

Description

The "Convert value" instruction reads the content of parameter IN and converts it according to the data types selected in the instruction box. The converted value is output at output OUT.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

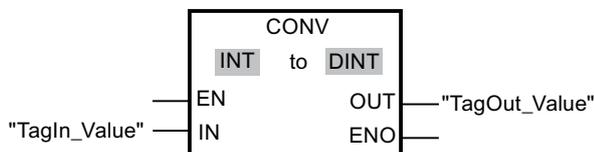
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN	Input	INT	Value to be converted.
OUT	Output	DINT	Result of the conversion

Example

The following example shows how the instruction works:



The instruction is always executed regardless of the signal state at enable input "EN". The content of the operand "TagIn_Value" is read and converted to a double integer (32 bit). The result is stored in operand "TagOut_Value".

13.3.9.2 BO_W: Convert 16 data elements of data type BOOL to a data element of data type WORD (STEP 7 Safety Advanced V11)

Description

This instruction converts the 16 values of data type BOOL at inputs IN0 to IN15 to a value of data type WORD, which is made available at output OUT. The conversion takes place as follows: The i-th bit of the WORD value is set to 0 (or 1), if the value at input INi = 0 (or 1).

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

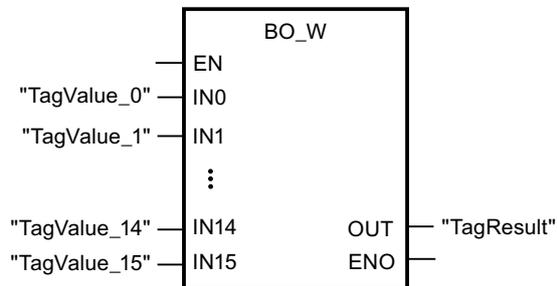
Note: To supply inputs IN0 to IN15 with Boolean constants "0" and "1", you can access tags "VKE0" and "VKE1" in the F-shared DB using a fully-qualified DB access ("F_GLOBDB".VKE0 or "F_GLOBDB".VKE1).

Parameters

Parameter	Declaration	Data type	Description
IN0	Input	BOOL	Bit 0 of WORD value
IN1	Input	BOOL	Bit 1 of WORD value
...			...
IN15	Input	BOOL	Bit 15 of WORD value
OUT	Output	WORD	WORD value consisting of IN0 to IN15

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN0	TagValue_0	FALSE
IN1	TagValue_1	FALSE
...		...
IN13	TagValue_13	FALSE
IN14	TagValue_14	TRUE
IN15	TagValue_15	TRUE
OUT	TagResult	W#16#0003

The values of operands "TagValue_0" to "TagValue_15" are combined to form data type WORD and assigned to operand "TagResult".

13.3.9.3 W_BO: Convert a data element of data type WORD to 16 data elements of data type BOOL (STEP 7 Safety Advanced V11)

Description

This instruction converts the value of data type WORD at input IN to 16 values of data type BOOL, which are provided at outputs OUT0 to OUT15. The conversion takes place as follows: Output OUT_i is set to 0 (or 1), if the i-th bit of the WORD value is 0 (or 1).

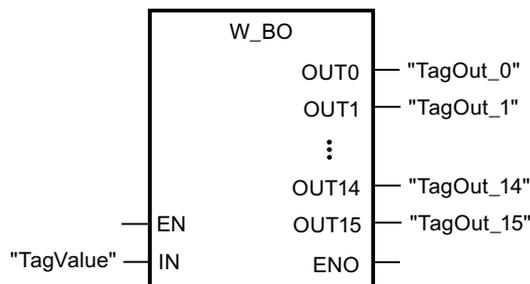
Enable input "EN" or enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Parameters

Parameter	Declaration	Data type	Description
IN	Input	WORD	WORD value
OUT0	Output	BOOL	Bit 0 of WORD value
OUT1	Output	BOOL	Bit 1 of WORD value
...			...
OUT15	Output	BOOL	Bit 15 of WORD value

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagValue	W#16#0003
OUT0	TagOUT_0	FALSE
OUT1	TagOUT_1	FALSE
...		...
OUT13	TagOUT_13	FALSE
OUT14	TagOUT_14	TRUE
OUT15	TagOUT_15	TRUE

The value of operand "TagValue" of data type WORD is converted to the 16 values "TagOUT_0" to "TagOUT_15" of data type BOOL.

13.3.9.4 SCALE: Scale values (STEP 7 Safety Advanced V11)

Description

This instruction scales the value at input IN in physical units between the low limit value at input LO_LIM and the high limit value at input HI_LIM. It is assumed that the value at input IN is between 0 and 27648. The scaling result is provided at output OUT.

The instruction uses the following equation:

$$\text{OUT} = [\text{IN} \times (\text{HI_LIM} - \text{LO_LIM})] / 27648 + \text{LO_LIM}$$

As long as the value at input IN is greater than 27648, output OUT is linked to HI_LIM and OUT_HI is set to 1.

As long as the value at input IN is less than 0, output OUT is linked to LO_LIM and OUT_LO is set to 1.

For inverse scaling, you must assign LO_LIM > HI_LIM. With inverse scaling, the output value at output OUT decreases while the input value at input IN increases.

Every call of the "Scale values" instruction must be assigned a data area in which the instruction data are stored. The "Call options" dialog is automatically opened for this reason when the instruction is inserted in the program; in it you can create a data block (single instance) (e.g., SCALE_DB_1) or a multi-instance (e.g., SCALE_Instance_1) for the "Scale values" instruction. Once you have created the data block, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Parameters

Parameter	Declaration	Data type	Description
IN	Input	INT	Input value to be scaled in physical units
HI_LIM	Input	INT	High limit value of value range of OUT
LO_LIM	Input	INT	Low limit value of value range of OUT
OUT	Output	INT	Result of scaling
OUT_HI	Output	BOOL	1 = Input value > 27648: OUT = HI_LIM
OUT_LO	Output	BOOL	1 = Input value < 0: OUT = LO_LIM

Behavior in the event of overflow or underflow of analog values and fail-safe value output

Note

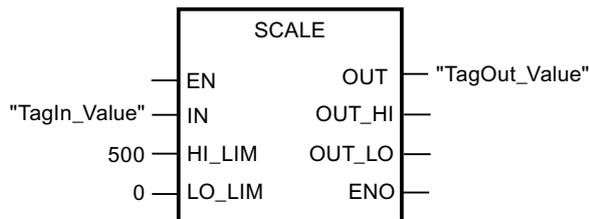
If inputs from the PII of an SM 336; AI 6 x 13Bit or SM 336; F-AI 6 x 0/4 ... 20 mA HART are used as input values, note that the F-system detects an overflow or underflow of a channel of this F-SM as an F-I/O fault or channel fault. The fail-safe value 0 is provided in place of 7FFF_H (for overflow) or 8000_H (for underflow) in the PII for the safety program.

If other fail-safe values are to be output in this case, you must evaluate the QBAD tag in the F-I/O DB (branch to output of an individual fail-safe value).

If the value in the PII of the F-SM is within the overrange or underrange, but is > 27648 or < 0, you can likewise branch to the output of an individual fail-safe value by evaluating outputs OUT_HI and OUT_LO, respectively.

Example

The following example shows how the instruction works:



When operand "TagIn_Value" = 20000, the result is "TagOut_Value" 361.

13.3.10 Program control operations

13.3.10.1 JMP: Jump if RLO = 1 (STEP 7 Safety Advanced V11)

Description

You can use the "Jump if RLO = 1" instruction to interrupt the linear execution of the program and resume it in another network. The destination network must be identified by a jump label (Page 495) (LABEL). The description of the jump label is specified in the placeholder above the instruction.

The specified jump label must be in the same block in which the instruction is executed. The name you specify can only occur once in the block.

If the result of logic operation (RLO) at the input of the instruction is "1" or the input is not connected, the jump to the network identified by the jump label is executed. The jump direction can be towards higher or lower network numbers.

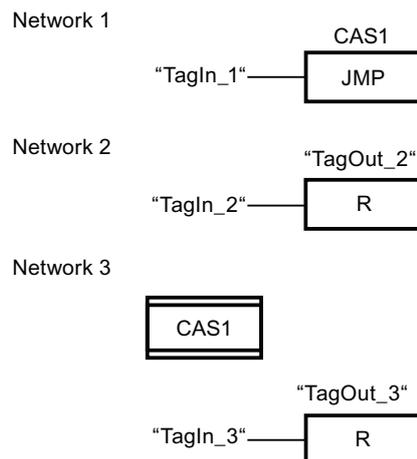
If the result of logic operation (RLO) at the input of the instruction is "0", the program continues executing in the next network.

Note

You are not permitted to program a SENDDP or SENDS7 call between a jump instruction and the associated destination of the jump instruction.

Example

The following example shows how the instruction works:



When operand "TagIn_1" has signal state "1", the "Jump if RLO = 1" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. When input "TagIn_3" has signal state "1", output "TagOut_3" is reset.

13.3.10.2 JMPN: Jump if RLO = 0 (STEP 7 Safety Advanced V11)

Description

You can use the "Jump if RLO = 0" instruction to interrupt the linear execution of the program and resume it in another network, when the result of logic operation at the input of the instruction is "0". The destination network must be identified by a jump label (Page 495) (LABEL). The designation of the jump label is specified in the placeholder above the instruction box.

The specified jump label must be in the same instruction in which the instruction is executed. The name you specify can only occur once in the instruction.

If the result of logic operation (RLO) at the input of the instruction is "0", the jump to the network identified by the jump label is executed. The jump direction can be towards higher or lower network numbers.

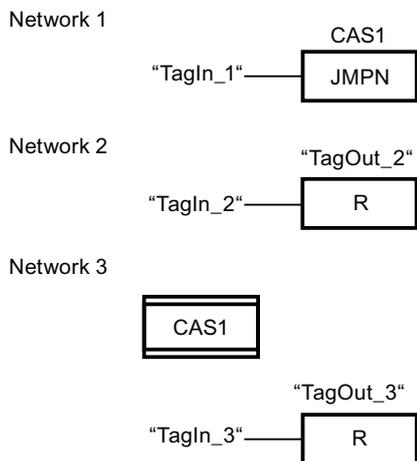
If the result of logic operation (RLO) at the input of the instruction is "1", the program continues executing in the next network.

Note

You are not permitted to program a SENDDP or SENDS7 call between a jump instruction and the associated destination of the jump instruction.

Example

The following example shows how the instruction works:



When operand "TagIn_1" has signal state "0", instruction "Jump if RLO = 0" is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. When input "TagIn_3" has signal state "1", output "TagOut_3" is reset.

13.3.10.3 LABEL: Jump label (STEP 7 Safety Advanced V11)

Description

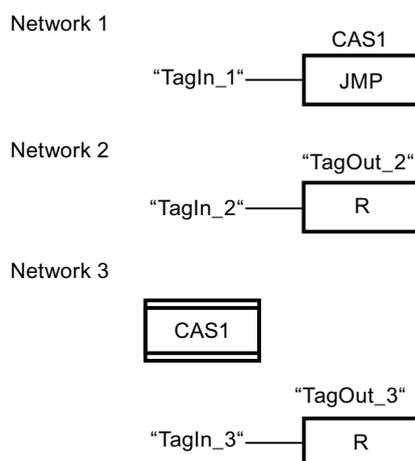
You can use a jump label to specify a destination network, in which the program execution should resume after a jump.

The jump label and the instruction in which the jump label is specified must be located in the same block. The name of a jump label can only be assigned once in a block.

Only one jump label can be placed in a network. To each jump label can be jumped from several locations.

Example

The following example shows how the instruction works:



When operand "TagIn_1" has signal state "1", the "JMP: Jump if RLO = 1" instruction is executed. The linear execution of the program is interrupted and continues in Network 3, which is identified by the jump label CAS1. When input "TagIn_3" has signal state "1", output "TagOut_3" is reset.

See also

JMP: Jump if RLO = 1 (STEP 7 Safety Advanced V11) (Page 493)

JMPN: Jump if RLO = 0 (STEP 7 Safety Advanced V11) (Page 494)

RET: Return (STEP 7 Safety Advanced V11) (Page 496)

13.3.10.4 RET: Return (STEP 7 Safety Advanced V11)

Description

You can use the "Return" operation to stop the processing of a block.

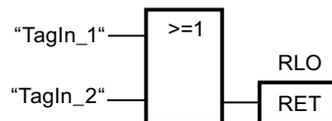
If the result of logic operation (RLO) at the input of the "Return" instruction is "1", program execution is terminated in the currently called block and continued in the calling block (for example, in the main safety block) after the call function. If the RLO at the input of the "Return" instruction is "0", the instruction is not executed. Program execution continues in the next network of the called block.

Note

You must not program a "Return" instruction in the Main Safety Block.

Example

The following example shows how the instruction works:



If either the "TagIn_1" or "TagIn_2" operand has the signal state "1", the "Return" instruction is executed. Program execution in the called block is terminated and continues in the calling block. Output ENO of the call function is reset to signal state "1".

See also

LABEL: Jump label (STEP 7 Safety Advanced V11) (Page 495)

JMPN: Jump if RLO = 0 (STEP 7 Safety Advanced V11) (Page 494)

JMP: Jump if RLO = 1 (STEP 7 Safety Advanced V11) (Page 493)

13.3.10.5 OPN: Open global data block (STEP 7 Safety Advanced V11)

Description

You can use the "Open global data block" instruction to open a data block. The number of the data block is transferred to the DB register. Subsequent DB commands access the relevant blocks depending on the register contents.

Note

Note when using the "Open global data block" instruction that the content of the DB register can be changed after calls of F-FB/F-FC and "fully qualified DB accesses", such that there is no guarantee that the last data block you opened with "Open global data block" is still open.

You should therefore use the following method for addressing data to avoid errors when accessing data of the DB register:

- Use symbolic addressing.
- Use only fully qualified DB accesses.

If you still want to use the "Open global data block" instruction, you must ensure that the DB register is restored by repeating the "Open global data block" instruction after calls of F-FB/F-FC and "fully qualified DB accesses." Otherwise, a malfunction could result.

Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
<Data block>	Input	BLOCK_DB	Data block that is opened

"Fully qualified DB access"

The initial access to data of a data block in an F-FB/F-FC must always be a "fully qualified DB access," or it must be preceded by the "Open global data block" instruction. This also applies to the initial access to data of a data block after a jump label.

An example of "fully-qualified DB access" and "non-fully-qualified DB access" can be found under Restrictions in the programming languages FBD/LAD (Page 59).

Access to instance DBs

You can also access instance DBs of F-FBs with fully qualified access, e.g., for transfer of block parameters. It is not possible to access static data in instance DBs of other F-FBs.

Note that accessing instance DBs of F-FBs that are not called in the safety program can cause the F-CPU to go to STOP mode.

Example

The following example shows how the instruction works:

Network 1

"Motor_DB"



Network 2



The "Motor_DB" data block is called in network 1. The number of the data block is transferred to the DB register. The "DBX0.0" operand is queried in network 2. The signal state of the "DBX0.0" operand is assigned to the "Tag_Output" operand.

13.3.11 Word logic operations

13.3.11.1 AND: AND logic operation (STEP 7 Safety Advanced V11)

Description

You can use the "AND logic operation" instruction to combine the value at input IN1 to the value at input IN2 bit-by-bit by AND logic and query the result at output OUT.

When the instruction is executed, bit 0 of the value at input IN1 and bit 0 of the value at input IN2 are ANDed. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified values.

The result bit has signal state "1" only when both of the bits in the logic operation also have signal state "1". If one of the two bits of the logic operation has signal state "0", the corresponding result bit is reset.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

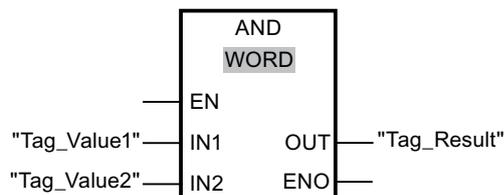
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN1	Input	WORD	First value of logic operation
IN2	Input	WORD	Second value of logic operation
OUT	Output	WORD	Result of the instruction

Example

The following example shows how the instruction works:



IN1	"Tag_Value1" = 01010101 01010101
IN2	"Tag_Value2" = 00000000 00001111
OUT	"Tag_Result" = 00000000 00000101

The value of the "Tag_Value1" operand and the value of the "Tag_Value2" operand are ANDed. The result is mapped bit-by-bit and output in the "Tag_Result" operand.

13.3.11.2 OR: OR logic operation (STEP 7 Safety Advanced V11)

Description

You can use the "OR logic operation" instruction to connect the value at input IN1 input to the value at input IN2 bit-by-bit by OR logic and query the result at output OR.

When the instruction is executed, bit 0 of the value at input IN1 and bit 0 of the value at input IN2 are ORed. The result is stored in bit 0 of output OUT. The same logic operation is executed for all bits of the specified tags.

The result bit has signal state "1" when at least one of the two bits in the logic operation has signal state "1". If both of the bits of the logic operation have signal state "0", the corresponding result bit is reset.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Parameters

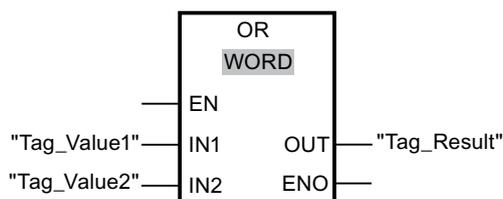
The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN1	Input	WORD	First value of logic operation
IN2	Input	WORD	Second value of logic operation
OUT	Output	WORD	Result of the instruction

The number of inputs is limited to two and cannot be expanded.

Example

The following example shows how the instruction works:



IN1	"Tag_Value1" = 01010101 01010101
IN2	"Tag_Value2" = 00000000 00001111
OUT	"Tag_Result" = 01010101 01011111

The value of the "Tag_Value1" operand and the value of the "Tag_Value2" operand are ORed. The result is mapped bit-by-bit and output in the "Tag_Result" operand.

13.3.11.3 XOR: EXCLUSIVE OR logic operation (STEP 7 Safety Advanced V11)

Description

You can use the "EXCLUSIVE OR logic operation" instruction to combine the value at input IN1 and the value at input IN2 bit-by-bit by EXCLUSIVE OR logic and query the result at output OUT.

When the instruction is executed, bit 0 of the value at input IN1 input and bit 0 of the value at input IN2 are logically combined by EXCLUSIVE OR. The result is stored in bit 0 of output OUT. The same logic operation is executed for all other bits of the specified value.

The result bit has signal state "1" when one of the two bits in the logic operation has signal state "1". If both of the bits of the logic operation have signal state "1" or "0", the corresponding result bit is reset.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Parameters

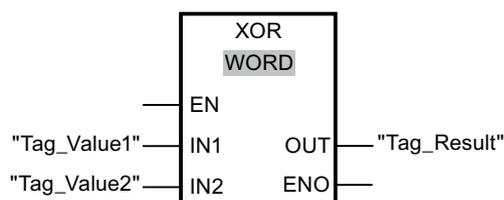
The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN1	Input	WORD	First value of logic operation
IN2	Input	WORD	Second value of logic operation
OUT	Output	WORD	Result of the instruction

The number of inputs is limited to two and cannot be expanded.

Example

The following example shows how the instruction works:



IN1	"Tag_Value1" = 01010101 01010101
IN2	"Tag_Value2" = 00000000 00001111
OUT	"Tag_Result" = 01010101 01011010

The value of the "Tag_Value1" operand and the value of the "Tag_Value2" operand are logically combined by EXCLUSIVE OR. The result is mapped bit-by-bit and output in the "Tag_Result" operand.

13.3.12 Shift and rotate

13.3.12.1 SHR: Shift right (STEP 7 Safety Advanced V11)

Description

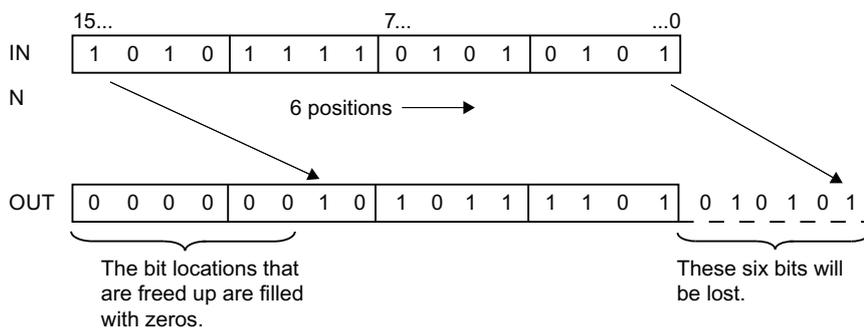
You can use the "Shift right" instruction to move the content of the operand at input IN bit-by-bit to the right and query the result at output OUT. You use parameter N (low-byte) to specify the number of bit positions by which the specified value is moved.

When the value at parameter N (low-byte) is "0", the value at input IN is copied into the operand at output OUT.

When the value at parameter N (low-byte) is greater than the number of available bit positions, the operand value at input IN is moved by the available number of bit positions to the right.

The bit locations that are freed up in the left area of the operand during the shift operation are filled with zeros.

The following figure shows how the content of an operand of data type WORD is moved by 6 bit positions to the right:



Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

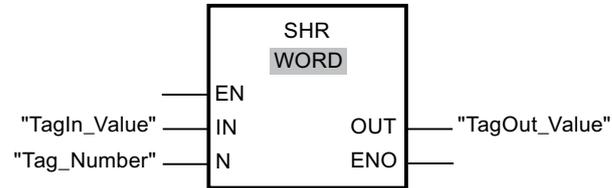
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN	Input	WORD	Value that is moved
N	Input	INT	Number of bit positions by which the value is moved
OUT	Output	WORD	Result of the instruction

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	3
OUT	TagOut_Value	0000 0111 1111 0101

The content of the operand "TagIn_Value" is moved three bit positions to the right. The result is output at output "TagOut_Value".

13.3.12.2 SHL: Shift left (STEP 7 Safety Advanced V11)

Description

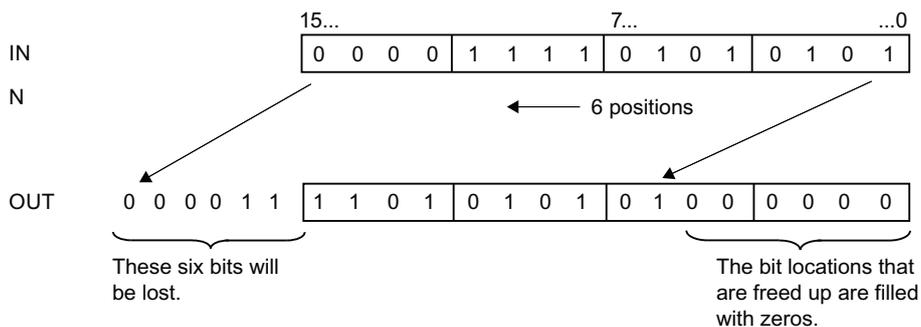
You can use the "Shift left" instruction to move the content of the operand at input IN bit-by-bit to the left and query the result at output OUT. You use parameter N (low-byte) to specify the number of bit positions by which the specified value is moved.

When the value at parameter N (low-byte) is "0", the value at input IN is copied into the operand at output OUT.

When the value at parameter N (low-byte) is greater than the number of available bit positions, the operand value at input IN is moved by the available number of bit positions to the left.

The bit positions that are freed up in the right area of the operand during the shift operation are filled with zeros.

The following figure shows how the content of an operand of data type WORD is moved by 6 bit positions to the left:



Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

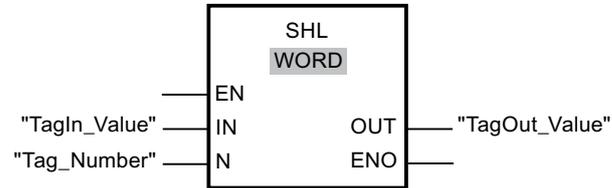
Parameters

The following table shows the parameters of the instruction:

Parameter	Declaration	Data type	Description
IN	Input	WORD	Value that is moved
N	Input	INT	Number of bit positions by which the value is moved
OUT	Output	WORD	Result of the instruction

Example

The following example shows how the instruction works:



The following table shows how the instruction works using specific operand values:

Parameter	Operand	Value
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	4
OUT	TagOut_Value	1111 1010 1111 0000

The content of the operand "TagIn_Value" is moved four bit positions to the left. The result is output at output "TagOut_Value".

13.3.13 Operating

13.3.13.1 ACK_OP: Fail-safe acknowledgment (STEP 7 Safety Advanced V11)

Description

This instruction enables fail-safe acknowledgment from an operator control and monitoring system. It allows, for example, reintegration of F-I/O to be controlled from the operator control and monitoring system. Acknowledgment takes place in two steps:

- In/out parameter IN changes to a value of 6.
- In/out parameter IN changes to a value of 9 within 1 minute.

Once the in/out parameter IN has changed to a value of 6, the instruction evaluates whether this parameter has changed to a value of 9 after 1 second, at the earliest, or 1 minute, at the latest. Output OUT (output for acknowledgment) is then set to 1 for one cycle.

If an invalid value is input or if in/out parameter IN has not changed to 9 within 1 minute or the change occurred before 1 second has elapsed, then in/out parameter IN is reset to 0, and both steps listed above must be repeated.

During the time in which in/out parameter IN must change from 6 to 9, output Q is set to 1. Otherwise, Q has a value of 0.

Every call of the "Fail-safe acknowledgment" instruction must be assigned a data area in which the instruction data are stored. The "Call options" dialog is automatically opened for this reason when the instruction is inserted in the program; in it you can create a data block (single instance) (e.g., ACK_OP_DB_1) or a multi-instance (e.g., ACK_OP_Instance_1) for the "Fail-safe acknowledgment" instruction. Once you have created the data block, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Note

A separate data area must be used for each call of ACK_OP. Each call can be processed only once in an F-runtime group cycle.

The F-CPU can go to STOP mode if the information above is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
 - "Data corruption in the safety program prior to output to partner F-CPU"
 - "Safety program: internal CPU fault; internal error information: 404"
-

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

WARNING

When using an instruction with time processing, take the following timing imprecision sources into account when determining your response times:

- Known timing imprecision (based on standard systems) resulting from cyclic processing
- Timing imprecision resulting from the update time of the time base used in the instruction (see figure in section "Timing imprecision resulting from the update time of the time base used in the instruction")
- Tolerance of internal time monitoring in the F-CPU
 - For time values up to 100 ms, a maximum of 20% of the (assigned) time value
 - For time values greater than or equal to 100 ms, a maximum of 2% of the (assigned) time value

You must choose the interval between two call times of an instruction with time processing in such a way that the required response times are achieved, taking into account the possible timing imprecision. (S034)

Note

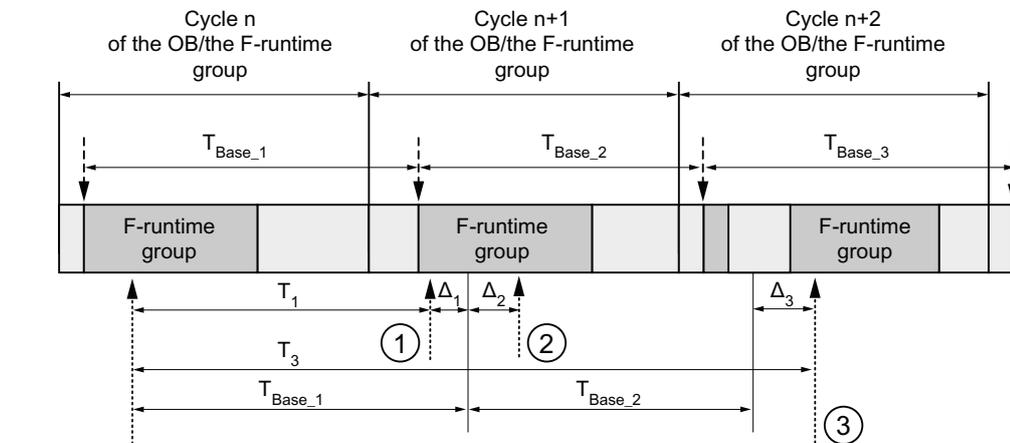
You can read out output Q by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program.

You can supply in/out parameter IN with just a memory word or nothing at all. In the safety program, read and write access to in/out parameter IN in the associated instance DB is not permitted!

Parameters

Parameter	Declaration	Data type	Description
IN	InOut	INT	Input variable from operator control and monitoring system
OUT	Output	BOOL	Output for acknowledgment
Q	Output	BOOL	Time status

Timing imprecision resulting from the update time of the time base used in the instruction:

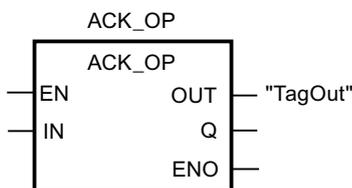


-----> = Time base update
> = Call time of an instruction with time processing

- ① For the first call in cycle n+1, the call time of the instruction relative to the start of the F-runtime group is earlier than that in cycle n by the amount of Δ_1 , e.g., because portions of the safety program of the F-runtime group before the call time of the instruction in cycle n+1 are skipped. For the time update, the instruction takes into account time T_{Base_1} instead of the time T_1 that has actually elapsed in cycle n since the call.
- ② The instruction is called a second time in cycle n+1. This does not involve another time update (by Δ_2).
- ③ For the call in cycle n+2, the call time of the instruction relative to the start of the F-runtime group is later than that in cycle n by the amount of Δ_3 , e.g., because the F-runtime group was interrupted by a higher priority interrupt prior to the time of the instruction call in cycle n+2. The instruction took into account time $T_{Base_1} + T_{Base_2}$ instead of the time T_3 that has actually elapsed in cycle n since the call. This would also be the case if no call occurred in cycle n+1.

Example

The following example shows how the instruction works:



See also

Implementing User Acknowledgment in the Safety Program of the F-CPU of a DP Master or IO Controller (Page 99)

Implementing user acknowledgment in the safety program of the F-CPU of a I-slave or I-device (Page 102)

13.3.14 Additional instructions

13.3.14.1 OV: Get status bit OV (STEP 7 Safety Advanced V11)

Description

You can use the "Get status bit OV" instruction to detect whether a number range overflow occurred in the last arithmetic instruction processed.

The "Get status bit OV" evaluation must be inserted in the network that follows the instruction that influences the OV. This network must not contain any jump labels.

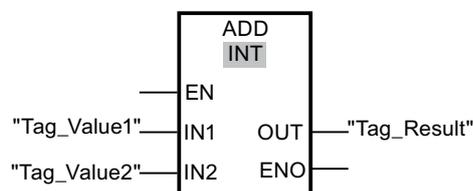
Note

The execution time of the OV-affecting instruction is extended when the "Get status bit OV" instruction is used (see also Excel file for response time calculation (<http://support.automation.siemens.com/WW/view/en/49368678/134200>)).

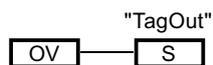
Example

The following example shows how the instruction works:

Network 1:



Network 2:



The value of the "Tag_Value1" operand is added to value of the Tag_Value2 operand. The result of the addition is stored in the "Tag_Result" operand.

If an overflow occurs during execution of the "Add" instruction, the status bit OV is set to "1". In network 2, following the query of the status bit OV, the "Set output" (S) instruction is executed and the "TagOut" operand is set.

13.3.15 Communication

13.3.15.1 PROFIBUS/PROFINET

SENDDP and RCVDP: Send and receive data via PROFIBUS DP/PROFINET IO (STEP 7 Safety Advanced V11)

Introduction

You use the SENDDP and RCVDP instructions for fail-safe sending and receiving of data using:

- Safety-related master-master communication
- Safety-related master-master communication for S7 Distributed Safety
- Safety-related master-I-slave communication
- Safety-related I-slave-I-slave communication
- Safety-related IO controller-IO controller communication
- Safety-related IO Controller-IO Controller communication for S7 Distributed Safety
- Safety-related IO controller-I-Device communication
- Safety-related IO controller-I-Slave communication

Description

The SENDDP instruction sends 16 data elements of data type BOOL and 2 data elements of data type INT in a fail-safe manner to another F-CPU via PROFIBUS DP/PROFINET IO. The data can be received there by the related RCVDP instruction.

Every call of this instruction must be assigned a data area in which the instruction data are stored. In addition, when the instruction is inserted in the program, the "Call options" dialog is automatically opened, where you can create a data block (single instance) (e.g., RCVDP_DB_1) or a multi-instance (e.g., RCVDP_Instance_1) for this instruction. Following the creation step, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

With the SENDDP instruction, the data to be sent (for example, outputs of other F-blocks/instructions) are available at input SD_BO_xx or SD_I_xx.

With the RCVDP instruction, the data received are available at output RD_BO_xx or RD_I_xx for additional processing by other F-blocks/instructions.

The operating mode of the F-CPU with the SENDDP instruction is provided at output SENDMODE. If the F-CPU with the SENDDP instruction is in deactivated safety mode, output SENDMODE = 1.

Communication between F-CPU's takes place in the background by means of a special safety protocol. You must define the communication relationship between a SENDDP instruction in one F-CPU and a RCVDP instruction in the other F-CPU by specifying an address relationship at the DP_DP_ID inputs of the SENDDP and RCVDP instructions. Associated SENDDP and RCVDP instructions are assigned the same value for DP_DP_ID.

⚠ WARNING

The value for each address relationship (input parameter DP_DP_ID; data type: INT) is user-defined; however, it must be unique from all other safety-related communication connections in the network. The uniqueness must be checked in the print-out of the safety program during acceptance testing of the safety program. Additional information can be found in Auto-Hotspot.

You must supply inputs DP_DP_ID and LADDR with constant values when calling the instruction. Direct read or write access to the associated instance DB is not permitted in the safety program! (S016)

Note

Within a safety program, you must assign a different start address for every call of the SENDDP and RCVDP instructions at input LADDR. A separate instance DB must be used for each call of the SENDDP and RCVDP instructions. You must not declare and call these instructions as multi-instances.

The input and output parameters of the RCVDP instruction must not be supplied with temporary or static local data of the main safety block.

The input parameters of the RCVDP instruction must not be initialized with output parameters (using fully qualified DB accesses) of a RCVDP or RCVS7 instruction called in a preceding network.

You must not use an actual parameter for an output parameter of a RCVDP, if it is already being used for an input parameter of the same or another RCVDP or RCVS7 instruction. The F-CPU can go to STOP if this is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"
- "Safety program: internal CPU fault; internal error information: 404"

Note

You must not program any SENDDP instruction between a JMP or JMPN instruction and the associated destination network of the JMP or JMPN instruction.

You must not program any RET instruction before a SENDDP instruction.

Parameters of the SENDDP instruction

Parameter	Declaration	Data type	Description
SD_BO_00	Input	BOOL	Send data BOOL 00
...			...
SD_BO_15	Input	BOOL	Send data BOOL 15
SD_I_00	Input	INT	Send data INT 00
SD_I_01	Input	INT	Send data INT 01
DP_DP_ID	Input	INT	Network-wide unique value for the address relationship between a SENDDP and RCVDP instruction
TIMEOUT	Input	TIME	Monitoring time in ms for safety-related communication (see also Monitoring and response times (Page 523))
LADDR	Input	INT	Start address of address area/transfer area: <ul style="list-style-type: none"> • Of DP/DP coupler for safety-related master-master communication • For safety-related master-I-slave communication • For safety-related I-slave-I-slave communication • Of PN/PN coupler for safety-related IO Controller-IO Controller communication • For safety-related IO Controller-I-Device communication • For safety-related IO Controller-I-slave communication
ERROR	Output	BOOL	1=Communication error
SUBS_ON	Output	BOOL	1=RCVDP outputs fail-safe values
RET_DPRD	Output	WORD	Error code RET_VAL of the DPRD_DAT instruction (for a description of error codes, refer to the online help for DPRD_DAT instruction ("Extended instructions > Distributed I/O > Other".))
RET_DPWR	Output	WORD	Error code RET_VAL of the DPWR_DAT instruction (for a description of error codes, refer to the online help for DPWR_DAT instruction ("Extended instructions > Distributed I/O > Other".))
DIAG	Output	BYTE	Service information

Parameters of the RCVDP instruction

Parameter	Declaration	Data type	Description
ACK_REI	Input	BOOL	1=Acknowledgment for reintegration of send data following communication error
SUBBO_00	Input	BOOL	Fail-safe value for receive data BOOL 00
...			...
SUBBO_15	Input	BOOL	Fail-safe value for receive data BOOL 15
SUBI_00	Input	INT	Fail-safe value for receive data INT 00
SUBI_01	Input	INT	Fail-safe value for receive data INT 01
DP_DP_ID	Input	INT	Network-wide unique value for the address relationship between a SENDDP and RCVDP instruction
TIMEOUT	Input	TIME	Monitoring time in ms for safety-related communication (see also Monitoring and response times (Page 523))
LADDR	Input	INT	Start address of address area/transfer area: <ul style="list-style-type: none"> • Of DP/DP coupler for safety-related master-master communication • For safety-related master-I-slave communication • For safety-related I-slave-I-slave communication • Of PN/PN coupler for safety-related IO Controller-IO Controller communication • For safety-related IO Controller-I-Device communication • For safety-related IO Controller-I-slave communication
ERROR	Output	BOOL	1=Communication error
SUBS_ON	Output	BOOL	1=Fail-safe values are output
ACK_REQ	Output	BOOL	1=Acknowledgment for reintegration of send data required
SENDMODE	Output	BOOL	1=F-CPU with SENDDP instruction in deactivated safety mode
RD_BO_00	Output	BOOL	Receive data BOOL 00
...			...
RD_BO_15	Output	BOOL	Receive data BOOL 15
RD_I_00	Output	INT	Receive data INT 00
RD_I_01	Output	INT	Receive data INT 01
RET_DPRD	Output	WORD	Error code RET_VAL of the DPRD_DAT instruction (for a description of error codes, refer to the online help for DPRD_DAT instruction ("Extended instructions > Distributed I/O > Other".))
RET_DPWR	Output	WORD	Error code RET_VAL of the DPWR_DAT instruction (for a description of error codes, refer to the online help for DPWR_DAT instruction ("Extended instructions > Distributed I/O > Other".))
DIAG	Output	BYTE	Service information

Placement

The RCVDP instruction must be inserted at the start of the main safety block and the SENDDP instruction at the end.

Startup characteristics

After the sending and receiving F-systems are started up, communication must be established between the connection partners for the first time (SENDDP and RCVDP instructions). During this time, the receiver (RCVDP instruction) outputs the fail-safe values present at its inputs SUBBO_xx and SUBBI_xx.

The SENDDP and RCVDP instructions signal this at output SUBS_ON with 1. Output SENDMODE has a default of 0 and is not updated, as long as output SUBS_ON = 1.

Behavior in the event of communication errors

If a communication error occurs, for example, due to a signature error (CRC) or when monitoring time TIMEOUT expires, outputs ERROR and SUBS_ON = 1 are set. The receiver (RCVDP instruction) then outputs the fail-safe values assigned at its SUBBO_xx inputs. Output SENDMODE is not updated while output SUBS_ON = 1.

The send data of the SENDDP instruction present at inputs SD_BO_xx and SUBI_xx are only output again when communication errors are no longer detected (ACK_REQ = 1) and you acknowledge the RCVDP instruction with a positive edge at input ACK_REI.

 **WARNING**

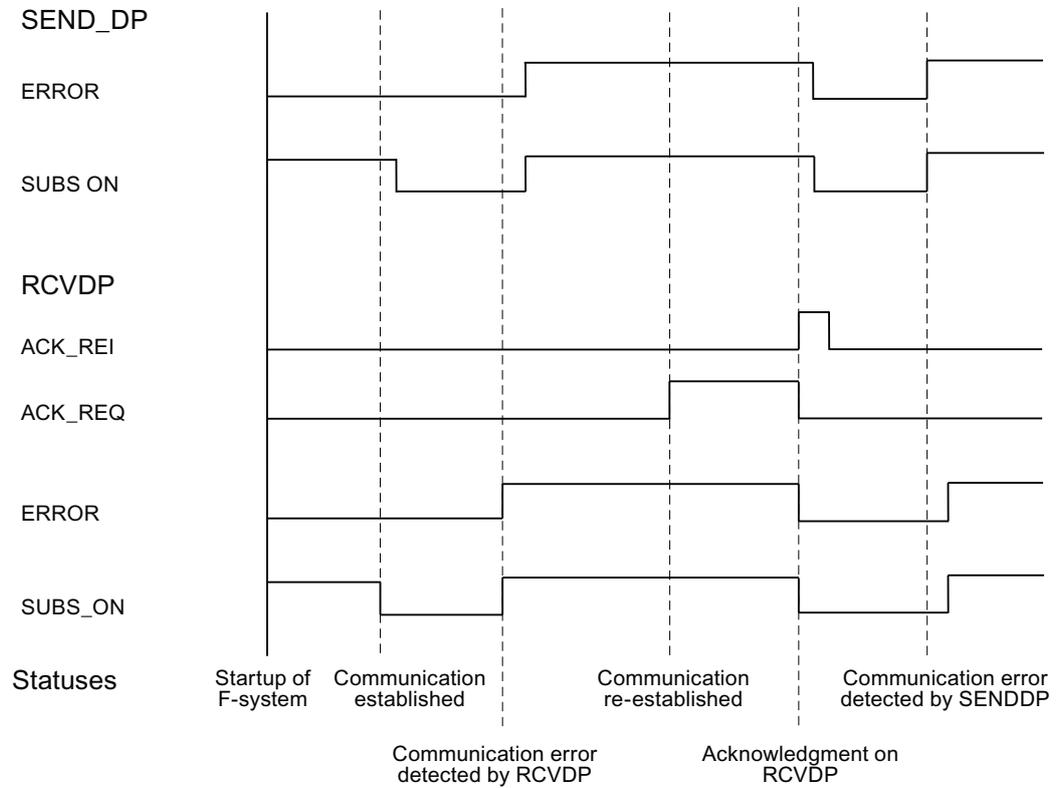
For the user acknowledgment, you must interconnect input ACK_REI with a signal generated by the operator input.

An interconnection with an automatically generated signal is not permitted. (S040)

Note that output ERROR (1=communication error) for a communication error will not be set unless communication between the connection partners (SENDDP and RCVDP instructions) has been previously established. If communication cannot be established after startup of the sending and receiving F-systems, check the configuration of the safety-related CPU-CPU communication, the parameter assignment of the SENDDP and RCVDP instructions, and the bus connection. You also obtain information on possible error causes by evaluating outputs RET_DPRD and RETDP_WR.

In general, always evaluate RET_DPRD and RETDP_WR, since it is possible that only one of the two outputs will contain error information.

Timing diagrams SENDDP/RCVDP



Output DIAG

In addition, non-fail-safe information about the type of communication errors that occurred is provided at output DIAG of the SENDDP and RCVDP instructions for service purposes.

You can read out this information by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program. DIAG bits are saved until you acknowledge the errors at input ACK_REI of the RCVDP instruction.

Structure of DIAG of the SENDDP/RCVDP instructions

Bit no.	Assignment	Possible error causes	Remedies
Bit 0	Reserved	—	—
Bit 1	Reserved	—	—
Bit 2	Reserved	—	—
Bit 3	Reserved	—	—
Bit 4	Timeout of SENDDP/RCVDP detected	Interference in bus connection to partner F-CPU.	Check bus connection and ensure that no external interference sources are present.
		Monitoring time setting for F-CPU and partner F-CPU is too low.	Check assigned monitoring time TIMEOUT for SENDDP and RCVDP of both F-CPU's. If necessary, set a higher value. Recompile safety program
		DP/DP coupler or PN/PN coupler configuration is invalid.	Check DP/DP coupler or PN/PN coupler configuration.
		Internal fault of DP/DP coupler or PN/PN coupler	Replace DP/DP coupler or PN/PN coupler
		CP in STOP mode, or internal fault in CP	Switch CP to RUN mode, check diagnostic buffer of CP, and replace CP, if necessary
		F-CPU/partner F-CPU in STOP mode, or internal fault in F-CPU/partner F-CPU	Switch F-CPU's to RUN mode, check diagnostic buffer of F-CPU's, and replace F-CPU's, if necessary
Bit 5	Sequence number error of SENDDP/RCVDP detected	See description for bit 4	See description for bit 4
Bit 6	CRC error of SENDDP/RCVDP detected	See description for bit 4	See description for bit 4
Bit 7	Reserved	—	—

See also

- Configuring and Programming Communication (Page 111)
- Correctness of the communication configuration (Page 208)
- Implementation of user acknowledgment (Page 99)
- Safety-related IO controller-IO controller communication (Page 114)
- Safety-related master-master communication (Page 122)
- Safety-related communication between I/O-controller and I-device (Page 130)
- Safety-related master-I-slave communication (Page 136)
- Safety-related IO Controller-I-slave communication (Page 154)
- Communication with S7 Distributed Safety via PN/PN coupler (IO Controller-IO Controller communication) (Page 162)
- Communication with S7 Distributed Safety via DP/DP coupler (master-master communication) (Page 163)

13.3.15.2 S7 communication

SENDS7 and RCVS7: Communication via S7 connections (STEP 7 Safety Advanced V11)

Introduction

You use the SENDS7 and RCVS7 instructions for fail-safe sending and receiving of data using S7 connections.

Note

In *STEP 7 Safety Advanced V11*, S7 connections are generally permitted over Industrial Ethernet only.

Safety-related communication via S7 connections is possible from and to F-CPU with PROFINET interface or S7-400 F-CPU with PROFINET-capable CPs. See also Auto-Hotspot.

Description

The SENDS7 instruction sends the send data contained in an F-communication DB to the F-communication DB of the associated RCVS7 instruction of another F-CPU in a fail-safe manner using an S7 connection.

Every call of this instruction must be assigned a data area in which the instruction data are stored. The "Call options" dialog is automatically opened for this reason when the instruction is inserted in the program; in it you can create a data block (single instance) (e.g., SENDS7_DB_1) or a multi-instance (e.g., SENDS7_Instance_1) for this instruction. Once you have created the data block, you will find the new data block in the project tree in the "STEP 7 Safety" folder under "Program blocks > System blocks" or the multi-instance as a local tag in the "Static" section of the block interface. For more information, refer to the help on *STEP 7 Professional*.

Enable input "EN" and enable output "ENO" cannot be connected. The instruction is therefore always executed (regardless of the signal state at enable input "EN").

Information on the F-communication DB is contained in "Safety-related communication via S7 connections (Page 155)".

An F-communication DB is an F-DB for safety-related CPU-CPU communication with special properties. You must specify the numbers of the F-communication DBs at inputs SEND_DB and RCV_DB of instructions SENDS7 and RCVS7.

The operating mode of the F-CPU with the SENDS7 instruction is provided at output SENDMODE of the RCVS7 instruction. If the F-CPU with the SENDS7 instruction is in deactivated safety mode, then output SENDMODE = 1.

To reduce the bus load, you can temporarily shut down communication between the F-CPU at input EN_SEND of the SENDS7 instruction. To do so, supply input EN_SEND with "0" (default = "1"). In this case, send data are no longer sent to the F-communication DB of the associated RCVS7 instruction, and the receiver provides fail-safe values for this period of time (initial values in its F-communication DB). If communication was already established between the partners, a communication error is detected.

You must specify the local ID - from the perspective of the F-CPU - of the S7 connection (from the connection table in the network view) at input ID of the SENDS7 instruction.

Communication between F-CPU's takes place in the background by means of a special safety protocol. You must define a communication relationship between an SENDS7 instruction in one F-CPU and a communication relationship between an RCVS7 instruction and the other F-CPU by assigning an odd number at input R_ID (of the SENDS7 and RCVS7 instructions). Associated SENDS7 and RCVS7 instructions receive the same value for R_ID.

 **WARNING**

The value for each address relationship (input parameter R_ID; data type: DWORD) is user-defined; however, it must be an odd number and be unique from all other safety-related communication connections in the network. The value R_ID + 1 is internally assigned and must not be used.

You must supply inputs ID and R_ID with constant values when calling the instruction. Direct read or write access to the associated instance DB is not permitted in the safety program! (S020)

Note

A separate instance DP must be used for each call of the SENDS7 and RCVS7 instructions within a safety program. You must not declare and call these instructions as multi-instances.

The input and output parameters of the RCVS7 instruction must not be initialized with temporary or static local data of the main safety block.

The input parameters of the RCVS7 instruction must not be initialized with output parameters (using fully qualified DB accesses) of a RCVS7 or RCVDP instruction called in a preceding network.

You must not use an actual parameter for an output parameter of an RCVS7 instruction, if it is already being used for an input parameter of the same or another RCVS7 or RCVDP instruction. The F-CPU can go to STOP if this is not observed. One of the following diagnostic events is then entered in the diagnostic buffer of the F-CPU:

- "Data corruption in the safety program prior to output to F-I/O"
- "Data corruption in the safety program prior to output to partner F-CPU"
- "Safety program: internal CPU fault; internal error information: 404"

Note

You must not program any SENDS7 instruction between a JMP or JMPN instruction and the associated destination network of the JMP or JMPN instruction.

You must not program a RET instruction prior to a SENDS7 instruction.

Parameters of the SENDS7 instruction

Parameter	Declaration	Data type	Description
SEND_DB	Input	BLOCK_DB	Number of F-communication DB
TIMEOUT	Input	TIME	Monitoring time in ms for safety-related communication (see also Monitoring and response times (Page 523))
EN_SEND	Input	BOOL	1= Send enable
ID	Input	WORD	Local ID of the S7 connection
R_ID	Input	DWORD	Network-wide unique value for an address relationship between a SENDS7 instruction and a RCVS7 instruction
ERROR	Output	BOOL	1=Communication error
SUBS_ON	Output	BOOL	1=Receiving block outputs fail-safe values
STAT_RCV	Output	WORD	Status parameter STATUS of the URCV instruction (You can find a description of error codes in the online help for the URCV instruction ("Communication > S7 Communication"))
STAT_SND	Output	WORD	Status parameter STATUS of the USEND instruction (You can find a description of error codes in the online help for the USEND instruction ("Communication > S7 Communication"))
DIAG	Output	BYTE	Service information

Parameters of the RCVS7 instruction

Parameter	Declaration	Data type	Description
ACK_REI	Input	BOOL	Acknowledgment for reintegration of send data after communication error
RCV_DB	Input	BLOCK_DB	Number of F-communication DB
TIMEOUT	Input	TIME	Monitoring time in ms for safety-related communication (see also Monitoring and response times (Page 523))
ID	Input	WORD	Local ID of the S7 connection
R_ID	Input	DWORD	Network-wide unique value for an address relationship between a SENDS7 instruction and a RCVS7 instruction
ERROR	Output	BOOL	1=Communication error
SUBS_ON	Output	BOOL	1=Fail-safe values are output
ACK_REQ	Output	BOOL	1=Acknowledgment for reintegration of send data required
SENDMODE	Output	BOOL	1=F-CPU with the SENDS7 instruction in deactivated safety mode
STAT_RCV	Output	WORD	Status parameter STATUS of the URCV instruction (You can find a description of error codes in the online help for the URCV instruction ("Communication > S7 Communication"))
STAT_SND	Output	WORD	Status parameter STATUS of the USEND instruction (You can find a description of error codes in the online help for the USEND instruction ("Communication > S7 Communication"))
DIAG	Output	BYTE	Service information

Startup characteristics

After the sending and receiving F-systems are started up, communication must be established between the connection partners for the first time (SENDS7 and RCVS7 instructions). The receiver (RCVS7 instruction) provides fail-safe values for this time period (initial values in its F-communication DB).

The SENDS7 and RCVS7 instructions signal this at output SUBS_ON with 1. Output SENDMODE (RCVS7 instruction) has a default of 0 and is not updated, as long as output SUBS_ON = 1.

Behavior in the event of communication errors

If a communication error occurs, for example, due to a signature error (CRC) or when monitoring time TIMEOUT expires, outputs ERROR and SUBS_ON = 1 are set. The receiver (RCVS7 instruction) then provides fail-safe values (initial values in its F-communication DB). Output SENDMODE is not updated while output SUBS_ON = 1.

The send data present in the F-communication DB (SENDS7 instruction) are not output before the communication error is no longer detected ((ACK_REQ = 1)) and you acknowledge (Page 99) with a positive edge at input ACK_REI of the RCVS7 instruction.



WARNING

For the user acknowledgment, you must interconnect input ACK_REI with a signal generated by the operator input.

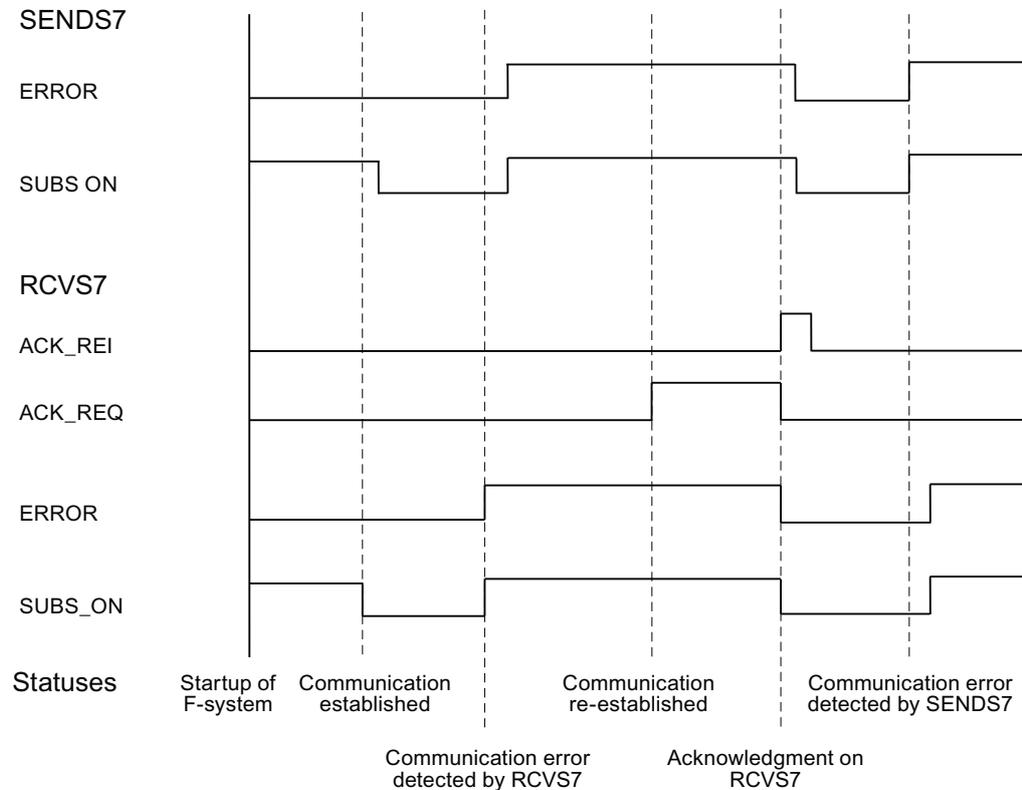
An interconnection with an automatically generated signal is not permitted. (S040)

Note that output ERROR (1=communication error) will be set for the first time on a communication error if communication has already been established between the connection partners (SENDS7 and RCVS7 instructions). If communication cannot be established after startup of the sending and receiving F-Systems, check the configuration of the safety-related CPU-CPU communication, parameter assignment of the SENDS7 and RCVS7 instructions, and the bus connection. You can also receive information on possible error causes by evaluating the STAT_RCV and STAT_SND outputs.

In general, always evaluate STAT_RCV and STAT_SND, since it is possible that only one of the two outputs will contain error information.

If one of the DIAG bits is set at output DIAG, also check whether the length and structure of the associated F-communication DB on both the sending and receiving ends match.

Timing diagrams SENDS7 and RCVS7



Output DIAG

Non-fail-safe information on the type of communication errors that have occurred is made available at output DIAG for service purposes. You can read out this information by means of operator control and monitoring systems or, if applicable, you can evaluate it in your standard user program. DIAG bits are saved until you acknowledge them at input ACK_REI of the associated RCVS7 instruction.

Structure of DIAG

Bit no.	Assignment of SENDS7 and RCVS7	Possible error causes	Remedies
Bit 0	Reserved	—	—
Bit 1	Reserved	—	—
Bit 2	Reserved	—	—
Bit 3	Reserved	—	—
Bit 4	Timeout detected by SENDS7 and RCVS7	Fault in bus connection to partner F-CPU	Check bus connection and ensure that no external fault sources are present.
		Monitoring time setting for F-CPU and partner F-CPU is too low	Check assigned monitoring time TIMEOUT for SENDS7 and RCVS7 of both F-CPU. If possible, set a higher value. Recompile safety program
		CPs in STOP mode, or internal fault in CPs	<ul style="list-style-type: none"> • Switch CPs to RUN mode • Check diagnostic buffer of CPs • Replace CPs, if necessary
		F-CPU/partner F-CPU in STOP mode, or internal fault in F-CPU/partner F-CPU	<ul style="list-style-type: none"> • Switch F-CPU to RUN mode • Check diagnostic buffer of F-CPU • Replace F-CPU, if necessary
		Communication was shut down with EN_SEND = 0.	Enable communication again at the associated SENDS7 with EN_SEND = 1
		S7 connection has changed, the IP address of the CP has changed, for example	Recompile the safety programs and download them to the F-CPU
Bit 5	Sequence number error detected by SENDS7 and RCVS7	See description for bit 4	See description for bit 4
Bit 6	CRC error detected by SENDS7 and RCVS7	See description for bit 4	See description for bit 4
Bit 7	RCVS7: Communication cannot be established	Configuration of the safety-related CPU-CPU communication is incorrect, parameter assignment of the SENDS7 and RCVS7 instructions is incorrect See also description for Bit 4	Check configuration of the safety-related CPU-CPU communication, parameter assignment of the SENDS7 and RCVS7 instructions is incorrect See also description for Bit 4
	SENDS7: Reserved	—	—

See also

Configuration (Page 25)

Monitoring and response times

Introduction

In the following, you will learn:

- Which F-specific monitoring times you must configure
- Which rules must be followed when specifying monitoring times
- Where you enter the F-specific monitoring times
- Which rules must be followed with regard to the maximum response time of a safety function

Support for calculations

An Excel file is available on the Internet (<http://support.automation.siemens.com/WW/view/en/49368678/133100>) to assist you in calculating approximately the runtimes of the F-runtime groups, the minimum F-specific monitoring times, and the maximum response times of your F-System.

Additional information

The monitoring and response times for the standard portion are calculated in *SIMATIC Safety* in exactly the same way as for standard S7-300 and S7-400 automation systems and are not addressed here. For a description of this calculation, refer to the *hardware manuals for the CPUs*.

A.1 Configuring the monitoring times

Monitoring times to be configured

You must configure the following monitoring times:

Monitoring...	Setting...	Parameters	See
F-cycle time of the F-runtime groups that contain the safety program	in <i>Safety Administration Editor</i> : <ul style="list-style-type: none"> Dialog for definition of an F-runtime group 	Maximum cycle time of the F-runtime group	Procedure for Defining an F-Runtime Group (Page 66)
of the safety-related communication between F-CPU and F-I/O via PROFIsafe (PROFIsafe monitoring time)	in the <i>hardware and network editor</i> : <ul style="list-style-type: none"> Centrally when configuring the F-CPU; properties of the F-CPU; or when configuring the F-I/O; properties of the F-I/O 	F-monitoring time F_WD_TIME	Configuring the F-CPU (Page 29) Configuring the F-I/O (Page 33) Configuring fail-safe DP standard slaves and fail-safe standard I/O devices (Page 38)
of the safety-related CPU-CPU communication	Parameter "TIMEOUT" of the instructions: <ul style="list-style-type: none"> SENDDP; RCVDP; SENDS7; RCVS7 	TIMEOUT	Communication (Page 510)

You do not have to configure the monitoring time for safety-related communication between F-runtime groups.

Rules for configuring monitoring times

When configuring monitoring times, you must take into account the availability as well as the safety of the F-system as follows:

- **Availability:** To ensure that the time monitoring is not triggered when there is no error, the monitoring times selected must be sufficiently long.
- **Safety:** To ensure that the process safety time is not exceeded for the process, the monitoring times selected must be sufficiently short.

 WARNING
<p>It can only be ensured (from a fail-safe standpoint) that a signal level to be transferred will be captured on the sender side and transferred to the receiver if the signal level is pending for at least as long as the assigned F-monitoring time. (S018)</p>

General procedure for configuring monitoring times

Use the following procedure for configuring monitoring times:

1. Configure the standard system.
Refer to the applicable *hardware manuals* and *help on STEP 7 Professional* for the necessary information.
2. Configure the specific monitoring times of the F-System with respect to availability. You use the Excel file for response time calculation (<http://support.automation.siemens.com/WW/view/en/49368678/133100>) to calculate the approximate minimum monitoring time.
3. Use the Excel file for response time calculation to calculate the maximum response time and then verify that the process safety time of the process is not exceeded. If necessary, you must reduce the specific monitoring times of the F-System.

A.1.1 Minimum Monitoring Time for F-Cycle Time

Parameter "Maximum cycle time of the F-runtime group"

You configure the monitoring time of the F-cycle time in the *Safety Administration Editor* in the work area for definition of the F-runtime group (Page 65).

The specified maximum cycle time of the F-runtime group must be high enough to prevent tripping the F-cycle time monitoring when no faults are present, thus causing the F-CPU to go to STOP.

Use the Excel file for response time calculation (<http://support.automation.siemens.com/WW/view/en/49368678/133100>) to determine the minimum monitoring time of the F-cycle time.

Note also the comments in the Excel file.

A.1.2 Minimum monitoring time for safety-related communication between F-CPU and F-I/O

Parameter "F-monitoring time"

You have two options for configuring the monitoring time of safety-related communication between the F-CPU and F-I/O:

- Centrally in the *hardware and network editor* during parameter assignment of the F-CPU (Page 29); in the properties of the F-CPU, or
- During parameter assignment of the F-I/O (Page 33) in the *hardware and network editor*, in the properties of the F-I/O

"F-monitoring time" = PROFIsafe monitoring time T_{PSTO}

The specified PROFIsafe monitoring time T_{PSTO} must be high enough to prevent tripping the F-cycle time monitoring when no faults are present.

Use the Excel file for response time calculation

(<http://support.automation.siemens.com/WW/view/en/49368678/133100>) available for *SIMATIC Safety* to calculate the minimum monitoring time for safety-related communication between the F-CPU and F-I/O.

Note also the comments in the Excel file.

Check to determine whether configured PROFIsafe monitoring time is too short

Note

During commissioning of the F-system, you can perform a check while safety mode is active to determine whether the configured PROFIsafe monitoring time is too short.

This check of the PROFIsafe monitoring time is useful if you want to ensure that the configured monitoring time exceeds the minimum monitoring time by a sufficient amount. In this way, you can avoid the possible occurrence of sporadic monitoring time errors.

Procedure:

1. Insert an F-I/O (one that will not be needed later for system operation).
2. Assign a shorter monitoring time for this F-I/O than for the F-I/O of the system.
3. If the additional F-I/O fails and the "Monitoring time for safety message frame exceeded" diagnostic is signaled, you have fallen below the minimum possible PROFIsafe monitoring time.
4. Increase the monitoring time for the additional F-I/O just to the point where it no longer fails. This monitoring time corresponds approximately to the minimum possible monitoring time.

Conditions:

The F-I/O to be inserted additionally and the F-I/O whose PROFIsafe monitoring time is to be checked must have the following properties in common:

- They must be inserted in the same rack
- They must be nodes in the same subnet

Tip:

It may be useful to leave the additional F-I/O in place for systems that will be modified or expanded during operation after commissioning. This F-I/O will then provide an early warning in the event of changes in the time behavior, enabling you to avoid a process shutdown triggered by the F-I/O in the process.

A.1.3 Minimum monitoring time of safety-related CPU-CPU communication

Parameter TIMEOUT at SENDDP and RCVDP or SENDS7 and RCVS7

The same monitoring time is used for the time monitoring in the SENDDP and RCVDP (Page 510) or SENDS7 and RCVS7 (Page 517) instructions. You must assign the time monitoring for both instructions in the TIMEOUT parameter.

The specified monitoring time TIMEOUT must be high enough to prevent tripping the monitoring when no faults are present.

Use the Excel file for response time calculation (<http://support.automation.siemens.com/WW/view/en/49368678/133100>) available for *SIMATIC Safety* to determine the minimum value for TIMEOUT.

Note also the comments in the Excel file.

A.1.4 Monitoring Time for Safety-Related Communication between F-Runtime Groups

The monitoring time for safety-related communication between F-runtime groups is determined automatically from the values for the "Maximum cycle time of F-runtime group" (work area for Definition of the F-runtime group (Page 65) in the *Safety Administration Editor*).

Monitoring time = (maximum cycle time of the 1st F-runtime group) + (maximum cycle time of the 2nd F-runtime group)

A.2 Response Times of Safety Functions

Definition of response time

The response time is the time from detection of an input signal until the linked output signal changes.

Fluctuation range

The actual response time lies between a minimum response time and maximum response time. You must always take the maximum response time into account in your system configuration.

Rules for maximum response time of a safety function

The maximum response time of a safety function must be less than the process safety time of the process.

Definition of process safety time of a process

The process safety time of a process is the time interval within which the process can be left on its own without causing injury to operating personnel or damage to the environment.

Within the process safety time, any type of F-system process control is tolerated. That is, during this time, the F-system can control its process incorrectly or it can even execute no control at all. The process safety time of a process depends on the process type and must be determined on a case-by-case basis.

Procedure for response time calculation

The Excel file for response time calculation (<http://support.automation.siemens.com/WW/view/en/49368678/133100>) is available for calculating the maximum response time of a safety function.

Use the Excel file to calculate the approximate maximum response time of the safety function and then check that the process safety time of the process is not exceeded.

If necessary, you must reduce the specific monitoring times of the F-system (see Minimum monitoring time for safety-related communication between F-CPU and F-I/O (Page 526)).

Checklist

Life cycle of fail-safe automation systems

The table below contains a checklist summarizing all activities in the life cycle of a fail-safe SIMATIC Safety system, including requirements and rules that must be observed for each activity.

Checklist

Legend:

- Stand-alone chapter references refer to this documentation.
- "*F-SMs Manual*" refers to the Automation System S7-300, ET 200M Distributed I/O System, Fail-Safe Signal Modules (<http://support.automation.siemens.com/WW/view/en/19026151>) manual.
- "*F-Modules Manual*" refers to the ET 200S Distributed I/O System, Fail-Safe Modules (<http://support.automation.siemens.com/WW/view/en/27235629>) manual.
- "*ET 200eco Manual*" refers to the ET 200eco Distributed I/O Station, Fail-Safe I/O Module (<http://support.automation.siemens.com/WW/view/en/19033850>) manual.
- "*ET 200pro Manual*" refers to the ET 200pro Distributed I/O System, Fail-Safe I/O Module (<http://support.automation.siemens.com/WW/view/en/22098524>) manual.
- "*ET 200iSP Manual*" refers to the ET 200iSP Distributed I/O Device, Fail-Safe Modules (<http://support.automation.siemens.com/WW/view/en/47357221>) manual.

Phase	Note the following	Reference	Check
Planning			
Requirement: "Safety requirements specification" available for the intended application	Process-dependent	—	
Specification of system architecture	Process-dependent	—	
Assignment of functions and subfunctions to system components	Process-dependent	Product Overview (Page 17)	
Selection of sensors and actuators	Requirements for actuators	<i>F-SMs Manual</i> , Chapter 6.5; <i>F-Modules Manual</i> , Chapter 4.5; <i>ET 200eco Manual</i> , Chapter 5.5 <i>ET 200pro Manual</i> , Chapter 4.4 <i>ET 200S Manual</i> , Section 4.5	
Specification of required safety properties for individual components	IEC 61508:2010	—	

Phase	Note the following	Reference	Check
Configuration			
Installing the optional package	Requirements for installation	Installation/uninstallation of the STEP 7 Safety Advanced V11 optional package (Page 21)	
Selection of S7 components	Descriptions of configuration	Product Overview (Page 17); <i>F-SMs Manual</i> , Chapter 3; <i>F-Modules Manual</i> , Chapter 3; <i>ET 200eco Manual</i> , Chapter 3 <i>ET 200pro Manual</i> , Chapter 2 <i>ET 200iSP Manual</i> , Section 3	
Configuration of hardware	<ul style="list-style-type: none"> • Description of F-systems • Verification of utilized hardware components based on Annex 1 of Report in the Certificate 	Configuration (Page 25); Annex 1 of Report on the Certificate	
Configuration of F-CPU	<ul style="list-style-type: none"> • Protection level, "Write protection for F-blocks" • Password • F-capability activated • Definition/setting of F-specific parameters • Cycle time for the F-runtime group in which the safety program is to be executed, defined in accordance with the requirements and safety regulations - same as with standard system 	Configuring the F-CPU (Page 29); <i>Standard S7-300</i> ; <i>Standard S7-400</i> ; <i>IM 151-7 CPU</i> <i>IM 151-8 PN/DP CPU</i> ; <i>IM 154-8 CPU</i> Monitoring and response times (Page 523)	
Configuration of F-I/O	<ul style="list-style-type: none"> • Settings for safety mode • Setting of passivation type • Configuring monitoring times • Defining type of sensor interconnection/evaluation • Defining diagnostic behavior • Special F-parameters • Assigning symbolic names 	Configuring the F-I/O (Page 33) ff.; Monitoring and response times (Page 523); <i>F-SMs Manual</i> , Chapters 3, 9, 10; <i>F-Modules Manual</i> , Chapters 2.4, 7; <i>ET 200eco Manual</i> , Chapters 3, 8; <i>ET 200pro Manual</i> , Chapters 2.4, 8; <i>ET 200iSP Manual</i> , Chapters 2.4, 7, 8	

Phase	Note the following	Reference	Check
Programming			
Defining program design and structure	<ul style="list-style-type: none"> Follow warnings and notes on programming 	Overview of Programming (Page 55); Structure of the safety program (Page 56); Programming startup protection (Page 76);	
Creating the F-runtime groups	<ul style="list-style-type: none"> Assignment of F-FB or F-FC as main safety block to the calling block. Setting maximum cycle time for the F-runtime group in accordance with requirements (dependent on process and safety regulations) Creating DB for F-runtime group communication Call of main safety blocks directly in OBs (e.g., OB 35), FBs, or FCs 	Defining F-Runtime Groups (Page 65) Monitoring and response times (Page 523)	
Creating/inserting the F-blocks	<ul style="list-style-type: none"> Generating, editing, and saving F-FBs, F-FCs, and F-DBs in accordance with the requirements of the program structure Description: <ul style="list-style-type: none"> F-I/O access Passivation and reintegration of F-I/O Inserting F-blocks from shared libraries Safety-related CPU-CPU communication Communication with the standard user program 	Creating F-blocks in FBD / LAD (Page 73) F-I/O access (Page 79) Implementation of user acknowledgment (Page 99) Use libraries (Page 75) Configuring and Programming Communication (Page 111) Data exchange between standard user program and safety program (Page 105)	
Compiling the safety program	—	Compiling the safety program (Page 169)	
Implementing call of safety program	Check whether the main safety block is called directly in OBs (e.g., OB 35), FBs, or FCs.	Defining F-Runtime Groups (Page 65)	
Installation			
Hardware configuration	Description of <ul style="list-style-type: none"> Installation Wiring 	Overview of Configuration (Page 25); Particularities for configuring the F-System (Page 28); <i>F-SMs Manual</i> , Chapters 5, 6; <i>F-Modules Manual</i> , Chapters 3, 4; <i>ET 200eco Manual</i> , Chapters 3, 4; <i>ET 200pro Manual</i> , Chapters 2, 3; <i>ET 200iSP Manual</i> , Chapters 3, 4	

Phase	Note the following	Reference	Check
Commissioning, Testing			
Switching on	Description of commissioning – same as in standard	Standard S7-300; standard S7-400	
Downloading safety program and standard user program	Description <ul style="list-style-type: none"> • Downloading • Program identification • Comparing safety programs 	Downloading the Safety Program (Page 171) Comparing Safety Programs (Page 183)	
Testing safety program	<ul style="list-style-type: none"> • Description of deactivation of safety mode • Procedures for changing safety program data 	Function Test of Safety Program and Protection through Program Identification (Page 178); Testing the Safety Program (Page 193); Deactivating Safety Mode (Page 190)	
Changing the safety program	Description <ul style="list-style-type: none"> • Deactivation of safety mode • Changing of safety program 	Modifying the safety program in RUN mode (Page 197); Deactivating Safety Mode (Page 190); Deleting the Safety Program (Page 199);	
Testing the safety-related parameters	Description of configuration	Printing project data (Page 187); <i>F-SMs Manual</i> , Chapters 4, 9, 10; <i>F-Modules Manual</i> , Chapters 2.4, 7; <i>ET 200eco Manual</i> , Chapters 3, 8; <i>ET 200pro Manual</i> , Chapters 2.4, 8; <i>ET 200iSP Manual</i> , Chapters 2.4, 7, 8	
System acceptance test			
Acceptance test	<ul style="list-style-type: none"> • Description and notes on acceptance test • Printouts 	System Acceptance Test (Page 201)	
Operation, maintenance			
General operation	Notes on operation	Notes on Safety Mode of the Safety Program (Page 213)	
Access protection	—	Access protection (Page 49)	
Diagnostics	Responses to faults and events	Guide to Diagnostics (Page 216)	
Replacement of software and hardware components	Description <ul style="list-style-type: none"> • Module replacement • Update of operating systems of F-CPU – same as in standard • Update of SW components Notes <ul style="list-style-type: none"> • Operating system update of IMs • Preventive maintenance 	Replacing Software and Hardware Components (Page 214); F-I/O access (Page 79); Help on <i>STEP 7 Professional</i>	
Uninstallation, disassembly	<ul style="list-style-type: none"> • Notes for uninstalling software components • Notes for disassembling modules 	Installation/uninstallation of the STEP 7 Safety Advanced V11 optional package (Page 21); Replacing Software and Hardware Components (Page 214);	

Glossary

Access protection

→ Fail-safe systems must be protected against dangerous, unauthorized access. Access protection for F-systems is implemented by assigning two passwords (one for the → F-CPU, and one for the → safety program).

Automatically generated F-blocks

→ F-blocks that are automatically generated and, if necessary, opened when the → safety program is compiled, in order to generate an executable safety program from the safety program programmed by the user.

Category

Category as defined by EN 954-01

With SIMATIC Safety, use in → safety mode up to category 4 is possible.

Channel fault

Channel-related fault, such as a wire break or short circuit.

Collective F-signatures

The collective F-signatures uniquely identify a particular state of the → safety program and the safety-related parameters of the F-CPU and F-I/O. They are important for the preliminary acceptance test of the safety program, e.g., by → experts.

CRC

Cyclic Redundancy Check → CRC signature

CRC signature

The validity of the process data in the → safety message frame, the correctness of the assigned address relationships, and the safety-related parameters are validated by means of a CRC signature contained in the safety message frame.

DB for F-runtime group communication

→ F-DB for safety-related communication between F-runtime groups of a safety program.

Deactivated safety mode

Temporary deactivation of → safety mode for test purposes, commissioning, etc.

The following actions are possible only in deactivated safety mode:

- Downloading changes of the → safety program to the → F-CPU during ongoing operation (in RUN mode)
- Test functions such as "Modify" or other write access to data of the → safety program (with limitations)

Whenever safety mode is deactivated, the safety of the system must be ensured by other organizational measures, such as operation monitoring and manual safety shutdown.

Depassivation

→ Reintegration

Discrepancy analysis

Discrepancy analysis for equivalence or nonequivalence is used for fail-safe inputs to determine errors based on the time characteristic of two signals with the same functionality. The discrepancy analysis is initiated when different levels are detected in two associated input signals (for nonequivalence testing, when identical levels are detected). On expiration of an assignable period (→ discrepancy time), a check is made to determine whether the difference in levels has disappeared (for nonequivalence testing, whether the identicalness of the levels has disappeared). If not, there is a discrepancy error. The discrepancy analysis is performed between the two input signals of the 1oo2 sensor evaluation (→ sensor evaluation) in the fail-safe input.

Discrepancy time

Assignable time for the → discrepancy analysis. If the discrepancy time is set too high, the fault detection time and → fault reaction time are prolonged unnecessarily. If the discrepancy time is set too low, availability is decreased unnecessarily because a discrepancy error is detected when, in reality, no error exists.

DP/DP coupler

Device for coupling two PROFIBUS DP subnets required for master-master communication between → safety programs in different → F-CPU's in SIMATIC Safety and S7 Distributed Safety.

Expert

The acceptance of a system, i.e., the safety-related acceptance test of the system, is usually carried out by an independent expert (for example, from TÜV).

Fail-safe DP standard slaves

Fail-safe DP standard slaves are standard slaves that are operated on PROFIBUS with the DP protocol. They must operate in accordance with IEC 61784-1: 2007 (fieldbus profile) and the PROFIsafe bus profile in accordance with IEC 61784-3-3: 2007. A GSD file is used for their configuration.

Fail-safe I/O modules

ET 200eco modules that can be used for safety-related operation (-> safety mode). These modules are provided with integrated → safety functions. They operate in accordance with IEC 61784-1: 2007 (fieldbus profile) and the PROFIsafe bus profile in accordance with IEC 61784-3-3: 2007.

Fail-safe modules

ET 200S, ET 200pro, and ET 200iSP modules that can be used for safety-related operation (→ safety mode) in the ET 200S, ET 200pro, or ET 200iSP distributed I/O systems. These modules are provided with integrated → safety functions. They operate in accordance with IEC 61784-1: 2007 (fieldbus profile) and CP 3/3 and the PROFIsafe bus profile in accordance with IEC 61784-3-3: 2007.

Fail-safe standard I/O devices

Fail-safe standard I/O devices are standard devices that are operated on PROFINET with the IO protocol. They must operate in accordance with IEC 61784-1: 2007 (fieldbus profile) and the PROFIsafe bus profile in accordance with IEC 61784-3-3: 2007 in V2 mode. A GSD file is used for their configuration.

Fail-Safe Systems

Fail-safe systems (F-systems) are systems that remain in a safe state or immediately switch to another safe state as soon as particular failures occur.

F-Attribute

All → F-blocks that belong to a → safety program are provided with an F-attribute. Only the blocks of the → safety program have the F-attribute after the → safety program is successfully compiled.

Fault reaction function

→ User safety function

Fault reaction time

The maximum fault reaction time for an F-system specifies the time between the occurrence of any error and a safe reaction at all affected fail-safe outputs.

F-blocks

The following fail-safe blocks are designated as F-blocks:

- those created by the user in LAD or FBD
- those created by the user as → F-DBs
- those selected by the user from a shared library
- those added automatically in the → safety program (→ F-SBs, → automatically generated F-blocks, → F-shared DB, → F-I/O DBs; instance DBs of F-FBs)

All F-blocks are shown in yellow in the project tree.

F-CALL

"F-call blocks" for the → safety program in *S7 Distributed Safety*.

F-communication DBs

Fail-safe data blocks for the safety-related CPU-CPU communication via S7 connections.

F-CPU

An F-CPU is a central processing unit with fail-safe capability that is approved for use in SIMATIC Safety and in which a → safety program can run in addition to the → standard user program.

F-DBs

Optional fail-safe data blocks that can be read-/write-accessed from anywhere within the safety program (exception: DBs for F-runtime group communication).

F-FBs

Fail-safe function blocks (with instance DBs), in which the user programs the → safety program in FBD or LAD.

F-FCs

Fail-safe FCs, in which the user programs the → safety program in → FBD or → LAD.

F-I/O

Collective name for fail-safe inputs and outputs available in *SIMATIC S7* for integration in SIMATIC Safety, among others. The following are available:

- → ET 200eco fail-safe I/O module
- → S7-300 fail-safe signal modules
- → Fail-safe modules for ET 200S
- → Fail-safe modules for ET 200pro
- → Fail-safe modules for ET 200iSP
- → Fail-safe DP standard slaves
- → Fail-safe standard I/O devices

F-I/O DB

Fail-safe data block for an → F-I/O in *STEP 7 Safety Advanced V11*. An F-I/O DB is automatically created for each F-I/O when the F-I/O is configured in the *hardware and network editor*. The F-I/O DB contains tags that the user can or must evaluate or write in the safety program as follows:

- For reintegration of F-I/O after communication errors, F-I/O faults, or channel faults
- If F-I/O are to be passivated as a result of particular safety program statuses (for example, group passivation)
- For reassignment of parameters for fail-safe DP standard slaves/standard I/O devices or enabling HART communication for the F-I/O with the corresponding functionality
- For evaluation of whether fail-safe values or process data are output

F-I/O faults

Module-related F-I/O fault, such as a communication error or parameter assignment error

F-modules

-> Fail-safe modules

F-runtime group

The -> safety program consists of one or two F-runtime groups. An F-runtime group is a logical construct of several associated → F-blocks. It is generated internally by the F-system. An F-runtime group consists of the following F-blocks:

→ Main safety block, if necessary → F-FBs/ → F-FCs, if necessary → F-DBs, → F-I/O DBs, F-blocks of shared libraries, instance DBs, → F-SBs, and → automatically generated F-blocks.

F-SBs

Fail-safe system blocks that are automatically inserted and called when the → safety program is compiled in order to generate an executable safety program from the user's safety program.

F-shared DB

Fail-safe data block that contains all of the shared data of the → safety program and additional information needed by the F-system. The F-shared DB is automatically inserted and expanded when the hardware configuration is compiled. Using its name F_GLOBDB, the user can evaluate certain data of the → safety program.

F-SMs

→ S7-300 fail-safe signal modules

F-systems

→ Fail-safe systems

I-device

The functionality of the "I-device" (intelligent I/O-device) of a CPU allows data exchange with an I/O-controller and thus, its use as an intelligent preprocessor of sub-processes, for example. In this case, the I-device is connected as an I/O-device to a "parent" I/O-controller.

IE/PB link

Device for coupling PROFINET IO and PROFIBUS DP systems required, among other things, for IO Controller-I-slave communication between → safety programs in different → F-CPU's in SIMATIC Safety.

i-parameter

Individual parameters of → fail-safe DP standard slaves and → fail-safe standard I/O devices

I-slave

An "I-slave (intelligent DP slave) is a signal preprocessing field device. One its features is that that the I/O area available for the DP master does not correspond to an actually available I/O, but rather an I/O area that is imaged by a preprocessing CPU.

Main safety block

"Introductory F-block" for fail-safe programming of the → safety program in *STEP 7 Safety Advanced V11*. The main safety block is an → F-FB or → F-FC that the user assigns to the → calling block (e.g., OB 35) of an → F-runtime group.

The main safety block contains the safety program and any calls of other → F-FBs/F-FCs for program structuring.

Passivation

When passivation occurs in an → F-I/O with inputs, the → F-system provides the safety program with fail-safe values (0) instead of the process data pending at the fail-safe inputs in the PII.

When passivation occurs in an F-I/O with outputs, the F-system transfers fail-safe values (0) to the fail-safe outputs instead of the output values in the PIQ provided by the safety program.

PL

Performance Level (PL) according to ISO 13849-1: 2006 or in accordance with EN ISO 13849-1: 2008

With SIMATIC Safety, use up to Performance Level (PL) e is possible in → safety mode.

PN/PN coupler

Device for coupling two PROFINET IO systems required for IO Controller-IO Controller communication between → safety programs in different → F-CPU's in SIMATIC Safety and S7 Distributed Safety.

PROFIsafe

Safety-related bus profile of PROFIBUS DP/PA and PROFINET IO according to IEC 61784-3-3: 2007 for communication between the → safety program and the → F-I/O in an → F-system.

PROFIsafe address

The PROFIsafe addresses are used to uniquely identify the source and destination. For this reason, every → F-I/O has two PROFIsafe addresses: a source address and a destination address. The PROFIsafe source address is automatically assigned and is not displayed for F-modules (except standard devices). You must configure the PROFIsafe destination address in the *hardware and network editor* and set this address on the F-I/O via a switch.

Program signature

→ Collective F-signature

Proof test interval

A component must be put into fail-free state following the proof-test interval. That is, it is replaced by an unused component or it is proven to be completely error-free.

Reintegration

The switchover from fail-safe values (0) to process data (reintegration of an → F-I/O) takes place automatically or following user acknowledgment in the F-I/O DB. The reintegration method depends on the following:

- The reason for → passivation of the F-I/O/channels of the F-I/O
- Parameter assignment in the > F-I/O DB

Following reintegration for an → F-I/O module with inputs, the process data pending at the inputs in the PII are provided again for the safety program. For an F-I/O with outputs, the F-system again transfers the output values provided in the PIQ in the safety program to the fail-safe outputs.

S7-300 fail-safe signal modules

Fail-safe signal modules of the S7-300 module range that can be used for safety-related operation (→ safety mode) as centralized modules in an S7 300 or as distributed modules in the ET 200M distributed I/O system. The fail-safe signal modules are furnished with integrated → safety functions.

S7-PLCSIM

The *S7-PLCSIM* application enables you to execute and test your program on a simulated automation system on your programming device or PC. Because the simulation takes place entirely in *STEP 7 Professional*, you do not require any hardware (CPU, I/O).

Safe state

The basic principle of the safety concept in → fail-safe systems is the existence of a safe state for all process variables. For digital → F-I/O that conform to IEC 61508:2010, this is always the value "0".

Safety function

Mechanism integrated in the → F-CPU and → F-I/O that allows them to be used in → fail-safe systems.

According to IEC 61508:2010, a function that is implemented by a safety device in order to maintain the system in the safe state or bring the system to a safe state in the event of a specific fault. (fault reaction function → user safety function)

Safety Integrity Level

Safety Integrity Level (SIL) according to IEC 61508:2010. The higher the Safety Integrity Level is, the more stringent the measures are for avoiding and controlling system faults and random hardware failures.

With SIMATIC Safety, up to Safety Integrity Level SIL3 is possible in safety mode.

Safety message frame

In → safety mode, data are transferred in a safety frame between the → F-CPU and → F-I/O, or between the F-CPU in safety-related CPU-CPU communication.

Safety mode

1. Operating mode of → F-I/O in which → safety-related communication can take place using → safety message frames.
2. Operating mode of the safety program. In safety mode of the safety program, all safety mechanisms for fault detection and reaction are activated. In safety mode, the safety program cannot be modified during operation. Safety mode can be deactivated by the user (→ deactivated safety mode).

Safety program

Safety-related user program

Safety protocol

→ Safety message frame

Safety-related communication

Safety-related communication is used to exchange fail-safe data.

Sensor evaluation

There are two types of sensor evaluation:

- 1oo1 evaluation – sensor signal is read once
- 1oo2 evaluation - sensor signal is read twice by the same → F-I/O and compared internally

Signature

→ F-collective signatures

Standard communication

Communication used to exchange non-safety-related data

Standard mode

Operating mode of → F-I/O in which → safety-related communication between the F-CPU and the F-I/O by means of → safety message frames is not possible; only → standard communication is possible in this operating mode.

Standard user program

Non-safety-related user program

Startup of F-system

When an → F-CPU is switched from STOP to RUN mode, the standard user program starts up in the normal way. When the → safety program is started up, all data blocks with an → F-attribute are initialized with the values from the load memory - as is the case with a cold restart. This means that saved error information is lost.

The → F-system performs an automation → reintegration of the → F-I/O.

User safety function

The → safety function for the process can be provided through a user safety function or a fault reaction function. The user only has to program the user safety function. In the event of an error, if the → F-system can no longer execute its actual user safety function, it executes the fault reaction function: for example, the associated outputs are deactivated, and the → F-CPU switches to STOP mode, if necessary.

Index

=
=, 379

A

Acceptance test
 of safety-related changes, 210
 of system, 201
Acceptance test for safety-relevant changes, 210
Access
 To tags of F-I/O DB, 88
Access permission
 Canceling, 54
 Setting up for the safety program, 51
 Setup for F-CPU, 54
 Validity, 51, 54
Access protection, 49
ACK_GL, 295, 451
ACK_NEC, 82
ACK_OP, 351, 506
ACK_REI, 82
ACK_REQ, 82
Acknowledgment
 Fail-safe, 351, 506
Activating
 F-capability, 29
 Safety mode, 191
Activating/deactivating F-capability, 29
Add, 317, 474
ADD, 317, 474
Address assignment
 F-destination address, 36
AND, 344, 374, 499
Approvals, 4
Assignment, 227, 379

B

Basis for PROFIsafe addresses, 30, 34
Behavior
 After communication errors, 92
 After F-I/O faults or channel faults, 94
 After startup, 90
Bit logic operation

AND, 374
Assignment, 227, 379
EXCLUSIVE OR, 377
Insert binary input, 372
Invert RLO, 226, 373
Normally closed contact, 225
Normally open contact, 224
OR, 376
Reset output, 228, 380
Reset/set flip-flop, 232, 385
Scan operand for negative signal edge, 236, 389
Scan operand for positive signal edge, 234, 387
Scan RLO for negative signal edge, 239, 393
Scan RLO for positive signal edge, 237, 391
Set output, 229, 381
Set/reset flip-flop, 230, 383
Bit memory, 105
Block size of automatically generated F-blocks, 177
BO_W, 332, 488

C

Category, 17
Change
 Acceptance test, 210
 Detecting, 210
 of the safety program in RUN mode, 197
Changing
 Data of the safety program, 193
Channel fault, 33, 94
Checking the program version, 209
Checklist, 529
CMP <, 316, 473
CMP <=, 314, 471
CMP <>, 312, 469
CMP ==, 311, 468
CMP >, 315, 472
CMP >=, 313, 470
Code review of the safety program, 202
Communication
 Monitoring time, 526, 527
 Standard user program and safety program, 105, 108
Communication error, 92, 356, 510
 SENDDP/RCVDP, 356, 510
Comparator operations
 Equal, 311, 468
 Greater or equal, 313, 470

- Greater than, 315, 472
- Less or equal, 314, 471
- Less than, 316, 473
- Not equal, 312, 469
- Comparing
 - Safety programs, 183
- Compiling
 - Hardware configuration, 169
 - Safety program, 169
- Compiling errors
 - Alarms, 170
- Completeness
 - Checking printout, 203
- Configuring
 - Fail-safe DP standard slaves, 38
 - Fail-safe I/O standard devices, 38
 - F-CPU, 29
 - F-I/O, 33
 - of F-components, 28
 - Overview, 25
- Consistency
 - Of the safety program, 170
- constants
 - Boolean, 62
- Conversion operations
 - Convert BOOL to WORD, 332, 488
 - Convert value, 331, 488
 - Convert WORD to BOOL, 334, 490
 - Scale values, 336, 491
- Convert
 - Data, 332, 334, 488, 490
 - Value, 331, 488
- CONVERT, 331, 488
- Convert data, 332, 334, 488, 490
- Conveyor equipment, stopped, 254, 410
- Correctness
 - Hardware configuration, 205
 - Safety-related CPU-CPU communication, 208
- Count
 - Down, 307, 464
 - Up, 306, 462
 - Up and down, 309, 466
- Count down, 307, 464
- Count up, 306, 462
- Count up and down, 309, 466
- CPU-CPU communication, 25
 - Options for safety-related, 25
 - Overview of safety-related, 111
- Create twos complement, 325, 482
- CTD, 307, 464
- CTU, 306, 462
- CTUD, 309, 466

- Cycle time
 - F-runtime group, 67
 - Maximum, 524
 - Monitoring time for, 525

D

- Data block, 105
- Data transfer
 - From safety program to standard user program, 105
 - From standard user program to safety program, 108
- Data types
 - For safety program, 59
- DB access, fully qualified, 63, 88
- Deactivating
 - F-capability, 29
 - Safety mode, 190
- Deleting
 - F-blocks, 199
 - Safety program, 199
- DIAG
 - ESTOP1, 240, 395
 - EV1oo2DI, 276, 432
 - FDBACK, 283, 439
 - F-I/O DB, 82
 - MUT_P, 265, 421
 - MUTING, 254, 410
 - RCVS7, 363, 517
 - SENDDP/RCVDP, 356, 510
 - SENDS7, 363, 517
 - SFDOOR, 289, 445
 - TWO_H_EN, 249, 405
- Diagnostic parameters, 216
- Diagnostic tag, 216
- Diagnostics
 - Fail-safe system, 216
- Discrepancy error, 254, 410
- DIV, 323, 480
- Divide, 323, 480
- Downloading
 - Hardware configuration, 171
 - Safety program, 171
 - Standard user program, 171
- Downloading Standard user program, 171
- DP/DP coupler, 122

E

- Empty box
 - Inserting a LAD element, 221
 - Inserting an FBD element, 370

EN, 59
 ENO, 59
 ESTOP1, 240, 395
 EV1oo2DI, 276, 432
 EXCLUSIVE OR, 346, 377, 501

F

F_IO_StructureDescCRC, 38
 Fail-safe acknowledgment, 351, 506
 Fail-safe DP standard slaves
 Configuring, 38
 Fail-safe standard I/O devices
 Configuring, 38
 Fail-safe system, 17
 Fail-safe value, 70, 80
 Fault reaction function, 7, 17
 FBD element
 Inserting, 370
 F-block
 Copying, 74
 Deleting, 199
 F-channel faults, fail-safe value output, 80
 F-collective signature, 190
 F-components, 25
 F-CPU, 25, 54
 Configuring, 29
 Going to STOP mode, 213
 Setting up access permission, 54
 F-cycle time, monitoring time, 525
 F-DB, 58
 Creating, 73
 for F-runtime group communication, 69
 F-shared DB, 107
 FDBACK, 283, 439
 F-destination address, 34
 F-FB, 58, 73
 F-FC, 58, 73
 F-I/O, 25
 Configuring, 33
 Reintegration, 81, 90, 92, 94
 Removing and inserting during operation, 214
 F-I/O access, 79
 During operation, 197
 Restrictions in RUN mode, 198
 Via the process image, 79, 153
 F-I/O DB, 58, 82
 Name, 33, 88
 Number, 33, 88
 Structure of DIAG, 82
 F-I/O faults or channel faults, 94
 F-I/O faults, fail-safe value output, 80

Firmware update, 214
 Flash Card, 178
 Flip-flop
 Reset/set, 232, 385
 Set/reset, 230, 383
 F-monitoring time, 31, 36, 524
 F-parameters, 28
 F-runtime group, 56, 58
 Changing, 72
 Default setting, 66
 Defining, 67
 Deleting, 72
 Maximum cycle time, 67, 528
 Rules, 65
 Safety-related communication, 69
 F-runtime group communication, 67, 69
 Monitoring time, 528
 Restrictions in RUN mode, 197
 F-shared DB, 105, 107, 190
 F-system
 Monitoring time, 523
 Response time, 523
 Fully qualified DB access, 63, 88
 Function test of the safety program, 178, 189, 202

G

Global data block
 Open, 342
 Global library, 75
 Group diagnostics for fail-safe signal modules, 37
 Group passivation, 97
 GSD files
 Configuration, 38

H

Hardware components, 18
 Hardware configuration, 28
 Checking for correctness, 205
 Compiling, 169
 Downloading, 171
 Help
 Open, 21

I

IE/PB Link, 154
 Implementation of user acknowledgment, 102
 Insert binary input, 372
 Installation

- STEP 7 Safety, 21
- Instance DB, 64, 73
- Instructions
 - for the safety program, 58
 - Get negated status bit OV, 355
 - Get status bit OV, 354, 509
 - Testing for acceptance, 204
- IPAR_EN, 82
- IPAR_OK, 82

J

- JMP, 338, 493
- JMPN, 339, 494
- Jump
 - If RLO = 0, 339, 494
 - If RLO = 1, 338, 493
- Jump label, 340, 495

L

- LABEL, 340, 495
- LAD element
 - Inserting, 221
- Library
 - Global libraries, 75
 - Project library, 76
- Life cycle of fail-safe automation systems, 529
- Light curtain, 254, 410
- Local data, 63

M

- Main safety block, 58, 73
- Math functions
 - Add, 317, 474
 - Create twos complement, 325, 482
 - Divide, 323, 480
 - Multiply, 321, 478
 - Subtract, 319, 476
- Maximum cycle time, 524, 528
- Memory Card, 178
- Memory reset, 178, 193
- Migrating projects
 - from S7 Distributed Safety, 22
- Migration
 - from S7 Distributed Safety, 22, 158
 - Printout, 188
- Modifying, 189
 - Safety program, 196
- Monitoring, 189

- Safety program, 196
 - Two-hand monitoring, 246, 249, 401, 405
- Monitoring time, 523, 524
 - Communication between F-CPU and F-I/O, 526
 - Communication between F-CPUs, 527
 - Communication between I-salve and slave, 526
 - F-cycle time, 525
- MOVE, 327, 484
- Move operations
 - Move value, 327, 484
 - Read value indirectly from an F-DB, 330, 487
 - Write value indirectly to an F-DB, 328, 485
- MUL, 321, 478
- Multiply, 321, 478
- MUT_P, 265, 421
- MUTING, 254, 410
 - Structure of DIAG, 254, 410
- Muting operation
 - With 4 muting sensors, 254, 410
 - With reflection light barriers, 254, 410

N

- N, 236, 389
- N_TRIG, 239, 393
- NEG, 325, 482
- Network
 - Inserting, 220, 369
- Normally closed contact, 225
- Normally open contact, 224
- NOT, 226

O

- Off delay, 303, 459
- Offline password, 51
- Offline-online comparison of safety programs, 183
- On delay, 300, 456
- Online password, 51
- Open global data block, 497
- Operand
 - Scan for negative signal edge, 236, 389
 - Scan for positive signal edge, 234, 387
- Operand area
 - For safety program, 60
- Operating principle
 - RCVDP, 356, 510
 - RCVS7, 363, 517
 - SENDDP, 356, 510
 - SENDS7, 363, 517
- Operating system update, 214

Operational safety of the system, 7
 OPN, 342, 497
 OR, 345, 376, 500
 Output
 Reset, 228, 380
 Set, 229, 381
 OV, 354, 355, 509

P

P, 234, 387
 P_TRIG, 237, 391
 Parameter types, 59
 Parameters, 265, 421
 Safety-related, 28
 PASS_ON, 82
 PASS_OUT/QBAD, 82
 Passivation
 Channel-level, 33
 F-I/O, 89
 Output of fail-safe values, 80
 Passivation of F-I/O, 89
 After communication errors, 92
 After F-I/O faults or channel faults, 94
 After startup, 90
 Group passivation, 97
 Password, 49, 191
 F-CPU, 54
 Offline, 51
 Online, 51
 Safety program, 51
 Performance Level, 17
 PLCSIM, 176, 189, 193
 PN/PN coupler, 114
 Printing
 Project data, 187
 Process image, 79, 105
 Process safety time, 528
 Productive operation, 49
 PROFIBUS DP, 18
 PROFINET IO, 18
 PROFIsafe, 38
 PROFIsafe destination address, 30
 Program control operations
 Jump if RLO = 0, 339, 494
 Jump if RLO = 1, 338, 493
 Jump label, 340, 495
 Open global data block, 342, 497
 Return, 341, 496
 Program identification, 178
 Programming
 Group passivation, 97

Overview, 55
 Startup protection, 76
 Validity checks, 108
 Programming an F-communication DB, 158
 Project data
 Printing, 187
 Project library, 76
 Proof test, 214
 Protection level of the F-CPU, 32
 Protection through program identification, 178
 Pulse
 Generate, 297

Q

QBAD, 88, 89

R

R, 228, 380
 RCVDP, 117, 118, 125, 126, 133, 134, 139, 140, 190, 356, 510
 Behavior in the event of communication errors, 356, 510
 Receiving data, 356, 510
 Structure of DIAG, 356, 510
 Timing diagrams, 356, 510
 RCVS7, 157, 158, 190, 363, 517
 RD_FDB, 330, 487
 Readme file, 21
 Reflection light barriers, 254, 410
 Reintegration of F-I/O, 81, 83, 89
 After communication errors, 92
 After F-I/O faults or channel faults, 94
 After startup of F-system, 90
 Programming a user acknowledgment, 99, 102
 with group passivation, 97
 Replacing
 Software components, 214
 Response time of F-system, 523, 528
 Restart inhibit
 MUT_P, 265, 421
 MUTING, 254, 410
 On interruption of the light curtain, 254, 265, 410, 421
 Restart protection, 76
 RET, 341, 496
 Return, 341, 496
 RLO
 Invert, 226, 373
 Scan for negative signal edge, 239, 393

Scan for positive signal edge, 237, 391
RS, 232, 385
Rules for testing, 193
RUN, 197
RUN mode, 197

S

S, 229, 381
S7 connection
 Safety-related communication, 155
S7-PLCSIM, 176, 189, 193
Safety Administration Editor, 39
Safety function, 17
 Calculation of response time, 528
 Example, 17
Safety functions
 ACK_GL, 295, 451
 ESTOP1, 240, 395
 EV1oo2DI, 276, 432
 FDBACK, 283, 439
 MUT_P, 265, 421
 MUTING, 254, 410
 SFDOOR, 289, 445
 TWO_H_EN, 249, 405
 TWO_HAND, 246, 401
Safety Integrity Level, 17
Safety mode
 Activating, 191
 Deactivating, 190
Safety printout, 187, 201
Safety program, 18
 Allow automatic generation, 31
 Comparing, 183
 Compiling, 169
 Deleting, 72, 199
 Downloading, 171
 Function test, 189
 Instructions, 58
 Modifying, 189, 196
 Monitoring, 189, 196
 Output of fail-safe values, 80
 Password, 51
 Structuring, 56
 Testing, 193
 Work memory requirement, 177
Safety requirements, 7, 17
Safety-related communication between F-runtime groups, 69
Safety-related communication via S7 connections
 Configuring, 155
 Data transfer limits, 161

Safety-related CPU-CPU communication, 25, 111, 363, 517
 Checking for correctness, 208
 F-communication DB, 158
 Options, 25
 RCVDP, 356, 510
 Restrictions in RUN mode, 197
 SENDDP, 356, 510
Safety-related IO Controller-I-Device communication
 Configuring, 130
 Data transfer limits, 135
 Programming, 134
Safety-related IO controller-IO controller communication
 Data transfer limits, 121
Safety-related IO Controller-IO Controller communication
 Configuring, 114
 Programming, 118
Safety-related IO Controller-I-slave communication, 154
Safety-related I-slave-I-slave communication
 Configuring, 143
 Data transfer limits, 142
 Programming, 140
Safety-related I-slave-slave communication
 Configuring, 148
 Data transfer limits, 153
Safety-related master-I-slave communication
 Configuring, 136
 Data transfer limits, 142
 Programming, 140
Safety-related master-master communication
 Configuring, 122
 Data transfer limits, 129
 Programming, 126
Safety-related parameters, 28
Scale
 Values, 336, 491
SCALE, 336, 491
SENDDP, 117, 118, 125, 126, 133, 134, 139, 140, 190, 356, 510
 Behavior in the event of communication errors, 356, 510
 Sending data, 356, 510
 Structure of DIAG, 356, 510
 Timing diagrams, 356, 510
Sending and receiving data via S7 connections, 363, 517
SENDS7, 157, 158, 190, 363, 517
Service & Support, 7
SFDOOR, 289, 445

- Shift and rotate
 - Shift left, 349, 504
 - Shift right, 347, 502
 - Shift left, 349, 504
 - Shift right, 347, 502
 - SHL, 349, 504
 - SHR, 347, 502
 - SIL, 17
 - SIMATIC Safety, 3, 17
 - Configuring and programming software, 18
 - Hardware and software components, 18
 - Principles of safety functions, 17
 - Product overview, 17
 - Safety program, 18
 - STEP 7 Safety optional package, 18
 - Simulation, 189
 - Simulation devices in the F-system, 213
 - Software components, 18, 214
 - Software requirements, 21
 - SR, 230, 383
 - Startup, 76, 90
 - Startup characteristics
 - MUT_P, 265, 421
 - RCVDP, 356, 510
 - RCVS7, 363, 517
 - SENDDP, 356, 510
 - SENDS7, 363, 517
 - Startup protection, 76
 - Status bit OV
 - Get, 354, 509
 - Get negated, 355
 - STEP 7 Safety, 18
 - Additional support, 3
 - Basic knowledge, required, 3
 - Documentation, 3
 - Information landscape, 4
 - Order number, 3
 - Service & Support, 3, 7
 - Writing conventions, 6
 - STOP, 213
 - STP, 213
 - Structure of the safety program, 56
 - SUB, 319, 476
 - Subtract, 319, 476
- T**
- Tag
 - F-I/O DB, 82
 - Monitoring/modifying, 193
 - Tag table, 193
 - Testing of safety program, 193
- TIMEOUT, 524, 527
 - Timer operations
 - Generate off-delay, 303, 459
 - Generate on-delay, 300, 456
 - Generate pulse, 297, 453
 - Timing diagrams, 254, 265, 356, 410, 421, 510
 - RCVDP, 356, 510
 - SENDDP, 356, 510
 - TOF, 303, 459
 - TON, 300, 456
 - TP, 297, 453
 - Training center, 3
 - Transferring
 - Safety program to the F-CPU, 178
 - Truth table
 - AND, 375
 - EXCLUSIVE OR, 378
 - OR, 376
 - TÜV certificate, 204
 - TWO_H_EN, 249, 405
 - TWO_HAND, 246, 401
- U**
- Undergoing a system acceptance test, 201
 - Uninstallation
 - STEP 7 Safety, 21
 - User acknowledgement, 99, 102, 254, 410
 - Example, 101
 - User safety function, 3, 17
- V**
- V2-MODE, 38
 - Validity check, 108
 - Data transfer from standard program to safety program, 209
 - Value
 - Convert, 331, 488
 - Move, 327, 484
 - Read indirectly from an F-DB, 330, 487
 - Scale, 336, 491
 - Write indirectly to an F-DB, 328, 485
- W**
- W_BO, 334, 490
 - Wiring test, 193
 - Word logic operations
 - AND, 344, 499
 - EXCLUSIVE OR, 346, 501

OR, 345, 500
Work memory requirement of safety program, 177
WR_FDB, 328, 485

X

X, 377
XOR, 346, 501