EcoStruxure™ Control Expert

Safety Block

Library Guide

Original instructions

QGH60275.11
11/2024

www.se.com

Legal Information

Table of Contents

Safety Information 9
About the Book 11
General information....17
Block Types and Their Applications 18
Block Types 19
FFB Structure....20
EN and ENO 22
Communication....24
S_RD_ETH_MX: Safe Mx80 Controller-to-Controller Ethernet Communication Function 25
Description....26
S_WR_ETH_MX: Safe Mx80 Controller-to-Controller Ethernet Communication Function 32
Description....33
S_RD_ETH_MX2: Safe Mx80 Controller-to-Controller Ethernet Communication Function 37
Description....38
S_WR_ETH_MX2: Safe Mx80 Controller-to-Controller Ethernet Communication Function 44
Description....45
Comparison 49
S_EQ_***: Equal To....50
Description....50
S_GE_***: Greater Than or Equal To 53
Description....53
S_GT_***: Greater Than 56
Description....56
S_LE_***: Less Than or Equal To 59
Description....59
S_LT_***: Less Than 62
Description....62
S_NE_***: Not Equal To....65

Description....65
Actuator....68
S_EDM: Actuator Error Detection Monitoring....69
Description....69
S_ENABLE_SWITCH: Three Position Enable Switch....79
Description....79
S_ESPE: Electro-Sensitive Protective Equipment....87
Description....87
S_GUARD_LOCKING: Guard Lock Control....95
Description....95
S_GUARD_MONITORING: Guard Lock Monitoring....103
Description....103
S_MODE_SELECTOR: Safety Mode Switch....113
Description....113
S_OUTCONTROL: Output Driver....121
Description....121
Sensor....129
S_AI_COMP: Analog Input Compare....130
Description....130
S_ANTIVALENT: Compare Antivalent Inputs....137
Description....137
S_EMERGENCYSTOP: Emergency Stop Monitor....144
Description....144
S_EQUIVALENT: Compare Equivalent Inputs....152
Description....152
S_MUTING_PAR: Parallel Muting....158
Description....158
S_MUTING_SEQ: Sequential Muting....173
Description....173
S_TWO_HAND_CONTROL_TYPE_II: Two Hand Control....184
Description....184
S_TWO_HAND_CONTROL_TYPE_III: Two Hand Control with Timer....191
Description....191
High Availability....199
S_AIHA: High Availability for Mx80 Safety Analog Inputs....200

Description ... 200
S_DIHA: High Availability for Mx80 Safety Digital Inputs ... 204
Description ... 204
Logic ... 209
S_AND_***: AND Function ... 210
Description ... 210
S_F_TRIG: Falling Edge Detection ... 212
Description ... 212
S_NOT_***: Negation ... 214
Description ... 214
S_OR_***: OR Function ... 216
Description ... 216
S_R_TRIG: Rising Edge Detection ... 218
Description ... 218
S_ROL_***: Rotate Left ... 220
Description ... 220
S_ROR_***: Rotate Right ... 223
Description ... 223
S_RS: Bistable Function Block, Reset Dominant ... 226
Description ... 226
S_SHL_***: Shift Left ... 228
Description ... 228
S_SHR_***: Shift Right ... 231
Description ... 231
S_SR: Bistable Function Block, Set Dominant ... 234
Description ... 234
S_XOR_***: Exclusive OR Function ... 236
Description ... 236
Mathematics ... 238
S_ABS_***: Absolute Value Computation ... 239
Description ... 239
S_ADD_***: Addition ... 242
Description ... 242
S_DIV_***: Division ... 245
Description ... 245

S_MUL_***: Multiplication 248
Description 248
S_MOVE: Assignment 251
Description 251
S_NEG_***: Negation 253
Description 253
S_SQRT_REAL: Safety Square Root 256
Description 256
S_SIGN_***: Sign Evaluation 258
Description 258
S_SUB_***: Subtraction 261
Description 261
Statistical 264
S_LIMIT_***: Limit 265
Description 265
S_MAX_***: Maximum Value Function 268
Description 268
S_MIN_***: Minimum Value Function 271
Description 271
S_MUX_***: Multiplexer 274
Description 274
S_SEL: Binary Selection 277
Description 277
System 279
S_SYST_CLOCK_MX: System Clock 280
Description 280
S_SYST_READ_TASK_BIT_MX: System Task Bit 283
Description 283
S_SYST_RESET_TASK_BIT_MX: Reset Task 285
Description 285
S_SYST_STAT_MX: System State 287
Description 287
S_SYST_TIME_MX: System Time 289
Description 289
Timers & Counters 291

S_CTD_***: Down Counter 292
Description 292
S_CTU_***: Up Counter 295
Description 295
S_CTUD_***: Up/Down Counter 298
Description 298
S_TOF: Off Delay 301
Description 301
S_TON: On Delay 304
Description 304
S_TP: Pulse 307
Description 307
Type to Type 310
S_BIT_TO_BYTE: Type Conversion 311
Description 311
S_BIT_TO_WORD: Type Conversion 314
Description 314
S_BOOL_TO_***: Type Conversion 317
Description 317
S_BYTE_TO_BIT: Type Conversion 319
Description 319
S_BYTE_TO_***: Type Conversion 322
Description 322
S_DWORD_TO_***: Type Conversion 324
Description 324
S_INT_TO_***: Type Conversion 327
Description 327
S_DINT_TO_***: Type Conversion 330
Description 330
S_REAL_TO_***: Type Conversion 333
Description 333
S_TIME_TO_UDINT: Type Conversion 336
Description 336
S_UDINT_TO_***: Type Conversion 338
Description 338

S_UINT_TO_***: Type Conversion 341
Description 341
S_WORD_TO_BIT: Type Conversion 344
Description 344
S_WORD_TO_***: Type Conversion 347
Description 347
Appendices 350
System Objects 351
M580 Safety System Bits 352
M580 Safety System Words 354
SRAC References 357
Glossary 359
Index 365

Safety Information

Important Information

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, service, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

DANGER

DANGER indicates a hazardous situation which, if not avoided, **will result in** death or serious injury.

WARNING

WARNING indicates a hazardous situation which, if not avoided, **could result in** death or serious injury.

CAUTION

CAUTION indicates a hazardous situation which, if not avoided, **could result** in minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

Please Note

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

About the Book

Document Scope

This document describes the functions and function blocks of the Safety library.

Validity Note

This document is valid for EcoStruxure™ Control Expert 16.1 or later.

Related Documents

Title of documentation	Reference number
M580 Safety, Safety Related Application Conditions — Verification Plan	EIO0000004540 (ENG) EIO0000004741 (FRE) EIO0000004742 (GER) EIO0000004744 (ITA) EIO0000004743 (SPA) EIO0000004745 (CHS)
Modicon M580, Safety Manual	QGH46982 (English), QGH46983 (French), QGH46984 (German), QGH46985 (Italian), QGH46986 (Spanish), QGH46987 (Chinese)
Modicon M580, Safety System Planning Guide	QGH60283 (English), QGH60284 (French), QGH60285 (German), QGH60286 (Spanish), QGH60287 (Italian), QGH60288 (Chinese)
Modicon Controllers Platform Cyber Security, Reference Manual	EIO0000001999 (English), EIO0000002001 (French), EIO0000002000 (German), EIO0000002002 (Italian), EIO0000002003 (Spanish), EIO0000002004 (Chinese)
Modicon M580, Hardware, Reference Manual	EIO0000001578 (English), EIO0000001579 (French), EIO0000001580 (German), EIO0000001582 (Italian), EIO0000001581 (Spanish), EIO0000001583 (Chinese)
Modicon M580 Standalone System Planning Guide for Frequently Used Architectures	HRB62666 (English), HRB65318 (French), HRB65319 (German), HRB65320 (Italian), HRB65321 (Spanish), HRB65322 (Chinese)
Modicon M580 System Planning Guide for Complex Topologies	NHA58892 (English), NHA58893 (French), NHA58894 (German), NHA58895 (Italian), NHA58896 (Spanish), NHA58897 (Chinese)

Title of documentation	Reference number
EcoStruxure™ Control Expert, Operating Modes	33003101 (English), 33003102 (French), 33003103 (German), 33003104 (Spanish), 33003696 (Italian), 33003697 (Chinese)
EcoStruxure™ Control Expert, System Bits and Words, Reference Manual	EIO0000002135 (English), EIO0000002136 (French), EIO0000002137 (German), EIO0000002138 (Italian), EIO0000002139 (Spanish), EIO0000002140 (Chinese)

To find documents online, visit the Schneider Electric download center (www.se.com/ww/en/download/).

Product Related Information

DANGER

HAZARD OF ELECTRIC SHOCK, EXPLOSION, OR ARC FLASH

- Disconnect all power from all equipment including connected devices prior to removing any covers or doors, or installing or removing any accessories, hardware, cables, or wires except under the specific conditions specified in the appropriate hardware guide for this equipment.
- Always use a properly rated voltage sensing device to confirm the power is off where and when indicated.
- Replace and secure all covers, accessories, hardware, cables, and wires and confirm that a proper ground connection exists before applying power to the equipment.
- Use only the specified voltage when operating this equipment and any associated equipment.

Failure to follow these instructions will result in death or serious injury.

⚠WARNING

LOSS OF CONTROL

- Perform a Failure Mode and Effects Analysis (FMEA), or equivalent risk analysis, of your application, and apply preventive and detective controls before implementation.
- Provide a fallback state for undesired control events or sequences.
- Provide separate or redundant control paths wherever required.
- Supply appropriate parameters, particularly for limits.
- Review the implications of transmission delays and take actions to mitigate them.
- Review the implications of communication link interruptions and take actions to mitigate them.
- Provide independent paths for control functions (for example, emergency stop, over-limit conditions, and error conditions) according to your risk assessment, and applicable codes and regulations.
- Apply local accident prevention and safety regulations and guidelines.[1]
- Test each implementation of a system for proper operation before placing it into service.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

[1] For additional information, refer to NEMA ICS 1.1 (latest edition), *Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control* and to NEMA ICS 7.1 (latest edition), *Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems* or their equivalent governing your particular location.

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

- Only use software approved by Schneider Electric for use with this equipment.
- Update your application program every time you change the physical hardware configuration.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Terminology Derived from Standards

The technical terms, terminology, symbols and the corresponding descriptions in the information contained herein, or that appear in or on the products themselves, are generally derived from the terms or definitions of international standards.

In the area of functional safety systems, drives and general automation, this may include, but is not limited to, terms such as *safety*, *safety function*, *safe state*, *fault*, *fault reset*, *malfunction*, *failure*, *error*, *error message*, *dangerous*, etc.

Among others, these standards include:

Standard	Description
IEC 61131-2:2007	Programmable controllers, part 2: Equipment requirements and tests.
ISO 13849-1:2023	Safety of machinery: Safety related parts of control systems. General principles for design.
EN 61496-1:2020	Safety of machinery: Electro-sensitive protective equipment. Part 1: General requirements and tests.
ISO 12100:2010	Safety of machinery - General principles for design - Risk assessment and risk reduction
EN 60204-1:2006	Safety of machinery - Electrical equipment of machines - Part 1: General requirements
ISO 14119:2013	Safety of machinery - Interlocking devices associated with guards - Principles for design and selection
ISO 13850:2015	Safety of machinery - Emergency stop - Principles for design
IEC 62061:2021	Safety of machinery - Functional safety of safety-related electrical, electronic, and electronic programmable control systems
IEC 61508-1:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: General requirements.
IEC 61508-2:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Requirements for electrical/electronic/programmable electronic safety-related systems.
IEC 61508-3:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Software requirements.
IEC 61784-3:2021	Industrial communication networks - Profiles - Part 3: Functional safety fieldbuses - General rules and profile definitions.
2006/42/EC	Machinery Directive
2014/30/EU	Electromagnetic Compatibility Directive
2014/35/EU	Low Voltage Directive

In addition, terms used in the present document may tangentially be used as they are derived from other standards such as:

Standard	Description
IEC 60034 series	Rotating electrical machines
IEC 61800 series	Adjustable speed electrical power drive systems
IEC 61158 series	Digital data communications for measurement and control – Fieldbus for use in industrial control systems

Finally, the term *zone of operation* may be used in conjunction with the description of specific hazards, and is defined as it is for a *hazard zone* or *danger zone* in the *Machinery Directive* (*2006/42/EC*) and *ISO 12100:2010*.

NOTE: The aforementioned standards may or may not apply to the specific products cited in the present documentation. For more information concerning the individual standards applicable to the products described herein, see the characteristics tables for those product references.

Trademarks

QR Code is a registered trademark of DENSO WAVE INCORPORATED in Japan and other countries.

Information on Non-Inclusive or Insensitive Terminology

As a responsible, inclusive company, Schneider Electric is constantly updating its communications and products that contain non-inclusive or insensitive terminology. However, despite these efforts, our content may still contain terms that are deemed inappropriate by some customers.

General information

What’s in This Part

Block Types and Their Applications 18

Overview

This section contains general information about the Safety library.

Block Types and Their Applications

What's in This Chapter

Block Types 19
FFB Structure 20
EN and ENO 22

Overview

This chapter describes the different block types and their applications.

Block Types

Block Types

Different block types are used in Control Expert. The general term for all block types is FFB.

There are the following types of block:

- Elementary Function (EF)
- Elementary Function Block (EFB)

Elementary Function

Elementary functions (EF) have no internal status. If the input values are the same, the value at the output is the same for all executions of the function, e.g. the addition of two values gives the same result at every execution.

An elementary function is represented in the graphical languages (FBD and LD) as a block frame with inputs and an output. The inputs are always represented on the left and the outputs always on the right of the frame. The name of the function, i.e. the function type, is shown in the center of the frame.

The number of inputs can be increased with some elementary functions.

Elementary Function Block

Elementary function blocks (EFB) have an internal status. If the inputs have the same values, the value on the output can have another value during the individual executions. For example, with a counter, the value on the output is incremented.

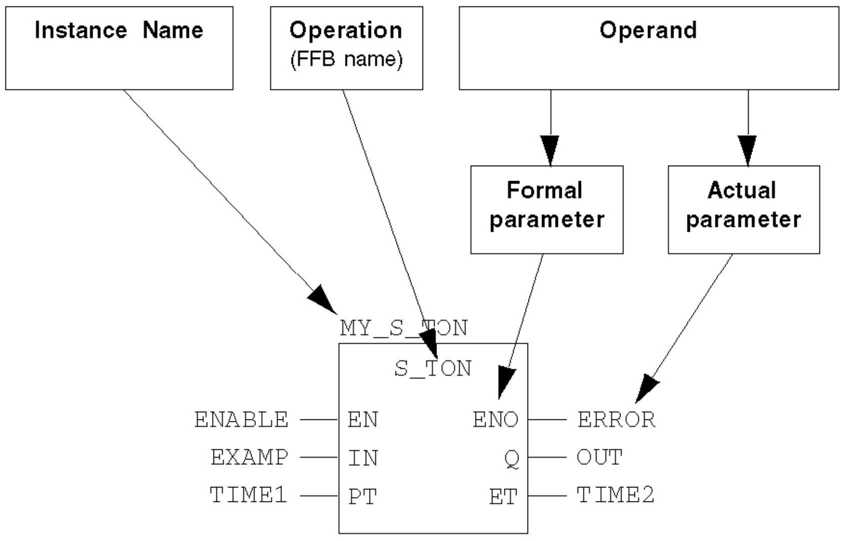
An elementary function block is represented in the graphical languages (FBD and LD) as a block frame with inputs and outputs. The inputs are always represented on the left and the outputs always on the right of the frame. The name of the function block, i.e. the function block type, is shown in the center of the frame. The instance name is displayed above the frame.

FFB Structure

Structure

Each FFB is made up of an operation (name of the FFB), the operands required for the operation (formal and actual parameters) and an instance name for elementary/derived function blocks.

Call of a function block in the FBD programming language:



Operation

The operation determines which function is to be executed with the FFB, e.g. shift register, conversion operations.

Operand

The operand specifies what the operation is to be executed with. With FFBs, this consists of formal and actual parameters.

Formal/Actual Parameters

Inputs and outputs are required to transfer values to or from an FFB. These are called formal parameters.

Objects are linked to formal parameters; these objects contain the process states. They are called actual parameters.

At program runtime, the values from the process are transferred to the FFB via the actual parameters and then output again after processing.

As a general rule, assign the same data type to the actual parameters that is used for the input/output (formal parameters). The only exceptions to this rule are generic inputs/outputs whose data type is determined by the actual parameter. If all actual parameters consist of literals, a suitable data type is selected for the function block.

EN and ENO

Description

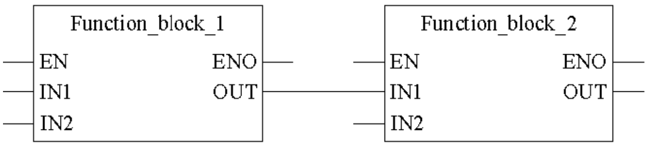
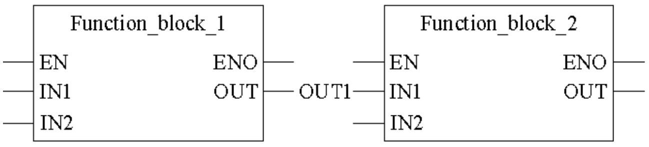
An EN input and an ENO output can be configured for all FFBs.

If the value of EN is ...	then ...
0 when the FFB is called up,	the algorithms defined by the FFB are not executed and ENO is set to 0.
1 when the FFB is called up,	the algorithms defined by the FFB are executed. After the algorithms have been executed successfully, the value of ENO is set to 1. **Note:** If an error is detected when executing these algorithms, ENO is set to 0.

Examples

The following tables describe examples if ENO is set to 0 (caused by EN=0 or a detected error during execution).

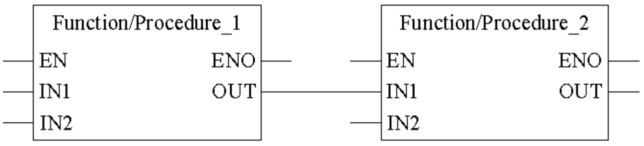
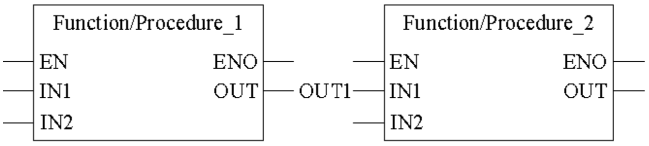
Function blocks

Example	Description
EN/ENO handling with function blocks that have 1 link as an output parameter:  Function_block_1 EN ENO IN1 OUT IN2 Function_block_2 EN ENO IN1 OUT IN2	If EN from FunctionBlock_1 is set to 0, the output connection OUT from FunctionBlock_1 retains the status it had in the last correctly executed cycle.
EN/ENO handling with function blocks that have 1 variable and 1 link as output parameters:  Function_block_1 EN ENO IN1 OUT IN2 OUT1 Function_block_2 EN ENO IN1 OUT IN2	If EN from FunctionBlock_1 is set to 0 • the output connection OUT from FunctionBlock_1 retains the status it had in the last correctly executed cycle; • the variable OUT1 on the same pin, either retains its previous status or can be changed externally without influencing the connection; • the variable and the link are saved independently of each other.

Functions/Procedures

As defined in IEC61131-3, the outputs from deactivated functions (EN-input set to 0) is undefined. The same applies to procedures.

Here is an explanation of the output statuses in this case:

Example	Description
EN/ENO handling with function/procedure blocks that (only) have 1 link as an output parameter:  Function/Procedure_1 EN ENO IN1 OUT IN2 Function/Procedure_2 EN ENO IN1 OUT IN2	If EN from Function/Procedure_1 is set to 0, the output connection OUT from Function/Procedure_1 is also set to 0.
EN/ENO handling with function/procedure blocks that have 1 variable and 1 link as output parameters:  Function/Procedure_1 EN ENO IN1 OUT IN2 OUT1 Function/Procedure_2 EN ENO IN1 OUT IN2	If EN from Function/Procedure_1 is set to 0 • the output connection OUT from Function/Procedure_1 is also set to 0; • the variable OUT1 on the same pin retains its previous value. **Note:** In this way it is possible for the variable and the link to have different values.

The output behavior of the FFBs does not depend on whether the FFBs are called up without EN/ENO or with EN=1.

Conditional/Unconditional FFB Call

Unconditional or conditional calls are possible with each FFB. The condition is realized by pre-linking the input EN.

- When EN is connected, then the FFB will be processed when EN = 1.
- When EN is shown or hidden and marked TRUE, or shown and not occupied, then the FFB is processed.

Communication

What's in This Part

S_RD_ETH_MX: Safe Mx80 Controller-to-Controller Ethernet Communication Function 25
S_WR_ETH_MX: Safe Mx80 Controller-to-Controller Ethernet Communication Function 32
S_RD_ETH_MX2: Safe Mx80 Controller-to-Controller Ethernet Communication Function 37
S_WR_ETH_MX2: Safe Mx80 Controller-to-Controller Ethernet Communication Function 44

Introduction

This section describes the derived function blocks of the `Communication` family.

The transfer of safety data between Safety controllers is firmware dependent:

- For controller with firmware 3.10 or earlier, use `S_RD_ETH_MX` and `S_WR_ETH_MX` function blocks together.
- For controller with firmware 3.20 or later, use `S_RD_ETH_MX2` and `S_WR_ETH_MX2` function blocks together.

S_RD_ETH_MX: Safe Mx80 Controller-to-Controller Ethernet Communication Function

What's in This Chapter

Description 26

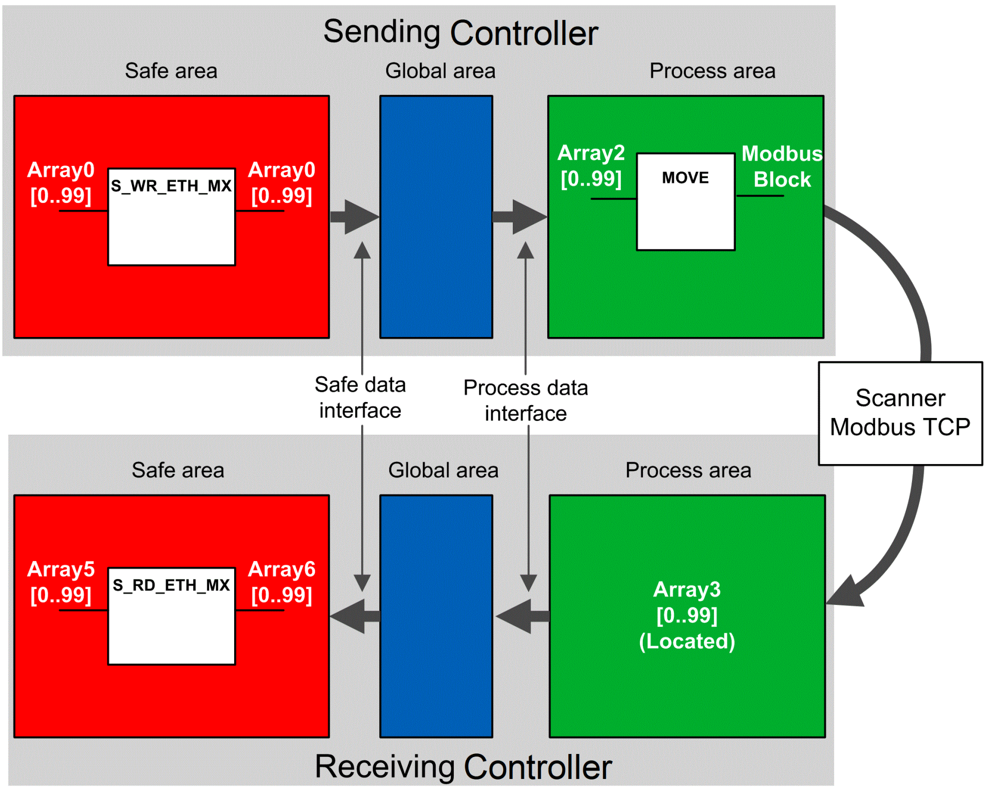
Introduction

This chapter describes the `S_RD_ETH_MX` block.

Description

Function Description

Use the `S_RD_ETH_MX` function block together with the `S_WR_ETH_MX` function block to complete a transfer of safety data between safety controllers with firmware 3.10 or earlier. Each safety controller is designed to isolate safety data from process data by passing data through a global area that is accessible to both the safety and process areas. In this way, process functions do not directly imcontrollert safety data.



The `S_RD_ETH_MX` function block is used by a receiver controller to copy the data received in the process area to the safety area and validate the accuracy of the received data.

`EN` and `ENO` can be configured as additional parameters.

The `S_RD_ETH_MX` function block:

- Copies the data received in the `INPUT_DATA` register to the `OUTPUT_DATA_SAFE` register if it passes the following tests:
  - The function block verifies the CRC of the last data packet received, via explicit messaging over Modbus TCP. If the CRC is not correct, the data is considered as corrupted and it is not written to the `OUTPUT_DATA_SAFE` register in the safety area.
  - The function block verifies the last data received to determine if it is more recent than the data already written in the `OUTPUT_DATA_SAFE` register in the safety area (by comparing time stamps). If the last data received is not more recent, it is not copied to the `OUTPUT_DATA_SAFE` register in the safety area.
- Verifies the age of the data in the safety area. If the age is higher than a configurable maximum value set in the `SAFETY_CONTROL_TIMEOUT` input register, the data is declared corrupted and the `HEALTH` bit is set to 0.

  NOTE: The data age is the time difference between the time when the data is computed in the sender controller and the time when the data is checked in the receiver controller. The time base reference is periodically updated with the time received from an NTP server.

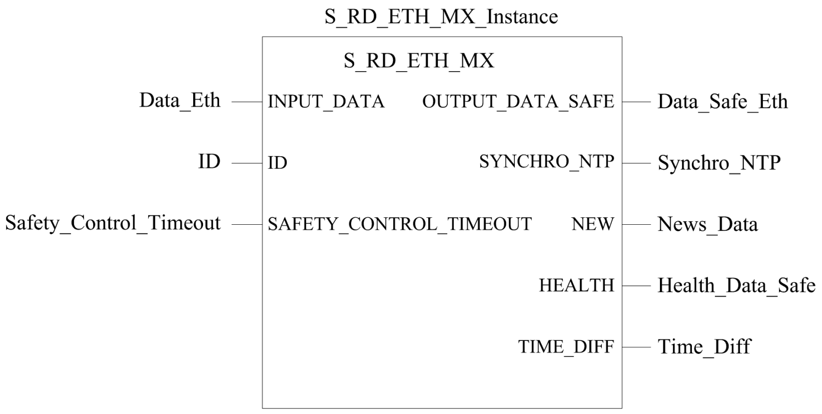
  If the `HEALTH` bit is set to 0, the data available in the `OUTPUT_DATA_SAFE` array is considered as corrupted. In this case, take appropriate measures.

Design your application so that the `S_RD_ETH_MX` function block is called at each cycle in the receiver controller SAFE task and executes before the data usage in the SAFE task cycle.

NOTE: For additional information, refer to chapter *Peer to Peer Communications* (see Modicon M580, Safety Manual).

Representation in FBD

Representation:



Input Parameters

Description of the input parameters:

Parameter	Data Type	Meaning
INPUT_DATA	ARRAY[0..99] of INT	Array of data variables received in the global memory area via explicit messaging over Modbus TCP. Composed of “User safety data” (from index 0 to 90) and “Reserved data” (from index 91 to 99). **NOTE:** Define these data variables as shared input variables, each with an equivalent global variable, using the **Safety Data Interface** tab in Control Expert.
ID	INT	Communication identifier. The unique ID value used to calculate the CRC. It is set to the same value as the value used by the sender. **NOTE:** Assign a unique value to the ID parameter that identifies the safe peer-to-peer communication between a sender and a receiver.
SAFETY_CONTROL_TIMEOUT	UINT	Time out value (in ms). Used to check the age of the data in the safety area and determine if that data is to be considered safe. Refer to the topic Calculating a SAFETY_CONTROL_ TIMEOUT Value, page 30.

Output Parameters

Description of the output parameters:

Parameter	Data Type	Meaning
OUTPUT_DATA_SAFE	ARRAY[0..99] of INT	Safety data variables array structure. Composed of “User safety data” (from index 0 to 90) and “Reserved data” (from index 91 to 99).
SYNCHRO_NTP	BOOL	• 1: Indicates that NTP time synchronization is healthy. • 0: Indicates NTP time synchronization is not healthy.
NEW	BOOL	• 1: Indicates that a new set of data has been refreshed in OUTPUT_DATA_SAFE during the present cycle. • 0: Indicates no new safe data has been refreshed.

Parameter	Data Type	Meaning
`HEALTH`	`BOOL`	• 1: Indicates the data in the **User Safety Data** area is safe. • 0: Indicates the data in the **User Safety Data** area is not safe.
`TIME_DIFF`	`INT`	Returns the age (in ms) of the data received and written in the `OUTPUT_DATA_SAFE` output parameter. Set to -1 if the internal `NTP` time is not initialized or if no correct data has yet been received.

`INPUT_DATA` and `OUTPUT_DATA_SAFE` Arrays Description

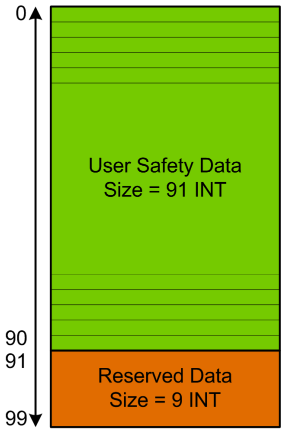
The `INPUT_DATA` arrays consist of data coming from the process data memory area. The `OUTPUT_DATA_SAFE` arrays consist of safety-related variables. Use the **Safety Data Interface** and the **Process Data Interface** tabs in Control Expert to make the link between the process variables and the safety-related variables.

`INPUT_DATA` and `OUTPUT_DATA_SAFE` arrays are composed of 2 zones:

- The **User Safety Data** zone contains user data. This zone starts at index 0 and finishes at index 90.
- The **Reserved Data** zone is reserved for auto-generated diagnostic data, including a CRC and time-stamp. This data is used by the receiving controller to determine if the data contained in the **User Safety Data** zone is safe or not. This zone starts at index 91 and finishes at index 99.

  NOTE: Do not write in the **Reserved Data** zone, as doing so will overwrite the auto-generated diagnostic data.

`INPUT_DATA` and `OUTPUT_DATA_SAFE` arrays (array[0..99] of INT) structure representation:



Calculating a SAFETY_CONTROL_TIMEOUT Value

When calculating a `SAFETY_CONTROL_TIMEOUT` value, consider the following:

- Minimum value: `SAFETY_CONTROL_TIMEOUT` > T1
- Typical value: `SAFETY_CONTROL_TIMEOUT` > 2 * T1

T1 = $\text{controller}_{\text{sender}}$ MAST cycle time + $\text{controller}_{\text{sender}}$ SAFE cycle time + Repetitive_rate + Network transmission time + $\text{controller}_{\text{receiver}}$ MAST cycle time + $\text{controller}_{\text{receiver}}$ SAFE cycle time

Where:

- *$\text{controller}_{\text{sender}}$ MAST cycle time* is the MAST cycle time of the sender controller.
- *$\text{controller}_{\text{sender}}$ SAFE cycle time* is the SAFE cycle time of the sender controller.
- *Repetitive_rate* is the time rate for the I/O scanner write query from the sender controller to the receiver controller.
- *Network transmission time* is the time consumed on the Ethernet network for the data transmission from the sender controller to the receiver controller.
- *$\text{controller}_{\text{receiver}}$ MAST cycle time* is the MAST cycle time of the receiver controller.
- *$\text{controller}_{\text{receiver}}$ SAFE cycle time* is the SAFE cycle time of the receiver controller.

Note that the value defined for the `SAFETY_CONTROL_TIMEOUT` parameter has a direct effect on the robustness and availability of the safe peer-to-peer communication. If the `SAFETY_CONTROL_TIMEOUT` parameter value greatly exceeds T1, the communication will be tolerant to various delays (for example network delays) or corrupted data transmissions.

Configure your Ethernet network so the load that does not cause an excessive delay on the network during data transmission, which could lead to the expiration of the timeout. To help safeguard your safe peer-to-peer communications from any excessive delays due to other non-safety data transmitted on the same network, consider using a dedicated Ethernet network for the safe peer-to-peer protocol.

When commissioning your project, you have to estimate the safe peer-to-peer communication performance by checking the values provided in the output parameter `TIME_DIFF` and evaluating the margin using the value defined in the `SAFETY_CONTROL_ TIMEOUT` parameter.

Understanding the `HEALTH` Bit

When the `HEALTH` bit value equals:

- 1: The integrity of the data is correct (CRC) and the age of the data is less than the value set in the `SAFTETY_CONTROL_TIMEOUT` input register.

  NOTE: The age of the data considered is the time between:

  - The beginning of the cycle where the data are computed in the sender controller.
  - The beginning of the cycle where the data are checked in the receiver controller.
- 0: New valid data are not received in the required time interval (the timer expires and the `HEALTH` bit is set to 0).

  NOTE: If the `HEALTH` bit is set to 0, the data in the output array `OUTPUT_DATA_ SAFE` is considered to be corrupted; take appropriate measures.

S_WR_ETH_MX: Safe Mx80 Controller-to-Controller Ethernet Communication Function

What's in This Chapter

Description 33

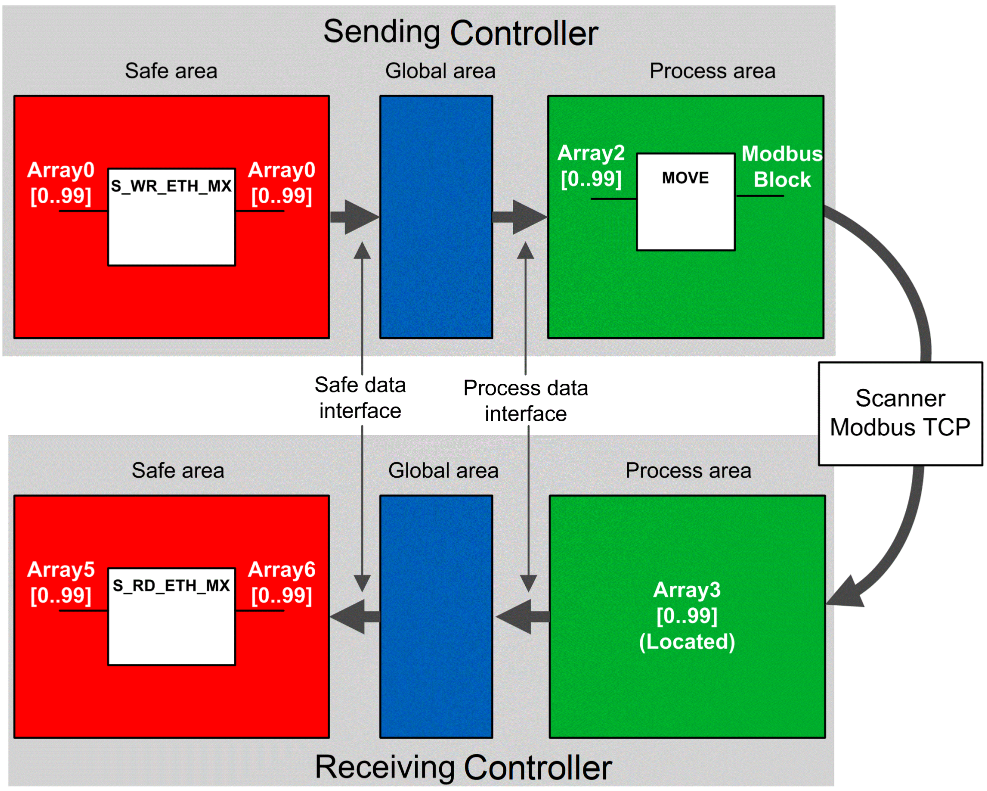
Introduction

This chapter describes the `S_WR_ETH_MX` block.

Description

Function Description

Use the `S_WR_ETH_MX` function block together with the `S_RD_ETH_MX` function block to complete a transfer of safety data between safety controllers with firmware 3.10 or earlier. Each safety controller is designed to isolate safety data from process data by passing data through a global area that is accessible to both the safety and process areas. In this way process functions do not directly impact safety data.



`EN` and `ENO` can be configured as additional parameters.

The `S_WR_ETH_MX` function block is used by a sender controller to:

- Calculate the CRC of the safety data to be sent.
- Calculate the time stamp to be sent with the data. The time stamp is based on a time base reference that is periodically updated with the time received from an NTP server.

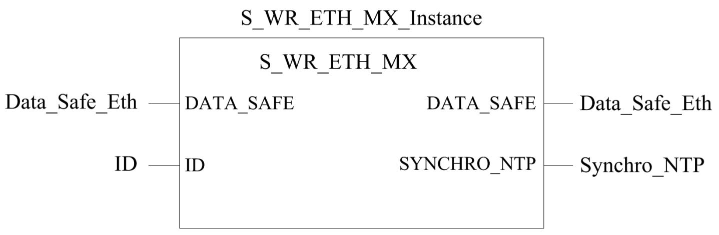
- Generate a unique message identifier that is added to the CRC to help prevent masquerade and insertion attacks on the transmission of safety data array.
- Insert the calculated CRC and time stamp at the end of the data array to be sent.

The `S_WR_ETH_MX` function block has to be called as part of each cycle in the sender controller. Within the cycle logic, it has to be executed after all required modifications have been performed on the data to be sent. The data to be sent must be modified after the execution of the `S_WR_ETH_MX` function block, otherwise the CRC information appended to the transmitted data will not be correct and the safe peer-to-peer communication will not succeed.

NOTE: For additional information, refer to chapter *Peer to Peer Communications* (see Modicon M580, Safety Manual).

Representation in FBD

Representation



Input/Output Parameter

Description of the input/output parameter:

Parameter	Data Type	Meaning
`DATA_SAFE`	`ARRAY [0..99] of INT`	Array of safety data variables. Composed of “User safety data” (from index 0 to 90) and “Reserved data” (from index 91 to 99). **NOTE:** Define these data variables as shared output variables, each with an equivalent global variable, using the **Safety Data Interface** tab in Control Expert.

Input Parameter

Description of the input parameter:

Parameter	Data Type	Meaning
ID	INT	Communication identifier. The ID value is used to calculate the CRC. It is unique and has the same value as the value used by the sender. **NOTE:** Assign a unique value to the ID parameter that identifies the safe peer-to-peer communication between a sender and a receiver.

Output Parameter

Description of output parameter:

Parameter	Data Type	Meaning
SYNCHRO_NTP	BOOL	• 1: Indicates that NTP time synchronization is healthy. • 0: Indicates NTP time synchronization is not healthy.

DATA_SAFE Array Description

Use the **Interface** tabs in both the **Safety Data Editor** and the **Process Data Editor** in Control Expert to make the link between the process variables and the safety-related variables.

Linking process and safety-related variables in this manner makes it possible to:

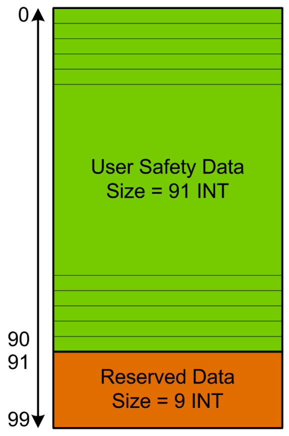
- Transfer the value of safety-related variables to process variables, via linked global variables.
- Send variable values from the process area of the sending controller to the process area of the receiving controller, via explicit messaging over Modbus TCP.

DATA_SAFE array is composed of two zones:

- The **User Safety Data** zone contains the data from the safe area of the controller. This zone starts at index 0 and finishes at index 90.
- The **Reserved Data** zone is reserved for auto-generated diagnostic data, including a CRC and time-stamp. This data is used by the receiving controller to determine if the data contained in the **User Safety Data** zone is safe or not. This zone starts at index 91 and finishes at index 99.

  NOTE: Do not write in the **Reserved Data** zone.

`DATA_SAFE` array (array[0..99] of INT) structure representation:



S_RD_ETH_MX2: Safe Mx80 Controller-to-Controller Ethernet Communication Function

What's in This Chapter

Description 38

Introduction

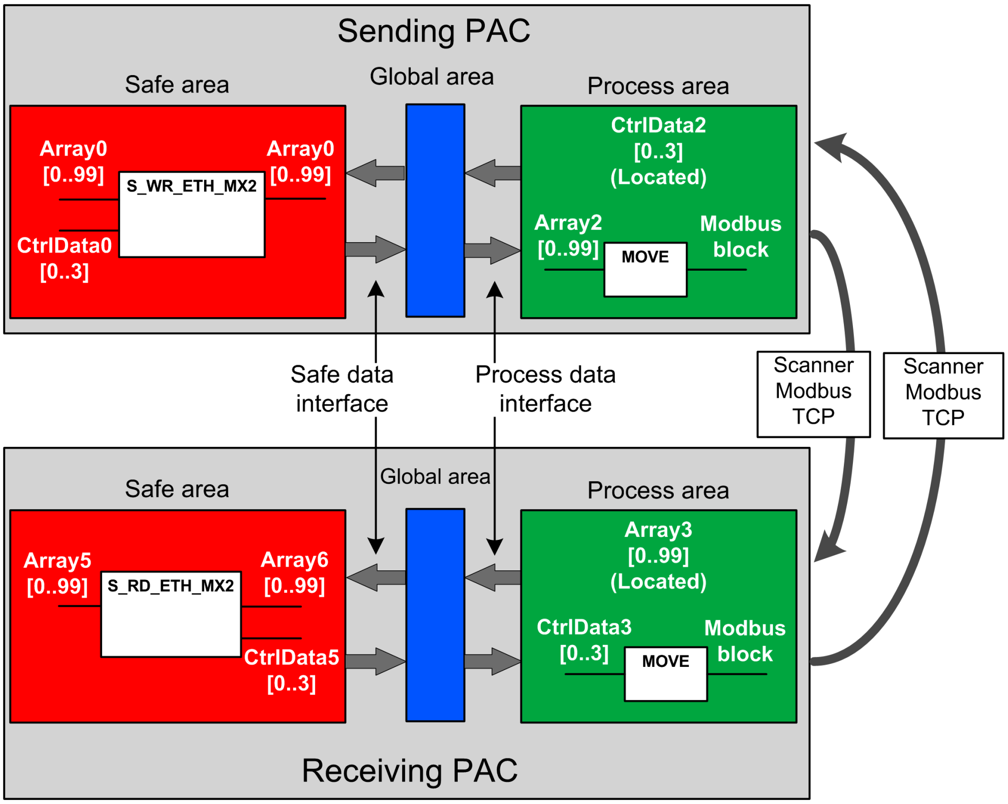
This chapter describes the `S_RD_ETH_MX2` block.

Description

Function Description

Use the S_RD_ETH_MX2 function block together with the S_WR_ETH_MX2 function block to complete a transfer of safety data between safety controllers with firmware 3.20 or later. Each safety controller is designed to isolate safety data from process data by passing data through a global area that is accessible to both the safety and process areas. In this way, process functions do not directly impact safety data.

NOTE: When configuring safe communications between M580 Safety controllers and Quantum Safety controllers, use the S_RD_ETH_MX, page 26 and S_WR_ETH_MX, page 33 function blocks, instead of the S_RD_ETH_MX2 and S_WR_ETH_MX2 function blocks.



The S_RD_ETH_MX2 function block is used by a receiver controller to copy the data received in the process area to the safety area and validate the accuracy of the received data.

EN and ENO can be configured as additional parameters.

The `S_RD_ETH_MX2` function block:

- Copies the data received in the `INPUT_DATA` register to the `OUTPUT_DATA_SAFE` register if it passes the following tests:
    - The function block verifies the CRC of the last data packet received, via explicit messaging over Modbus TCP. If the CRC is not correct, the data is considered as corrupted and it is not written to the `OUTPUT_DATA_SAFE` register in the safety area.
    - The function block verifies the last data received to determine if it is more recent than the data already written in the `OUTPUT_DATA_SAFE` register in the safety area (by comparing time stamps). If the last data received is not more recent, it is not copied to the `OUTPUT_DATA_SAFE` register in the safety area.
- Verifies the age of the data in the safety area. If the age is higher than a configurable maximum value set in the `SAFETY_CONTROL_TIMEOUT` input register, the data is declared corrupted and the `HEALTH` bit is set to 0.

    NOTE: The data age is the time difference between the time when the data is computed in the sender controller and the time when the data is checked in the receiver controller. The time base reference is periodically updated with the monotonic time received from the operating system.

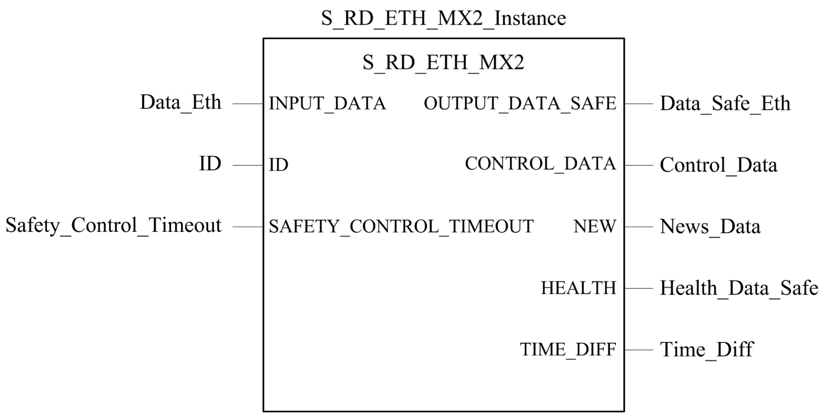
  If the `HEALTH` bit is set to 0, the data available in the `OUTPUT_DATA_SAFE` array is considered as corrupted. In this case, take appropriate measures.

Design your application so that the `S_RD_ETH_MX2` function block is called at each cycle in the receiver controller SAFE task and executes before the data usage in the SAFE task cycle.

NOTE: For additional information, refer to chapter *Peer to Peer Communications* (see Modicon M580, Safety Manual).

Representation in FBD

Representation:



Input Parameters

Description of the input parameters:

Parameter	Data Type	Meaning
`INPUT_DATA`	`ARRAY[0..99] of INT`	Array of data variables received in the global memory area via explicit messaging over Modbus TCP. Composed of “User safety data” (from index 0 to 90) and “Reserved data” (from index 91 to 99). **NOTE:** Define these data variables as shared input variables, each with an equivalent global variable, using the **Safety Data Interface** tab in Control Expert.
`ID`	`INT`	Communication identifier. The unique `ID` value used to calculate the CRC. It is set to the same value as the value used by the sender. **NOTE:** Assign a unique value to the `ID` parameter that identifies the safe peer-to-peer communication between a sender and a receiver.
`SAFETY_CONTROL_TIMEOUT`	`UINT`	Time out value (in ms). Used to check the age of the data in the safety area and determine if that data is to be considered safe. Refer to the topic Calculating a SAFETY_CONTROL_ TIMEOUT Value, below.

Output Parameters

Description of the output parameters:

Parameter	Data Type	Meaning
OUTPUT_DATA_SAFE	ARRAY[0..99] of INT	Safety data variables array structure. Composed of “User safety data” (from index 0 to 90) and “Reserved data” (from index 91 to 99).
CONTROL_DATA	ARRAY[0..3] of INT	Contains ID and time stamp based on system monotonic time to send to correspondent sender (S_WR_ETH_MX2 function block).
NEW	BOOL	• 1: Indicates that a new set of data has been refreshed in OUTPUT_DATA_SAFE during the present cycle. • 0: Indicates no new safe data has been refreshed.
HEALTH	BOOL	• 1: Indicates the data in the **User Safety Data** area is safe. • 0: Indicates the data in the **User Safety Data** area is not safe.
TIME_DIFF	INT	Returns the age (in ms) of the data received and written in the OUTPUT_DATA_SAFE output parameter. Set to -1 if the internal NTP time is not initialized or if no correct data has yet been received.

INPUT_DATA and OUTPUT_DATA_SAFE Arrays Description

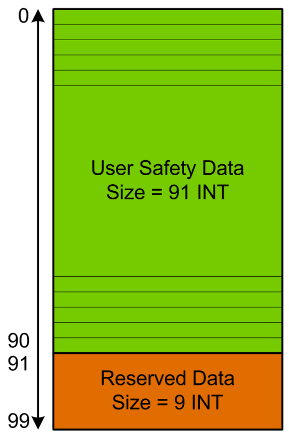
The INPUT_DATA arrays consist of data coming from the process data memory area. The OUTPUT_DATA_SAFE arrays consist of safety-related variables. Use the **Safety Data Interface** and the **Process Data Interface** tabs in Control Expert to make the link between the process variables and the safety-related variables.

INPUT_DATA and OUTPUT_DATA_SAFE arrays are composed of 2 zones:

- The **User Safety Data** zone contains user data. This zone starts at index 0 and finishes at index 90.
- The **Reserved Data** zone is reserved for auto-generated diagnostic data, including a CRC and time-stamp. This data is used by the receiving controller to determine if the data contained in the **User Safety Data** zone is safe or not. This zone starts at index 91 and finishes at index 99.

  NOTE: Do not write in the **Reserved Data** zone, as doing so will overwrite the auto-generated diagnostic data.

`INPUT_DATA` and `OUTPUT_DATA_SAFE` arrays (array[0..99] of INT) structure representation:



`CONTROL_DATA` Array Description

The `CONTROL_DATA` array has to be linked with variables in “Global” area (defined through the “Safety Data Interface”) and then, “Global” variables have to be linked to located variables in “Process” area (defined through the “Process Data Interface”) so that the data can be sent by IO Scanner to the correspondent sender.

Calculating a SAFETY_CONTROL_TIMEOUT Value

When calculating a `SAFETY_CONTROL_TIMEOUT` value, consider the following:

- Minimum value: `SAFETY_CONTROL_TIMEOUT` > 2*T1
- Typical value: `SAFETY_CONTROL_TIMEOUT` > 3 * T1

T1 = $\text{controller}_{\text{sender}}$ MAST cycle time + $\text{controller}_{\text{sender}}$ SAFE cycle time + Repetitive_rate + Network transmission time + $\text{controller}_{\text{receiver}}$ MAST cycle time + $\text{controller}_{\text{receiver}}$ SAFE cycle time

Where:

- *$\text{controller}_{\text{sender}}$ MAST cycle time* is the MAST cycle time of the sender controller.
- *$\text{controller}_{\text{sender}}$ SAFE cycle time* is the SAFE cycle time of the sender controller.
- *Repetitive_rate* is the time rate for the I/O scanner write query from the sender controller to the receiver controller.

- *Network transmission time* is the time consumed on the Ethernet network for the data transmission from the sender controller to the receiver controller.
- *controller$_{receiver}$ MAST cycle time* is the MAST cycle time of the receiver controller.
- *controller$_{receiver}$ SAFE cycle time* is the SAFE cycle time of the receiver controller.

Note that the value defined for the `SAFETY_CONTROL_TIMEOUT` parameter has a direct effect on the robustness and availability of the safe peer-to-peer communication. If the `SAFETY_CONTROL_TIMEOUT` parameter value greatly exceeds T1, the communication will be tolerant to various delays (for example network delays) or corrupted data transmissions.

Configure your Ethernet network so the load that does not cause an excessive delay on the network during data transmission, which could lead to the expiration of the timeout. To help safeguard your safe peer-to-peer communications from any excessive delays due to other non-safety data transmitted on the same network, consider using a dedicated Ethernet network for the safe peer-to-peer protocol.

When commissioning your project, you have to estimate the safe peer-to-peer communication performance by checking the values provided in the output parameter `TIME_DIFF` and evaluating the margin using the value defined in the `SAFETY_CONTROL_TIMEOUT` parameter.

Understanding the `HEALTH` Bit

When the `HEALTH` bit value equals:

- 1: The integrity of the data is correct (CRC) and the age of the data is less than the value set in the `SAFTETY_CONTROL_TIMEOUT` input register.

  NOTE: The age of the data considered is the time between:

  - The beginning of the cycle where the data are computed in the sender controller.
  - The beginning of the cycle where the data are checked in the receiver controller.
- 0: New valid data are not received in the required time interval (the timer expires and the `HEALTH` bit is set to 0).

  NOTE: If the `HEALTH` bit is set to 0, the data in the output array `OUTPUT_DATA_SAFE` is considered to be corrupted; take appropriate measures.

S_WR_ETH_MX2: Safe Mx80 Controller-to-Controller Ethernet Communication Function

What's in This Chapter

Description 45

Introduction

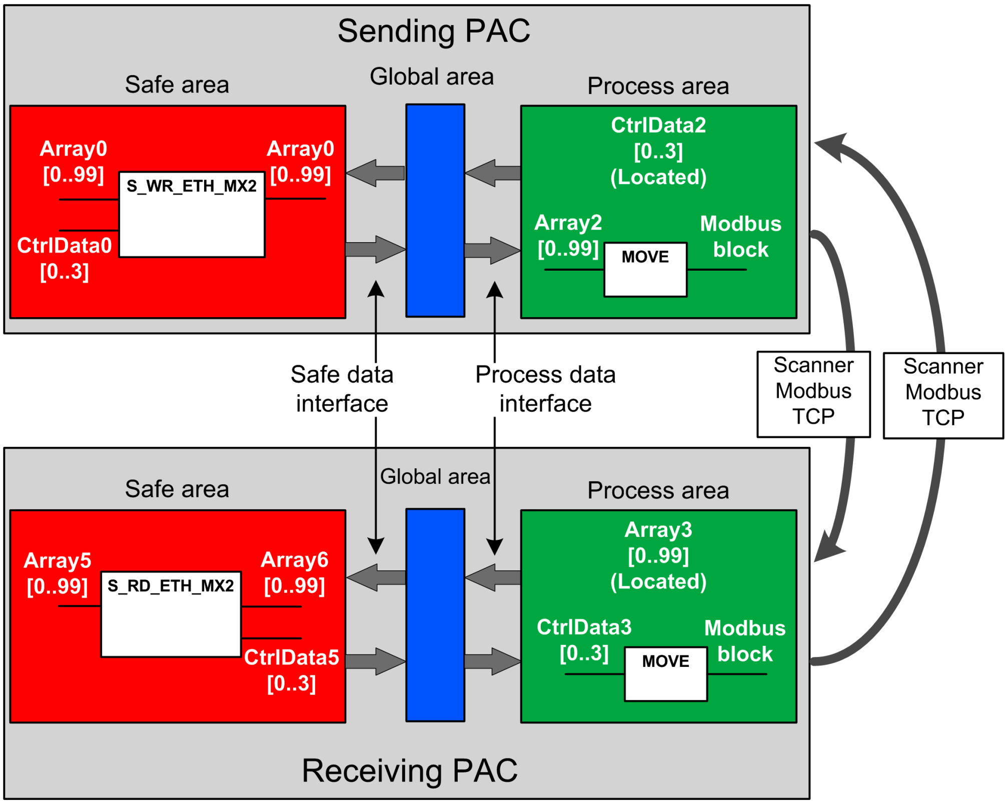
This chapter describes the `S_WR_ETH_MX2` block.

Description

Function Description

Use the `S_WR_ETH_MX2` function block together with the `S_RD_ETH_MX2` function block to complete a transfer of safety data between safety controllers with firmware 3.20 or later. Each safety controller is designed to isolate safety data from process data by passing data through a global area that is accessible to both the safety and process areas. In this way process functions do not directly impact safety data.

NOTE: When configuring safe communications between M580 Safety controllers and Quantum Safety controllers, use the S_RD_ETH_MX, page 26 and S_WR_ETH_MX, page 33 function blocks, instead of the S_RD_ETH_MX2 and S_WR_ETH_MX2 function blocks.



`EN` and `ENO` can be configured as additional parameters.

The `S_WR_ETH_MX2` function block is used by a sender controller to:

- Calculate the CRC of the safety data to be sent.

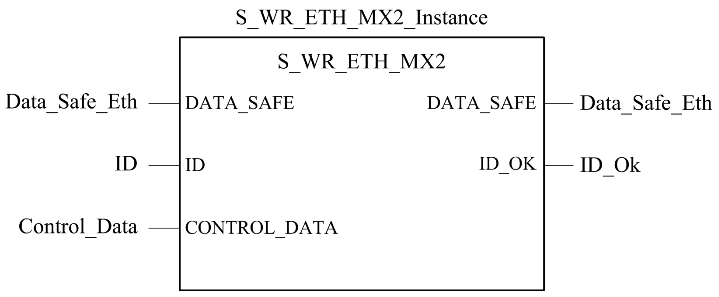
- Calculate the time stamp to be sent with the data. The time stamp is based on a time base reference that is periodically updated with the monotonic time received from the operating system.
- Generate a unique message identifier that is added to the CRC to help prevent masquerade and insertion attacks on the transmission of safety data array.
- Insert the calculated CRC and time stamp at the end of the data array to be sent.

The `S_WR_ETH_MX2` function block has to be called as part of each cycle in the sender controller. Within the cycle logic, it has to be executed after all required modifications have been performed on the data to be sent. The data to be sent must not be modified after the execution of the `S_WR_ETH_MX2` function block, otherwise the CRC information appended to the transmitted data will not be correct and the safe peer-to-peer communication will not succeed.

NOTE: For additional information, refer to chapter *Peer to Peer Communications* (see Modicon M580, Safety Manual).

Representation in FBD

Representation



Input/Output Parameter

Description of the input/output parameter:

Parameter	Data Type	Meaning
`DATA_SAFE`	`ARRAY [0..99] of INT`	Array of safety data variables. Composed of “User safety data” (from index 0 to 90) and “Reserved data” (from index 91 to 99). **NOTE:** Define these data variables as shared output variables, each with an equivalent global variable, using the **Safety Data Interface** tab in Control Expert.

Input Parameter

Description of the input parameter:

Parameter	Data Type	Meaning
ID	INT	Communication identifier. The ID value is used to calculate the CRC. It is unique and has the same value as the value used by the sender. **NOTE:** Assign a unique value to the ID parameter that identifies the safe peer-to-peer communication between a sender and a receiver.
CONTROL_DATA	ARRAY [0..3] of INT	Contains ID and time stamp based on system monotonic time coming from correspondent receiver (S_RD_ETH_MX2 function block).

Output Parameter

Description of output parameter:

Parameter	Data Type	Meaning
ID_OK	BOOL	• 1: Indicates that ID in data on CONTROL_DATA input and data on ID input match. • 0: Indicates that ID in data on CONTROL_DATA input and data on ID input do not match.

DATA_SAFE Array Description

Use the **Interface** tabs in both the **Safety Data Editor** and the **Process Data Editor** in Control Expert to make the link between the process variables and the safety-related variables.

Linking process and safety-related variables in this manner makes it possible to:

- Transfer the value of safety-related variables to process variables, via linked global variables.
- Send variable values from the process area of the sending controller to the process area of the receiving controller, via explicit messaging over Modbus TCP.

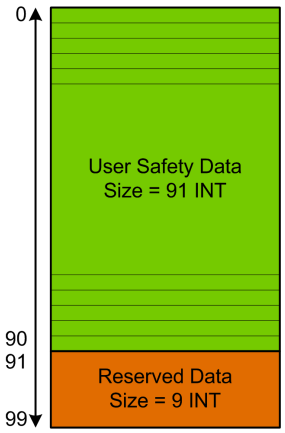
DATA_SAFE array is composed of two zones:

- The **User Safety Data** zone contains the data from the safe area of the controller. This zone starts at index 0 and finishes at index 90.

- The **Reserved Data** zone is reserved for auto-generated diagnostic data, including a CRC and time-stamp. This data is used by the receiving controller to determine if the data contained in the **User Safety Data** zone is safe or not. This zone starts at index 91 and finishes at index 99.

  NOTE: Do not write in the **Reserved Data** zone.

`DATA_SAFE` array (array[0..99] of INT) structure representation:



Comparison

What's in This Part

S_EQ_***: Equal To 50
S_GE_***: Greater Than or Equal To 53
S_GT_***: Greater Than 56
S_LE_***: Less Than or Equal To 59
S_LT_***: Less Than 62
S_NE_***: Not Equal To 65

Introduction

This section describes the elementary functions and elementary function blocks of the `Comparison` family.

S_EQ_***: Equal To

What's in This Chapter

Description 50

Introduction

This chapter describes the S_EQ_*** block.

Description

Function Description

The function verifies the inputs for equality, i.e. the output becomes 1 if there is equality at all inputs; otherwise, the output remains at 0.

The data types of all input values must be identical.

The number of inputs can be increased to a maximum of 32.

When comparing variables of the BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT and REAL data types, the values are compared with each other.

EN and ENO can be configured as additional parameters.

Formula

$\mathrm{OUT} = 1$, if $(\mathrm{IN1} = \mathrm{IN2})$ & $(\mathrm{IN2} = \mathrm{IN3})$ & .. & $(\mathrm{IN}_{(n-1)} = \mathrm{IN}_n)$

Available Functions

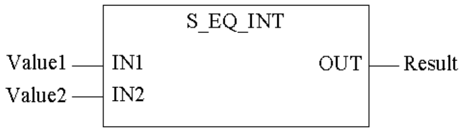
List of available functions

- S_EQ_BOOL
- S_EQ_BYTE

- S_EQ_WORD
- S_EQ_DWORD
- S_EQ_INT
- S_EQ_DINT
- S_EQ_UINT
- S_EQ_UDINT
- S_EQ_REAL

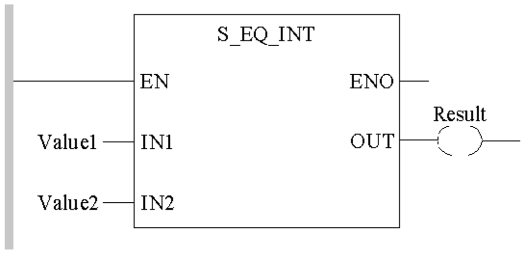
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
IN1	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL	Input 1 value
IN2	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL	Input 2 value
INn	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL	Input n value n = maximum 32

Description of the output parameter

Parameter	Data Type	Meaning
OUT	BOOL	output

S_GE_***: Greater Than or Equal To

What’s in This Chapter

Description 53

Introduction

This chapter describes the `S_GE_***` block.

Description

Function Description

The function verifies the values of successive inputs for a decreasing sequence or equality.

The data types of all input values must be identical.

The number of inputs can be increased to a maximum of 32.

When comparing variables of the `BOOL`, `BYTE`, `WORD`, `DWORD`, `INT`, `DINT`, `UINT`, `UDINT` and `REAL` data types, the values are compared with each other.

`EN` and `ENO` can be configured as additional parameters.

Formula

$\mathtt{OUT} = 1$, if $(\mathtt{IN1} \geq \mathtt{IN2})$ & $(\mathtt{IN2} \geq \mathtt{IN3})$ & .. & $(\mathtt{IN}_{(n-1)} \geq \mathtt{IN}_{n})$

Available Functions

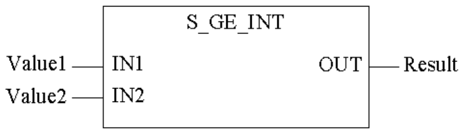
List of available functions

- S_GE_BOOL
- S_GE_BYTE
- S_GE_WORD

- S_GE_DWORD
- S_GE_INT
- S_GE_DINT
- S_GE_UINT
- S_GE_UDINT
- S_GE_REAL

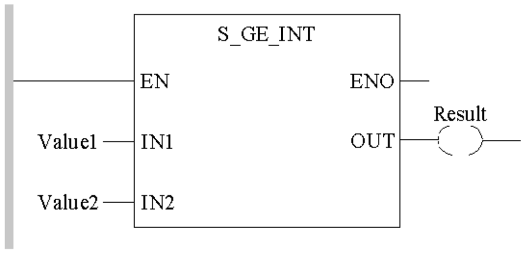
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
IN1	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL	Input 1 value
IN2	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL	Input 2 value
INn	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL	Input n value n = maximum 32

Description of the output parameter

Parameter	Data Type	Meaning
OUT	BOOL	output

S_GT_***: Greater Than

What's in This Chapter

Description 56

Introduction

This chapter describes the `S_GT_***` block.

Description

Function Description

The function verifies the values of successive inputs for a decreasing sequence.

The data types of all input values must be identical.

The number of inputs can be increased to a maximum of 32.

When comparing variables of the `BOOL`, `BYTE`, `WORD`, `DWORD`, `INT`, `DINT`, `UINT`, `UDINT` and `REAL` data types, the values are compared with each other.

`EN` and `ENO` can be configured as additional parameters.

Formula

$\mathtt{OUT} = 1$, if $(\mathtt{IN1} > \mathtt{IN2})\ \&\ (\mathtt{IN2} > \mathtt{IN3})\ \&\ ..\ (\mathtt{IN}_{(n-1)} > \mathtt{IN}_{n})$

Available Functions

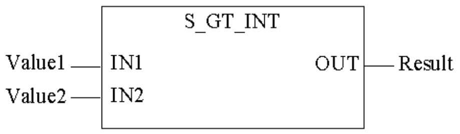
List of available functions

- S_GT_BOOL
- S_GT_BYTE
- S_GT_WORD

- S_GT_DWORD
- S_GT_INT
- S_GT_DINT
- S_GT_UINT
- S_GT_UDINT
- S_GT_REAL

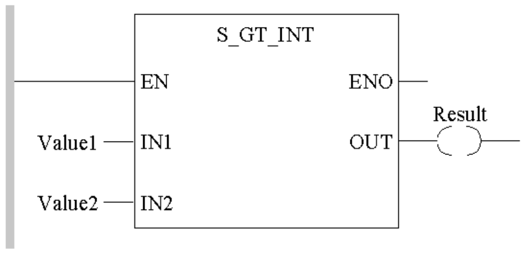
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
IN1	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL	Input 1 value
IN2	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL	Input 2 value
INn	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL	Input n value n = maximum 32

Description of the output parameter

Parameter	Data Type	Meaning
OUT	BOOL	output

S_LE_***: Less Than or Equal To

What's in This Chapter

Description 59

Introduction

This chapter describes the `S_LE_***` block.

Description

Function Description

The function verifies the values of successive inputs for an increasing sequence or equality.

The data types of all input values must be identical.

The number of inputs can be increased to a maximum of 32.

When comparing variables of the `BOOL`, `BYTE`, `WORD`, `DWORD`, `INT`, `DINT`, `UINT`, `UDINT` and `REAL` data types, the values are compared with each other.

`EN` and `ENO` can be configured as additional parameters.

Formula

$\mathtt{OUT} = 1$, if $(\mathtt{IN1} \leq \mathtt{IN2})$ & $(\mathtt{IN2} \leq \mathtt{IN3})$ & .. & $(\mathtt{IN}_{(n-1)} \leq \mathtt{IN}_{n})$

Available Functions

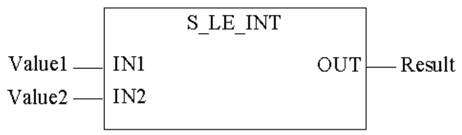
List of available functions

- S_LE_BOOL
- S_LE_BYTE
- S_LE_WORD

- S_LE_DWORD
- S_LE_INT
- S_LE_DINT
- S_LE_UINT
- S_LE_UDINT
- S_LE_REAL

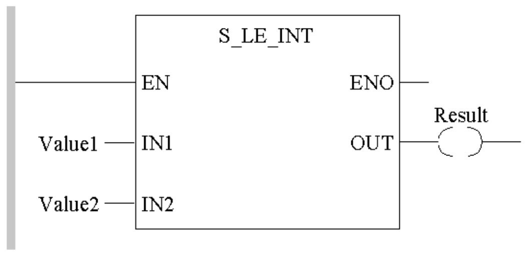
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
IN1	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL	Input 1 value
IN2	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL	Input 2 value
INn	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL	Input n value n = maximum 32

Description of the output parameter

Parameter	Data Type	Meaning
OUT	BOOL	output

S_LT_***: Less Than

What's in This Chapter

Description 62

Introduction

This chapter describes the S_LT_*** block.

Description

Function Description

The function verifies the values of successive inputs for an increasing sequence.

The data types of all input values must be identical.

The number of inputs can be increased to a maximum of 32.

When comparing variables of the BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT and REAL data types, the values are compared with each other.

EN and ENO can be configured as additional parameters.

Formula

OUT = 1, if $(IN1 < IN2)$ & $(IN2 < IN3)$ & .. & $(IN_{(n-1)} < IN_n)$

Available Functions

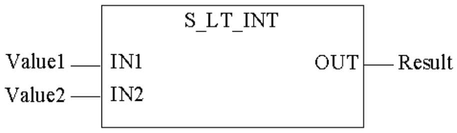
List of available functions

- S_LT_BOOL
- S_LT_BYTE
- S_LT_WORD

- S_LT_DWORD
- S_LT_INT
- S_LT_DINT
- S_LT_UINT
- S_LT_UDINT
- S_LT_REAL

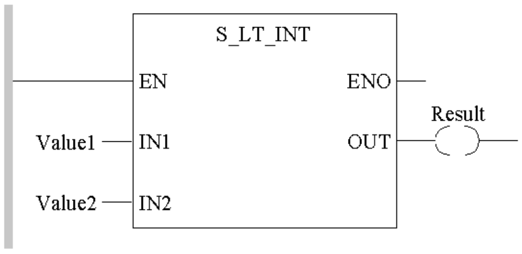
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
IN1	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL	Input 1 value
IN2	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL	Input 2 value
INn	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL	Input n value n = maximum 32

Description of the output parameter

Parameter	Data Type	Meaning
OUT	BOOL	output value

S_NE_***: Not Equal To

What's in This Chapter

Description 65

Introduction

This chapter describes the `S_NE_***` block.

Description

Function Description

The function verifies the input values for inequality.

The data types of the input values must be identical.

`EN` and `ENO` can be configured as additional parameters.

Formula

`OUT` = 1, if `IN1` < > `IN2`

Available Functions

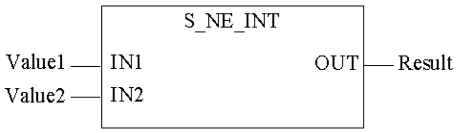
List of available functions

- S_NE_BOOL
- S_NE_BYTE
- S_NE_WORD
- S_NE_DWORD
- S_NE_INT
- S_NE_DINT

- S_NE_UINT
- S_NE_UDINT
- S_NE_REAL

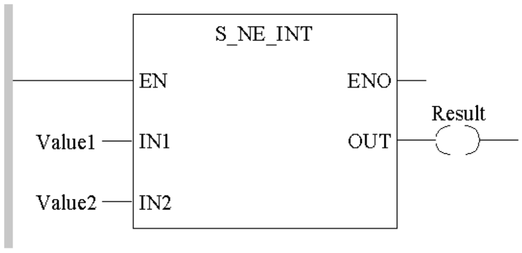
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
IN1	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL	Input 1 value
IN2	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT, REAL	Input 2 value

Description of the output parameter

Parameter	Data Type	Meaning
`OUT`	`BOOL`	output

Actuator

What's in This Part

S_EDM: Actuator Error Detection Monitoring 69
S_ENABLE_SWITCH: Three Position Enable Switch 79
S_ESPE: Electro-Sensitive Protective Equipment 87
S_GUARD_LOCKING: Guard Lock Control 95
S_GUARD_MONITORING: Guard Lock Monitoring.................. 103
S_MODE_SELECTOR: Safety Mode Switch............................ 113
S_OUTCONTROL: Output Driver.. 121

Introduction

This section describes the elementary functions and elementary function blocks of the `Actuator` family.

S_EDM: Actuator Error Detection Monitoring

What's in This Chapter

Description 69

Introduction

This chapter describes the `S_EDM` block.

Description

Function Description

The `S_EDM` function block:

- Controls a safety-related output and monitors controlled actuators.
- Monitors the initial state of the actuators via the feedback signals `S_EDM1` and `S_EDM2` before enabling the actuators.
- Monitors the switching state of the actuators (`MonitoringTime`) after the actuators have been enabled by the function block.

Use two single feedback signals to diagnose the connected actuators. The function block uses a common feedback signal from the two connected actuators for a restricted yet simple diagnostic function of the connected actuators. Connect this common signal to both parameter `S_EDM1` and parameter `S_EDM2` so that these two parameters can be controlled by the same signal.

⚠WARNING

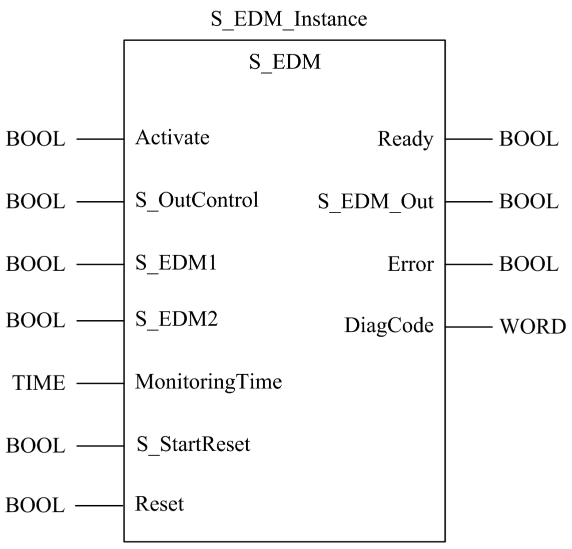
UNINTENDED EQUIPMENT OPERATION

Activate the S_StartReset and S_AutoReset inputs only after you verify that no hazardous situation can occur if the system is started.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Representation in FBD

Representation



Input Parameters

Parameter	**Data type**	**Init Value**	**Meaning**
`Activate`	`BOOL`	FALSE	The variable or constant value indicating activation state of the function block. This parameter can be connected to the variable, which represents the status (Active or Not Active) of the safety-related device. This causes no irrelevant diagnostic information to be generated if a device is disabled: • When FALSE, the output variables are set to the initial values. • Assign a static TRUE signal if no device is connected.
`S_OutControl`	`BOOL`	FALSE	The variable control signal of the preceding safety-related function blocks. Typical function block signals from the library (e.g., `S_OutControl`, `S_Two_Hand_Control_Type_II`): • FALSE: Disable the safety-related output (`S_EDM_Out`). • TRUE: Enable the safety-related output (`S_EDM_Out`).

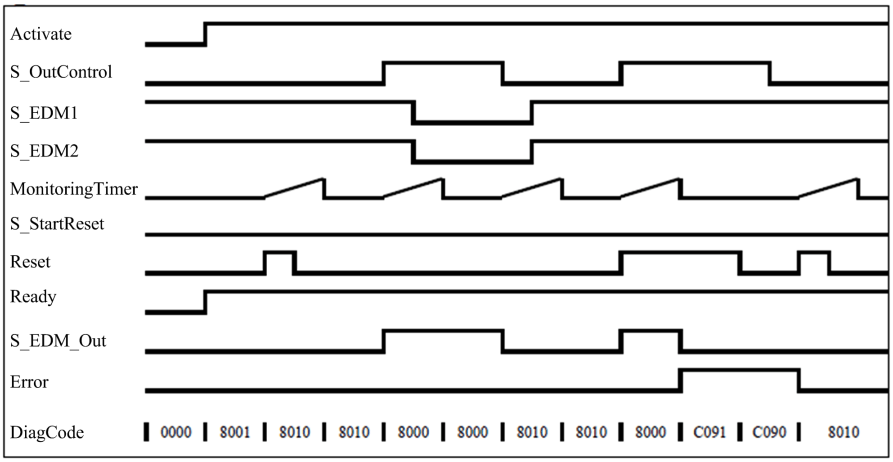
Parameter	Data type	Init Value	Meaning
`S_EDM1`	`BOOL`	FALSE	The variable feedback signal of the first connected actuator: • FALSE: Switching state of the first connected actuator. • TRUE: Initial state of the first connected actuator.
`S_EDM2`	`BOOL`	FALSE	The variable feedback signal of the second connected actuator. If using only one signal in the application, use a graphic connection to jumper the `S_EDM1` and `S_EDM2` parameters. `S_EDM1` and `S_EDM2` are then controlled by the same signal: • FALSE: Switching state of the second connected actuator. • TRUE: Initial state of the second connected actuator.
`Monitoring-Time`	`TIME`	T#0ms	Maximum response time of the connected and monitored actuators. A constant value.
`S_StartReset`	`BOOL`	FALSE	The variable or constant reset setting: • FALSE: Manual reset when system is started (warm or cold). • TRUE: Automatic reset when system is started (warm or cold). **NOTE:** Activate this function only after you have confirmed that no hazard can occur at the start of the PES. Use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system or application measures to guard against unexpected (or unintended) start-up. Refer to the hazard message in the Function Description above.
`Reset`	`BOOL`	FALSE	The variable value indicating reset of the state machine, coupled with detected error and status messages as indicated in the `DiagCode` parameter, after the underlying cause of the detected error has been removed. This behavior is designed as a detected error reset. **NOTE:** This function is only active on a signal change from FALSE to TRUE.

Output Parameters

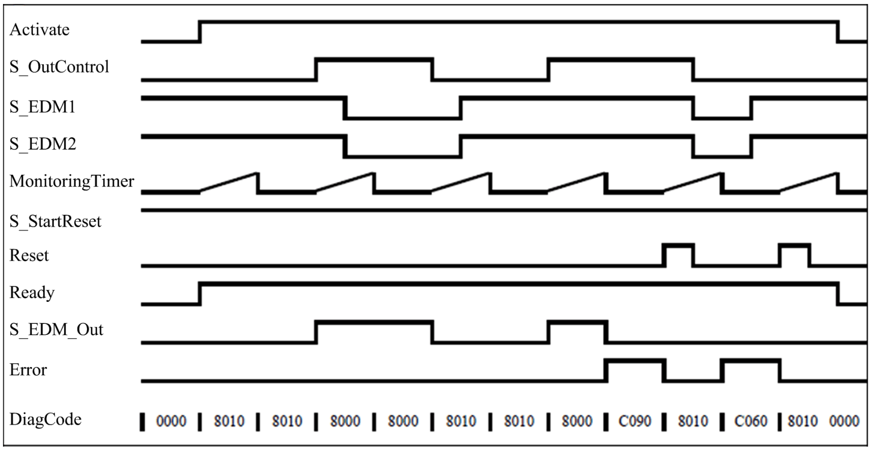
Parameter	Data type	Init Value	Meaning
Ready	BOOL	FALSE	• TRUE indicates that the function block is activated and the output results are valid. • FALSE indicates that the function block is not active and the program is not executed. **NOTE:** This parameter can be useful in debug mode or to activate/deactivate additional function blocks, as well as for further processing in the functional program.
S_EDM_Out	BOOL	FALSE	Controls the actuator. The result is monitored by the feedback signal S_EDMx: • FALSE: Disable connected actuators. • TRUE: Enable connected actuators.
Error	BOOL	FALSE	Function block detected error message.
DiagCode	WORD	0000 hex	Function block diagnostic code.

Typical Timing Diagrams

S_StartReset = FALSE

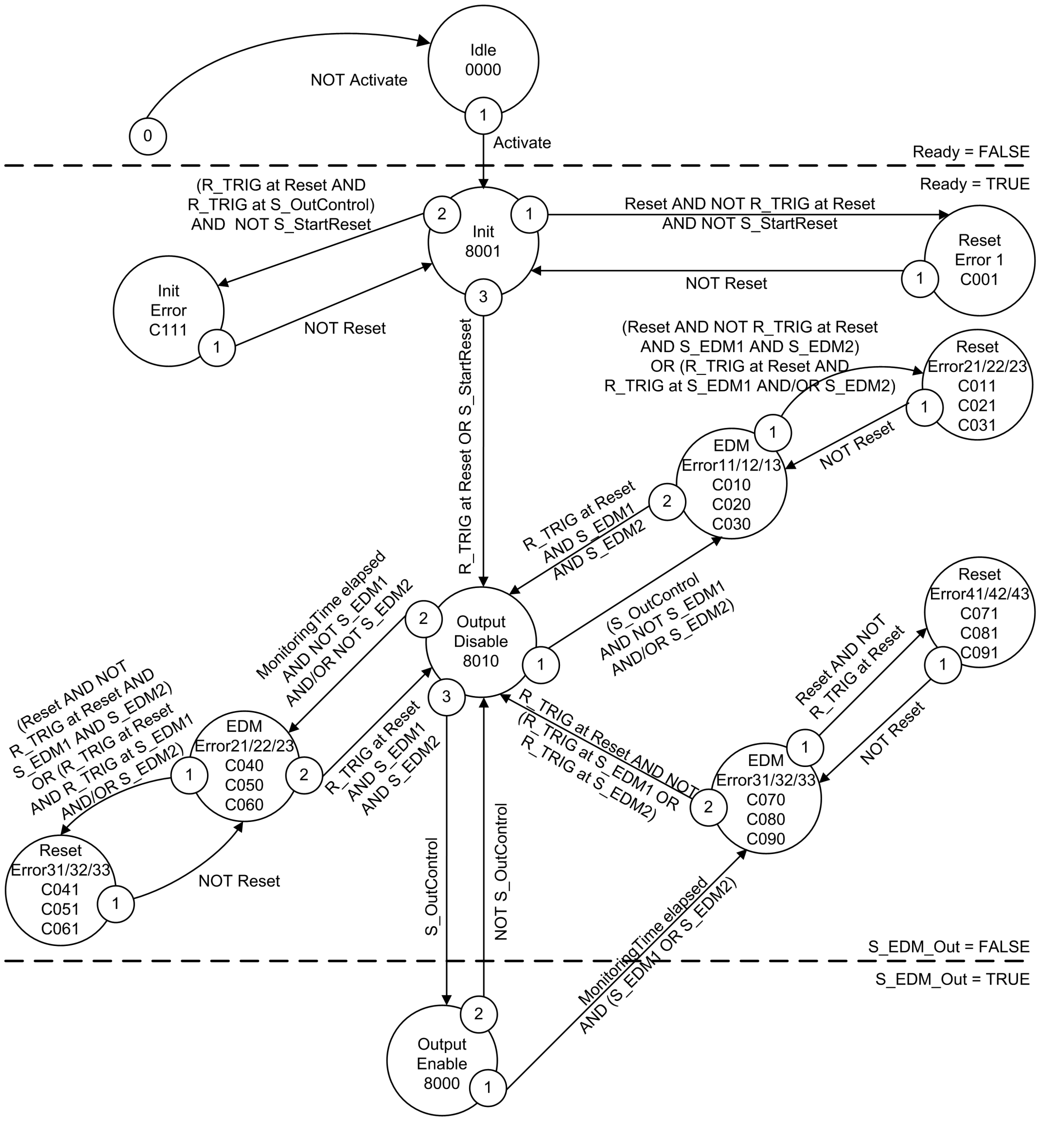


`S_StartReset` = TRUE



State Diagram

The following diagram describes the state transitions of the `S_EDM` function block:



Source: *PLCopen - Technical Committee 5, Safety Software, Technical Specification, Part 1: Concepts and Function Blocks, Version 1.0*.

NOTE: The transition to the Idle state from any other state, occurring because Activate = FALSE, is not depicted. Such a transition has the highest priority (0).

Error Detection

The following conditions force a transition to the detected error state:

- Invalid static signal in the process.
- Invalid `S_EDM1` or `S_EDM2` signal in the process.
- `S_OutControl` and `Reset` are incorrectly interconnected as a result of a programming error.

Detected Error Management

When an error is detected, the outputs are as follows:

- `S_EDM_Out` is set to FALSE and remains in the defined safe state.
- Reset a detected EDM error message by means of a rising edge at `Reset`.
- A detected `Reset error` message can be reset by setting `Reset` to FALSE.

After block activation, the optional start-up inhibit can be reset by a rising edge at the `Reset` input.

When returning a detected error message, the `DiagCode` parameter can present one of the following detected error values:

DiagCode	**State Name**	**State Description and Output Settings**
C001	Reset Error 1	Static `Reset` signal in state 8001: • `S_EDM_Out` = FALSE • `Error` = TRUE
C011	Reset Error 21	Static `Reset` signal or same signals at `S_EDM1` and `Reset` (rising edge at `Reset` and `S_EDM1` at the same time) in state C010: • `S_EDM_Out` = FALSE • `Error` = TRUE
C021	Reset Error 22	Static `Reset` signal or same signals at `S_EDM2` and `Reset` (rising edge at `Reset` and `S_EDM2` at the same time) in state C020: • `S_EDM_Out` = FALSE • `Error` = TRUE
C031	Reset Error 23	Static `Reset` signal or same signals at `S_EDM1`, `S_EDM2`, and `Reset` (rising edge at `Reset`, `S_EDM1`, and `S_EDM2` at the same time) in state C030: • `S_EDM_Out` = FALSE • `Error` = TRUE

DiagCode	State Name	State Description and Output Settings
C041	Reset Error 31	Static Reset signal or same signals at `S_EDM1` and `Reset` (rising edge at `Reset` and `S_EDM1` at the same time) in state C040: • `S_EDM_Out` = FALSE • `Error` = TRUE
C051	Reset Error 32	Static `Reset` signal or same signals at `S_EDM2` and `Reset` (rising edge at `Reset` and `S_EDM2` at the same time) in state C050: • `S_EDM_Out` = FALSE • `Error` = TRUE
C061	Reset Error 33	Static `Reset` signal or same signals at `S_EDM1`, `S_EDM2`, and Reset (rising edge at `Reset`, `S_EDM1`, and `S_EDM2` at the same time) in state C060: • `S_EDM_Out` = FALSE • `Error` = TRUE
C071	Reset Error 41	Static `Reset` signal in state C070: • `S_EDM_Out` = FALSE • `Error` = TRUE
C081	Reset Error 42	Static `Reset` signal in state C080: • `S_EDM_Out` = FALSE • `Error` = TRUE
C091	Reset Error 43	Static `Reset` signal in state C090: • `S_EDM_Out` = FALSE • `Error` = TRUE
C010	EDM Error 11	The signal at `S_EDM1` is not valid in the initial actuator state. In state 8010 the `S_EDM1` signal is FALSE when enabling `S_OutControl`: • `S_EDM_Out` = FALSE • `Error` = TRUE
C020	EDM Error 12	The signal at `S_EDM2` is not valid in the initial actuator state. In state 8010 the `S_EDM2` signal is FALSE when enabling `S_OutControl`: • `S_EDM_Out` = FALSE • `Error` = TRUE
C030	EDM Error 13	The signals at `S_EDM1` and `S_EDM2` are not valid in the initial actuator states. In state 8010 the `S_EDM1` and `S_EDM2` signals are FALSE when enabling `S_OutControl`: • `S_EDM_Out` = FALSE • `Error` = TRUE
C040	EDM Error 21	The signal at `S_EDM1` is not valid in the initial actuator state. In state 8010 the `S_EDM1` signal is FALSE and the monitoring time has elapsed: • `S_EDM_Out` = FALSE • `Error` = TRUE

DiagCode	State Name	State Description and Output Settings
C050	EDM Error 22	The signal at S_EDM2 is not valid in the initial actuator state. In state 8010 the S_EDM2 signal is FALSE and the monitoring time has elapsed: • S_EDM_Out = FALSE • Error = TRUE
C060	EDM Error 23	The signals at S_EDM1 and S_EDM2 are not valid in the initial actuator states. In state 8010 the S_EDM1 and S_EDM2 signals are FALSE and the monitoring time has elapsed: • S_EDM_Out = FALSE • Error = TRUE
C070	EDM Error 31	The signal at S_EDM1 is not valid in the actuator switching state. In state 8000 the S_EDM1 signal is TRUE and the monitoring time has elapsed: • S_EDM_Out = FALSE • Error = TRUE
C080	EDM Error 32	The signal at S_EDM2 is not valid in the actuator switching state. In state 8000 the S_EDM2 signal is TRUE and the monitoring time has elapsed: • S_EDM_Out = FALSE • Error = TRUE
C090	EDM Error 33	The signals at S_EDM1 and S_EDM2 are not valid in the initial actuator state. In state 80090 the S_EDM1 and S_EDM2 signals are TRUE and the monitoring time has elapsed: • S_EDM_Out = FALSE • Error = TRUE
C111	Init Error	Similar signals at S_OutControl and Reset (R_TRIG at same cycle) detected (may be a programming error): • S_EDM_Out = FALSE • Error = TRUE

Diagnostic Codes Management

When returning a status message, the Error parameter is set to FALSE, and the DiagCode parameter displays one of the following hexadecimal values:

DiagCode	State Name	State Description and Output Settings
0	IDLE	The function block is not active (initial state): • S_EDM_Out = FALSE • Error = FALSE
8001	INIT	Block activation startup inhibit is active. Reset required: • S_EDM_Out = FALSE • Error = FALSE

DiagCode	State Name	State Description and Output Settings
8010	Output Disable	EDM control is not active. Timer starts when state is entered: • `S_EDM_Out` = FALSE • `Error` = FALSE
8000	Output Enable	EDM control is active. Timer starts when state is entered. EDM control is active. Timer starts when state is entered: • `S_EDM_Out` = TRUE • `Error` = FALSE

S_ENABLE_SWITCH: Three Position Enable Switch

What’s in This Chapter

Description 79

Introduction

This chapter describes the `S_ENABLE_SWITCH` block.

Description

Function Description

Use the `S_ENABLE_SWITCH` function block to support the suspension of monitoring by means of a manually operated three-position enable switch, when the corresponding operating mode (for example, limitation of the speed or the power of the motion, or limitation of the range of the motion) has been selected and is active.

You select the corresponding operating mode outside the function block; then the function block evaluates the signals sent by the three-position enable switch.

⚠WARNING

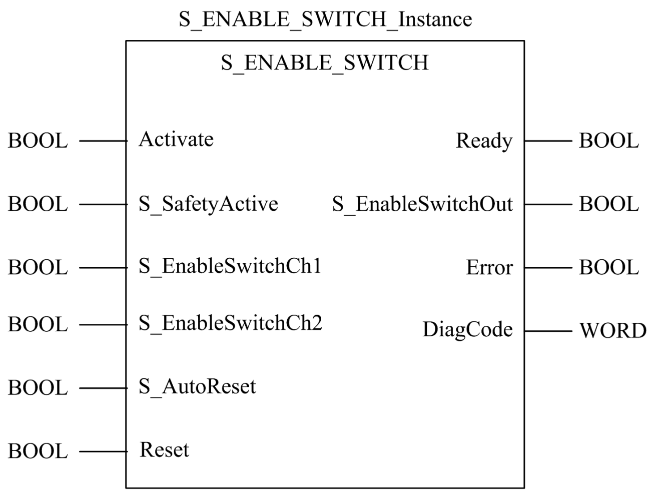
UNINTENDED EQUIPMENT OPERATION

Activate the S_StartReset and S_AutoReset inputs only after you verify that no hazardous situation can occur if the system is started.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Representation in FBD

Representation



Input Parameters

Parameter	Data type	Init Value	Meaning
`Activate`	`BOOL`	FALSE	The variable or constant value indicating activation state of the function block. This parameter can be connected to the variable, which represents the status (Active or Not Active) of the safety-related device. This causes no irrelevant diagnostic information to be generated if a device is disabled: • When FALSE, the output variables are set to the initial values. • Assign a static TRUE signal if no device is connected.
`S_SafetyActive`	`BOOL`	FALSE	A variable or constant input confirmation signal (feedback signal) indicating that the selected Safe Operation Mode is active: • FALSE: Safe Operation Mode is not active. • TRUE: Safe Operation Mode is active.
`S_ EnableSwitchCh1`	`BOOL`	FALSE	A variable value resulting from the signal of contacts E1 and E2 of the connected manually operated three-position enable switch: • FALSE: Connected switches are open. • TRUE: Connected switches are closed.

Parameter	Data type	Init Value	Meaning
`S_EnableSwitchCh2`	`BOOL`	FALSE	A variable value resulting from the signal of contacts E3 and E4 of the connected manually operated three-position enable switch: • FALSE: Connected switches are open. • TRUE: Connected switches are closed.
`S_AutoReset`	`BOOL`	FALSE	A variable or constant value indicating the state of the auto reset function: • FALSE: Manual reset when emergency stop button is released. • TRUE: Automatic reset when emergency stop button is released **NOTE:** Activate this function only after you have confirmed that no hazard can occur at the start of the system. Use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system or application measures to guard against unexpected (or unintended) start-up. Refer to the hazard message in the Function Description above.
`Reset`	`BOOL`	FALSE	The variable value indicating reset of the state machine, coupled with detected error and status messages as indicated in the `DiagCode` parameter, after the underlying cause of the detected error has been removed. This behavior is designed as a detected error reset. **NOTE:** This function is only active on a signal change from FALSE to TRUE.

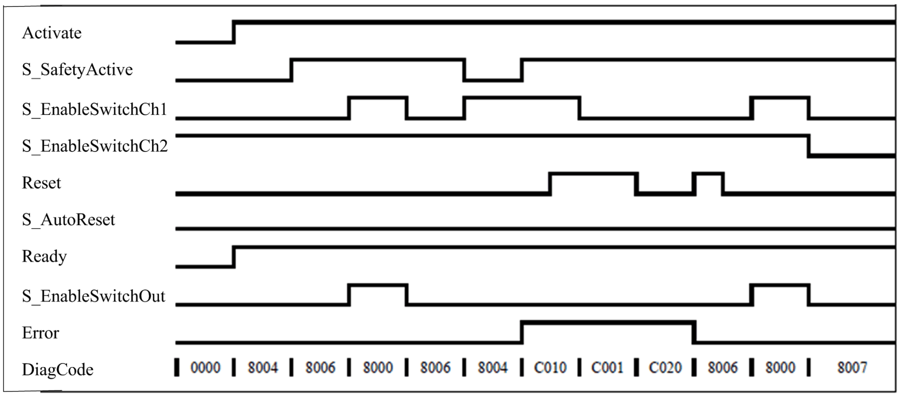
Output Parameters

Parameter	Data type	Init Value	Meaning
`Ready`	`BOOL`	FALSE	• TRUE indicates that the function block is activated and the output results are valid. • FALSE indicates that the function block is not active and the program is not executed. **NOTE:** This parameter can be useful in debug mode or to activate/deactivate additional function blocks, as well as for further processing in the functional program.
`S_EnableSwitchOut`	`BOOL`	FALSE	A safety related output that indicates suspension of monitoring: • FALSE: Disable suspension of the monitoring. • TRUE: Enable suspension of the monitoring.

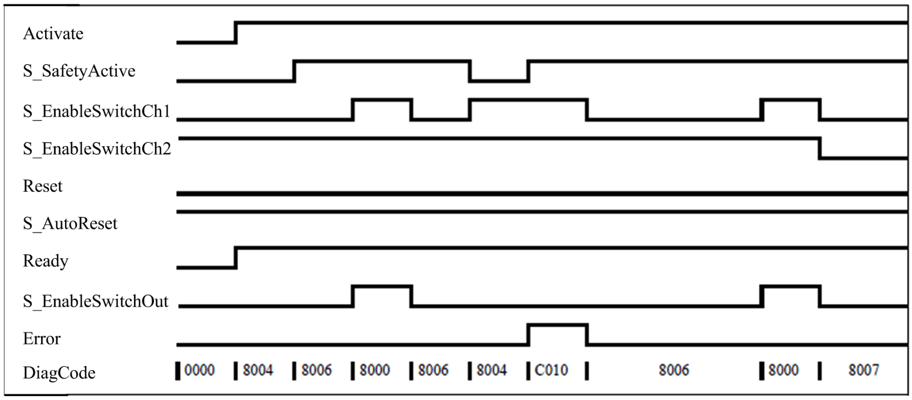
Parameter	Data type	Init Value	Meaning
`Error`	`BOOL`	FALSE	Function block detected error message.
`DiagCode`	`WORD`	0000 hex	Function block diagnostic code.

Typical Timing Diagrams

`S_AutoReset` = FALSE

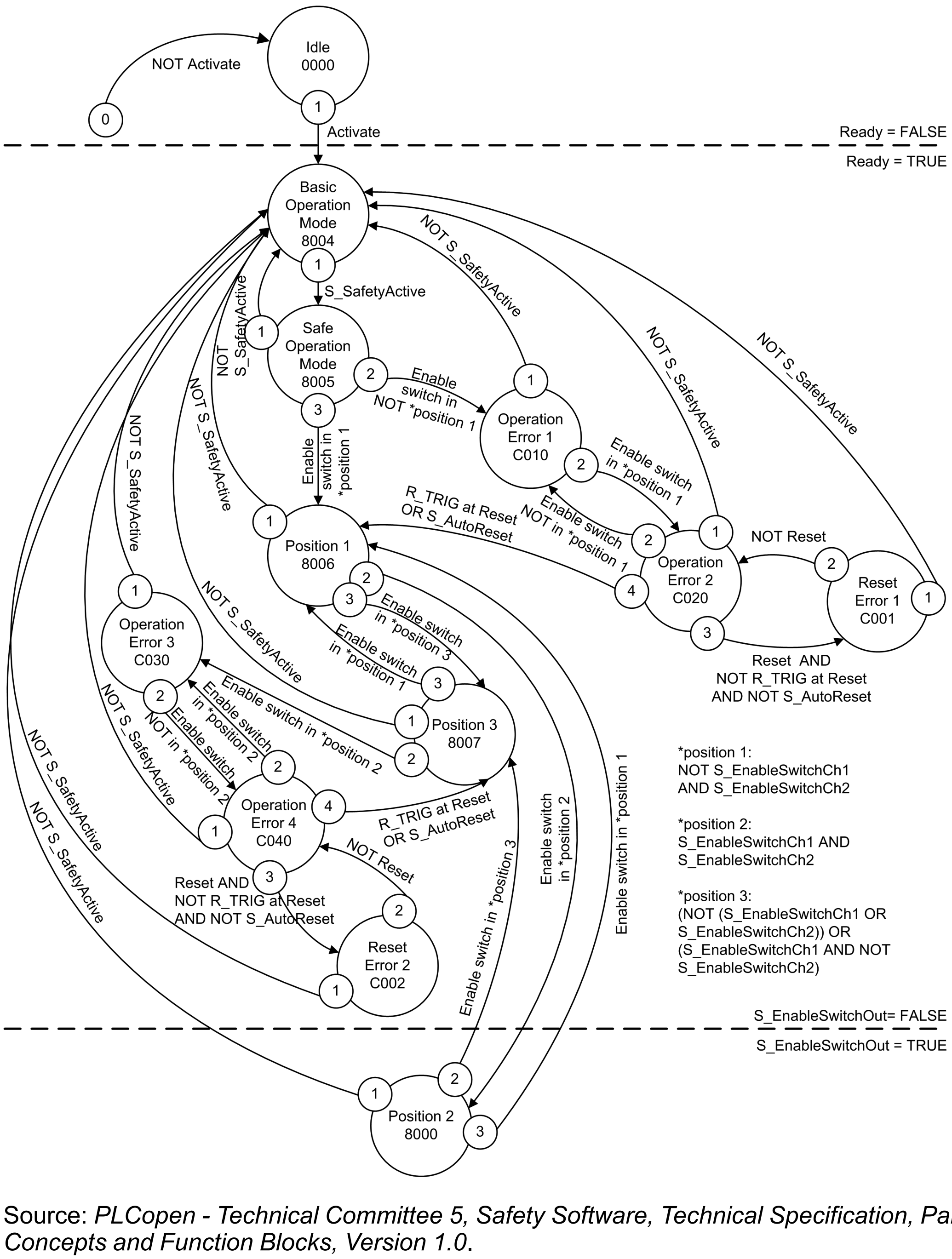


`S_AutoReset` = TRUE



State Diagram

The following diagram describes the state transitions of the `S_ENABLE_SWITCH` function block:



Source: *PLCopen - Technical Committee 5, Safety Software, Technical Specification, Part 1: Concepts and Function Blocks, Version 1.0*.

NOTE: The transition to the Idle state from any other state, occurring because Activate = FALSE, is not depicted. Such a transition has the highest priority (0).

Error Detection

The following conditions force a transition to the detected error state:

- Invalid static `Reset` signal in the process.
- Invalid switch positions.

Detected Error Management

When an error is detected, the outputs are as follows:

- `S_EnableSwitchOut` is set to FALSE and remains in the defined safe state.
- Unlike other function blocks, a `Reset` error state does not need to be reset, but instead can be allowed to persist by the condition `Reset` = FALSE or when the signal `S_SafetyActive` = FALSE.
- After the detected error has been resolved, the `S_EnableSwitchOut` output can be set to TRUE using the enable switch, *provided* that the enable switch is in the initial position specified in the process. If `S_AutoReset` = FALSE, a rising edge is required at `Reset`.

When returning a detected error message, the `DiagCode` parameter can present one of the following detected error values

DiagCode	State Name	State Description and Output Settings
C001	Reset Error 1	Static Reset signal detected in state C020: • `S_EnableSwitchOut` = FALSE • `Error` = TRUE
C002	Reset Error 2	Static Reset signal detected in state C040: • `S_EnableSwitchOut` = FALSE • `Error` = TRUE
C010	Operation Error 1	Enable switch not in position 1 during activation of `S_SafetyActive`: • `S_EnableSwitchOut` = FALSE • `Error` = TRUE
C020	Operation Error 2	Enable switch in position 1 after C010: • `S_EnableSwitchOut` = FALSE • `Error` = TRUE

DiagCode	State Name	State Description and Output Settings
C030	Operation Error 3	Enable switch in position 2 after position 3: • `S_EnableSwitchOut` = FALSE • `Error` = TRUE
C040	Operation Error 4	Enable switch not in position 2 after C030: • `S_EnableSwitchOut` = FALSE • `Error` = TRUE

Diagnostic Codes Management

When returning a status message, the `Error` parameter is set to FALSE, and the `DiagCode` parameter displays one of the following hexadecimal values:

DiagCode	State Name	State Description and Output Settings
0	IDLE	The function block is not active (initial state): • `S_EnableSwitchOut` = FALSE • `Error` = FALSE
8004	Basic Operation Mode	Safe Operation Mode is not active: • `S_EnableSwitchOut` = FALSE • `Error` = FALSE
8005	Safe Operation Mode	Safe Operation Mode is active: • `S_EnableSwitchOut` = FALSE • `Error` = FALSE
8006	Position 1	Safe Operation Mode is active and the enable switch is in position 1: • `S_EnableSwitchOut` = FALSE • `Error` = FALSE
8007	Position 3	Safe Operation Mode is active and the enable switch is in position 3: • `S_EnableSwitchOut` = FALSE • `Error` = FALSE
8000	Position 2	Safe Operation Mode is active and the enable switch is in position 2: • `S_EnableSwitchOut` = TRUE • `Error` = FALSE

S_ESPE: Electro-Sensitive Protective Equipment

What’s in This Chapter

Description 87

Introduction

This chapter describes the `S_ESPE` block.

Description

Function Description

Use the `S_ESPE` function block to support the *electro-sensitive protective equipment* (ESPE) function in an application. This function block can monitor an item of electro-sensitive protective equipment.

⚠ WARNING

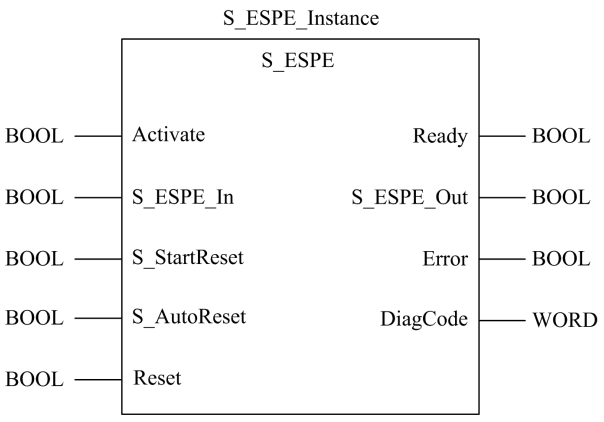
UNINTENDED EQUIPMENT OPERATION

Activate the S_StartReset and S_AutoReset inputs only after you verify that no hazardous situation can occur if the system is started.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Representation in FBD

Representation



Input Parameters

Parameter	Data type	Init Value	Meaning
`Activate`	`BOOL`	FALSE	The variable or constant value indicating activation state of the function block. This parameter can be connected to the variable, which represents the status (Active or Not Active) of the safety-related device. This causes no irrelevant diagnostic information to be generated if a device is disabled: • When FALSE, the output variables are set to the initial values. • Assign a static TRUE signal if no device is connected.
`S_ESPE_In`	`BOOL`	FALSE	A variable or constant input signaling the status of the electro-sensitive protective equipment: • FALSE: ESPE activated, request for safety related response. • TRUE: ESPE not activated, no request for a safety related response. **NOTE:** When the ESPE is used in applications as a trip device, verify that the safety control system is able to detect a very short interruption of the sensor (as specified in IEC61496-1: a minimum of 80 ms).

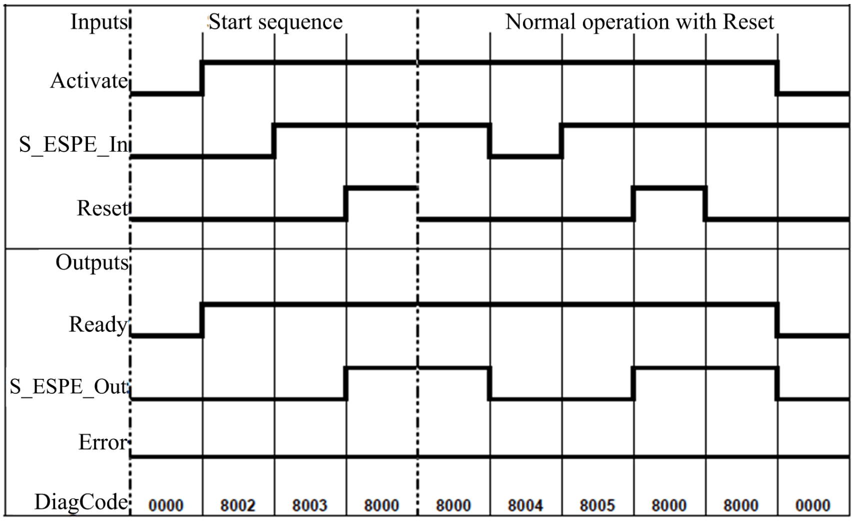
Parameter	Data type	Init Value	Meaning
`S_StartReset`	`BOOL`	FALSE	A variable or constant value that indicates: • FALSE: Manual reset when system is started (warm or cold). • TRUE: Automatic reset when system is started (warm or cold). **NOTE:** Activate this function only after you have confirmed that no hazard can occur at the start of the PES. Use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system or application measures to guard against unexpected (or unintended) start-up. Refer to the hazard message in the Function Description above.
`S_AutoReset`	`BOOL`	FALSE	A variable or constant value indicating the state of the auto reset function: • FALSE: Manual reset when emergency stop button is released. • TRUE: Automatic reset when emergency stop button is released **NOTE:** Activate this function only after you have confirmed that no hazard can occur at the start of the system. Use of the Automatic Circuit Reset feature of the function blocks requires implementation of other system or application measures to guard against unexpected (or unintended) start-up. Refer to the hazard message in the Function Description above.
`Reset`	`BOOL`	FALSE	The variable value indicating reset of the state machine, coupled with detected error and status messages as indicated in the `DiagCode` parameter, after the underlying cause of the detected error has been removed. This behavior is designed as a detected error reset. **NOTE:** This function is only active on a signal change from FALSE to TRUE.

Output Parameters

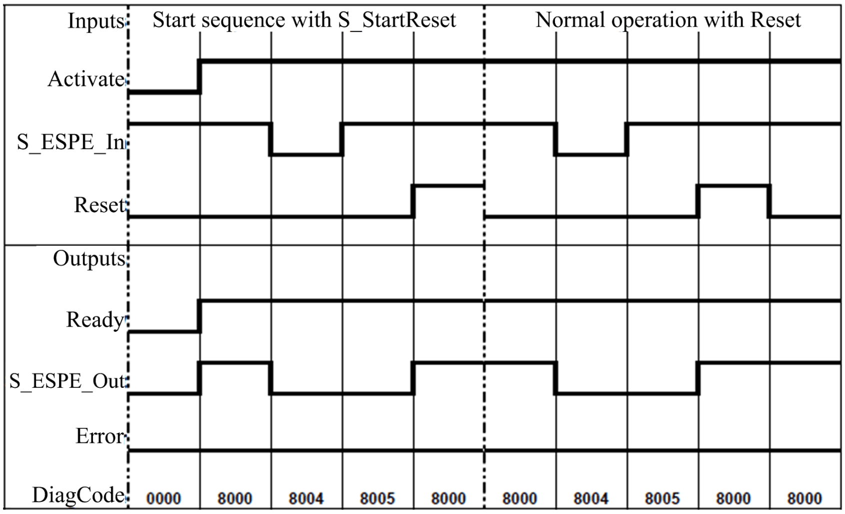
Parameter	Data type	Init Value	Meaning
Ready	BOOL	FALSE	• TRUE indicates that the function block is activated and the output results are valid. • FALSE indicates that the function block is not active and the program is not executed. **NOTE:** This parameter can be useful in debug mode or to activate/deactivate additional function blocks, as well as for further processing in the functional program.
S_ESPE_Out	BOOL	FALSE	A safety-related output that indicates: • FALSE: Safety-related output disabled. • TRUE: Safety-related output enabled.
Error	BOOL	FALSE	Function block detected error message.
DiagCode	WORD	0000 hex	Function block diagnostic code.

Typical Timing Diagrams

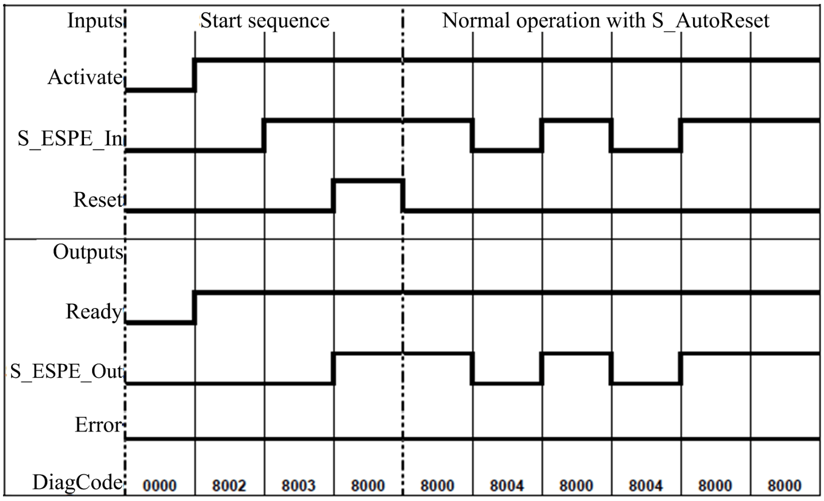
S_StartReset = FALSE; S_AutoReset = FALSE



`S_StartReset` = TRUE; `S_AutoReset` = FALSE

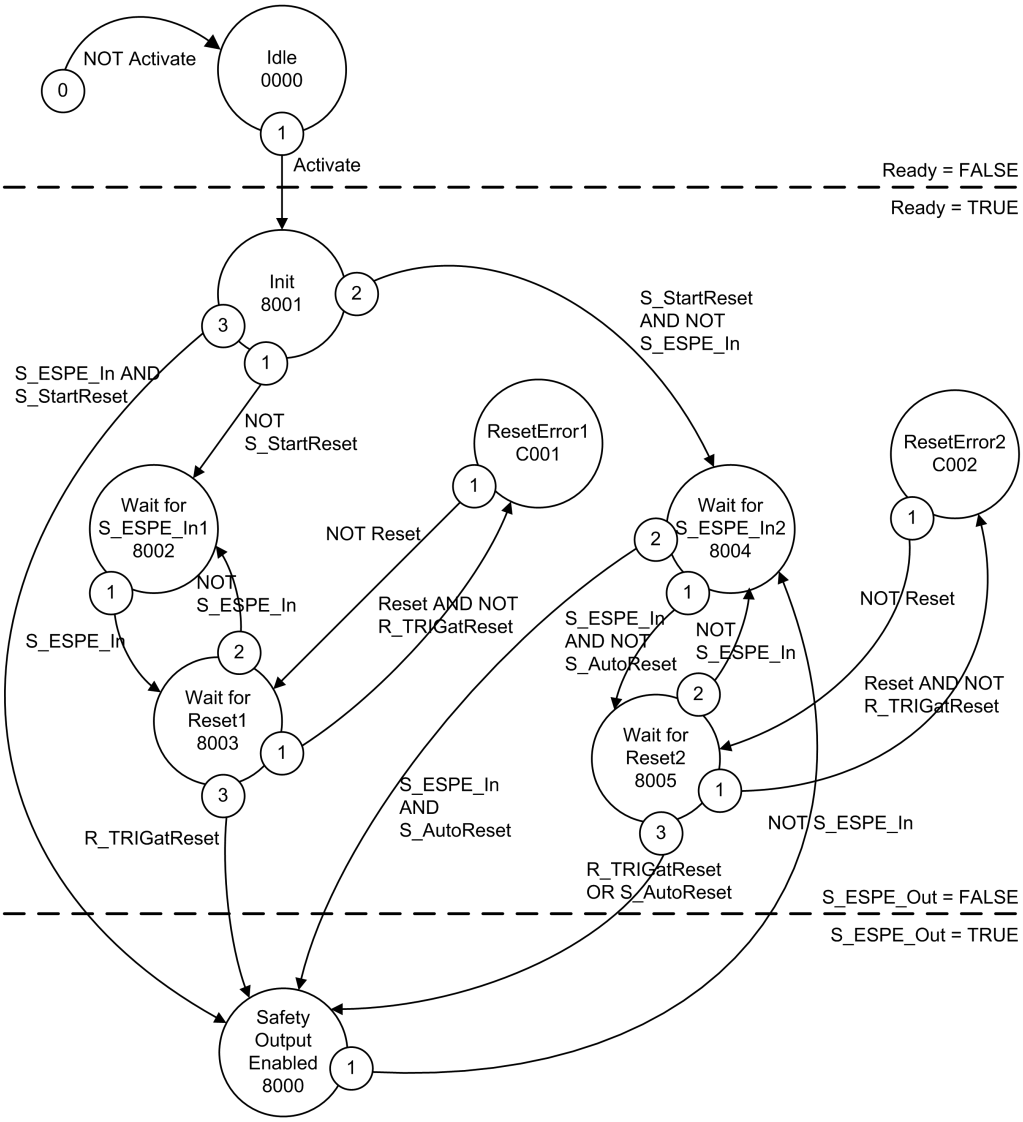


`S_StartReset` = FALSE; `S_AutoReset` = TRUE



State Diagram

The following diagram describes the state transitions of the `S_ESPE` function block:



Source: *PLCopen - Technical Committee 5, Safety Software, Technical Specification, Part 1: Concepts and Function Blocks, Version 1.0*.

NOTE: The transition to the Idle state from any other state, occurring because Activate = FALSE, is not depicted. Such a transition has the highest priority (0).

Error Detection

The function block detects a static TRUE signal at `Reset` input.

Detected Error Management

`S_ESPE_Out` is set to an initial value of FALSE.

If a static TRUE signal is received at the `Reset` input, the `DiagCode` output indicates the detected error code and the `Error` output is set to TRUE.

To leave the detected error state, set `Reset` input to FALSE.

When returning a detected error message, the `DiagCode` parameter can present one of the following detected error values

DiagCode	State Name	State Description and Output Settings
C001	Reset Error 1	`Reset` signal is TRUE while waiting for S_ESPE_In = TRUE: • `S_ESPE_Out` = FALSE • `Error` = TRUE
C002	Reset Error 2	`Reset` signal is TRUE while waiting for S_ESPE_In = TRUE: • `S_ESPE_Out` = FALSE • `Error` = TRUE

Diagnostic Codes Management

When returning a status message, the `Error` parameter is set to FALSE, and the `DiagCode` parameter displays one of the following hexadecimal values:

DiagCode	State Name	State Description and Output Settings
0	IDLE	The function block is not active (initial state): • `S_ESPOut` = FALSE • `Error` = FALSE
8001	INIT	`Activate` is TRUE. The function block was enabled. Check if `S_StartReset` needs to be set: • `S_ESPOut` = FALSE • `Error` = FALSE

DiagCode	State Name	State Description and Output Settings
8002	Wait for S_ESPE_In 1	Activate is TRUE. Check if Reset is FALSE and wait for S_ESPE_In = TRUE: • S_ESPOut = FALSE • Error = FALSE
8003	Wait for Reset 1	Activate is TRUE. Wait for rising edge of Reset: • S_ESPOut = FALSE • Error = FALSE
8004	Wait for S_ESPE_In 2	Activate is TRUE. Safety-related request detected. Check if Reset is FALSE and wait for S_ESPE_In = TRUE: • S_ESPOut = FALSE • Error = FALSE
8005	Wait for Reset 2	Activate is TRUE. S_ESPE_In = TRUE. Check for S_AutoReset or wait for rising edge of Reset: • S_ESPOut = FALSE • Error = FALSE
8000	Safety Output Enabled	Activate is TRUE. S_ESPE_In = TRUE. Functional mode with S_ESPE_Out = TRUE: • S_ESPOut = TRUE • Error = FALSE

S_GUARD_LOCKING: Guard Lock Control

What's in This Chapter

Description 95

Introduction

This chapter describes the `S_GUARD_LOCKING` block.

Description

Function Description

Use the `S_GUARD_LOCKING` function with a mechanical guard lock switch that restricts access to a secured, hazardous area. The function controls the guard lock and monitors the position of the guard and the lock.

When a request is made to gain access to the secured hazardous area, the guard should be unlocked only if the hazardous area is in a pre-defined safe state.

The guard can be locked if the guard is closed. The secured equipment should be started only when the guard is closed and the guard is locked. The function detects an opened guard or unlocked guard.

⚠WARNING

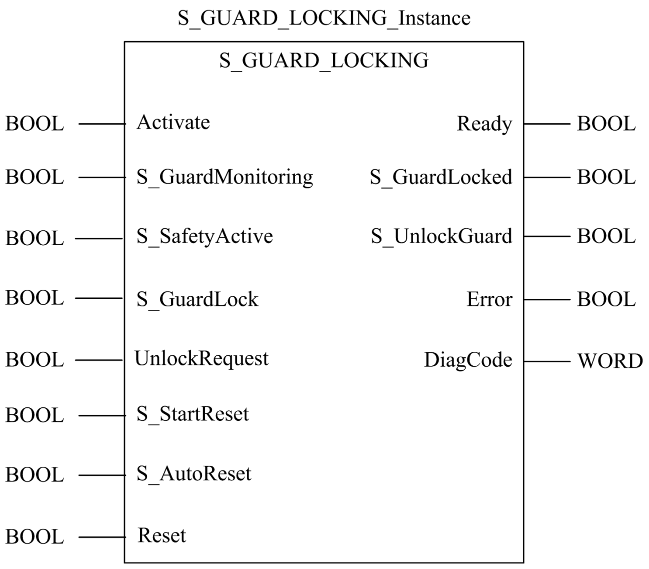
UNINTENDED EQUIPMENT OPERATION

Activate the S_StartReset and S_AutoReset inputs only after you verify that no hazardous situation can occur if the system is started.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Representation in FBD

Representation



Input Parameters

Parameter	Data type	Init Value	Meaning
`Activate`	`BOOL`	FALSE	The variable or constant value indicating activation state of the function block. This parameter can be connected to the variable, which represents the status (Active or Not Active) of the safety-related device. This causes no irrelevant diagnostic information to be generated if a device is disabled: • When FALSE, the output variables are set to the initial values. • Assign a static TRUE signal if no device is connected.
`S_ GuardMonitoring`	`BOOL`	FALSE	A variable input signal monitoring the guard interlocking: • FALSE: Guard open. • TRUE: Guard closed.

Parameter	Data type	Init Value	Meaning
`S_SafetyActive`	`BOOL`	FALSE	A variable input signal indicating the status of the hazardous area based on speed monitoring or safe time off delay: • FALSE: Equipment is not in the defined safe state. • TRUE: Equipment is in the defined state.
`S_GuardLock`	`BOOL`	FALSE	A variable input signal indicating the status of the mechanical guard locking: • FALSE: Guard is not locked. • TRUE: Guard is locked.
`UnlockRequest`	`BOOL`	FALSE	A variable input signaling operator intervention, i.e., the request to unlock the mechanical guard lock: • FALSE: No request made. • TRUE: A request has been made.
`S_StartReset`	`BOOL`	FALSE	A variable or constant value that indicates: • FALSE: Manual reset when system is started (warm or cold). • TRUE: Automatic reset when system is started (warm or cold). **NOTE:** Activate this function only after you have confirmed that no hazard can occur at the start of the controller. Use of an automatic reset feature of the function blocks requires implementation of other system or application measures to guard against unexpected (or unintended) start-up. Refer to the hazard message in the Function Description above.

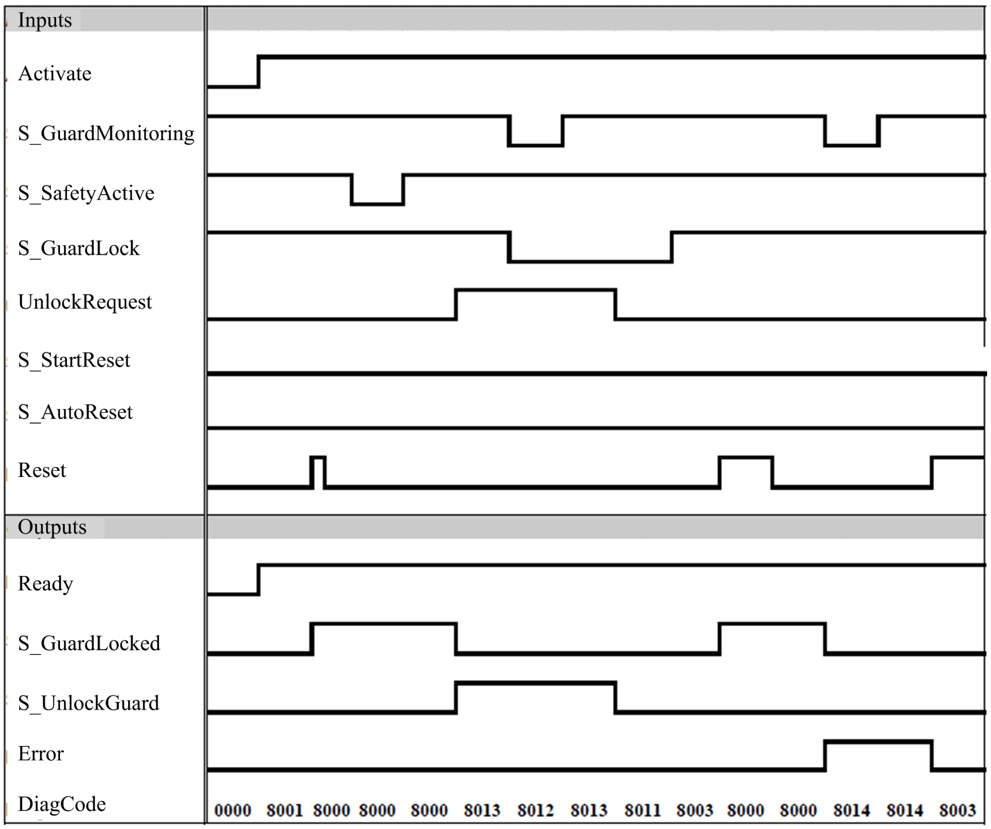
Parameter	Data type	Init Value	Meaning
`S_AutoReset`	`BOOL`	FALSE	A variable or constant value indicating the state of the auto reset function: • FALSE: Manual reset when emergency stop button is released. • TRUE: Automatic reset when emergency stop button is released **NOTE:** Activate this function only after you have confirmed that no hazard can occur at the start of the controller. Use of an automatic reset feature of the function blocks requires implementation of other system or application measures to guard against unexpected (or unintended) start-up. Refer to the hazard message in the Function Description above.
`Reset`	`BOOL`	FALSE	The variable value indicating reset of the state machine, coupled with detected error and status messages as indicated in the `DiagCode` parameter, after the underlying cause of the detected error has been removed. This behavior is designed as a detected error reset. **NOTE:** This function is only active on a signal change from FALSE to TRUE.

Output Parameters

Parameter	Data type	Init Value	Meaning
`Ready`	`BOOL`	FALSE	• TRUE indicates that the function block is activated and the output results are valid. • FALSE indicates that the function block is not active and the program is not executed. **NOTE:** This parameter can be useful in debug mode or to activate/deactivate additional function blocks, as well as for further processing in the functional program.
`S_GuardLocked`	`BOOL`	FALSE	State of the mechanical guard lock switch that restricts access to a secured area: • FALSE: Not in the defined safe. • TRUE: Defined safe state.
`S_UnlockGuard`	`BOOL`	FALSE	A signal to unlock the guard: • FALSE: Locked guard. • TRUE: Unlock the guard.
`Error`	`BOOL`	FALSE	Function block detected error message.
`DiagCode`	`WORD`	0000 hex	Function block diagnostic code.

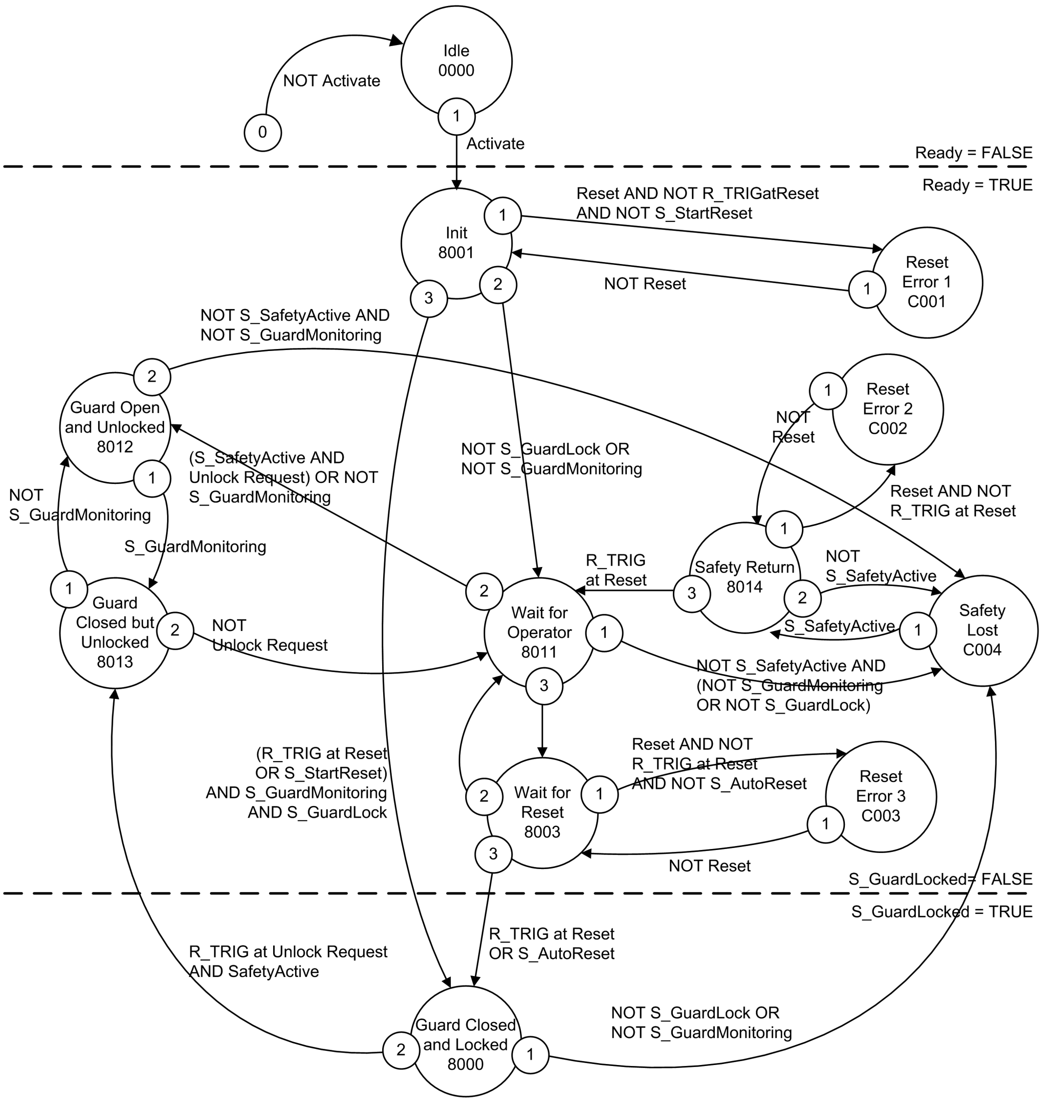
Typical Timing Diagrams

S_GUARD_LOCKING



Safety Diagram

The following diagram describes the state transitions of the `S_GUARD_LOCKING` function block:



Source: *PLCopen - Technical Committee 5, Safety Software, Technical Specification, Part 1: Concepts and Function Blocks, Version 1.0*.

NOTE: The transition to the Idle state from any other state, occurring because Activate = FALSE, is not depicted. Such a transition has the highest priority (0).

Error Detection

The function block detects a static TRUE signal at `Reset` input. Errors are detected at the guard switches.

Detected Error Management

In the event an error is detected:

- The `S_GuardLocked` and `S_UnlockGuard` outputs are set to FALSE.
- The `DiagCode` output indicates the detected error code.
- The `Error` output is set to TRUE.

Detected errors are acknowledged by a rising edge at the `Reset` input.

When presenting a detected error message, the `DiagCode` parameter can present one of the following detected error values

DiagCode	State Name	State Description and Output Settings
C001	Reset Error 1	Static `Reset` detected in state 8001: • `S_GuardLocked` = FALSE • `S_UnlockGuard` = FALSE • `Error` = TRUE
C002	Reset Error 2	Static `Reset` detected in state C004: • `S_GuardLocked` = FALSE • `S_UnlockGuard` = FALSE • `Error` = TRUE
C003	Reset Error 3	Static `Reset` detected in state 8011: • `S_GuardLocked` = FALSE • `S_UnlockGuard` = FALSE • `Error` = TRUE
C004	Safety Lost	Guard opened or guard unlocked: • `S_GuardLocked` = FALSE • `S_UnlockGuard` = FALSE • `Error` = TRUE

Diagnostic Codes Management

When returning a status message, the `Error` parameter is set to FALSE, and the `DiagCode` parameter displays one of the following hexadecimal values:

DiagCode	State Name	State Description and Output Settings
0	IDLE	The function block is not active (initial state): • `S_GuardLocked` = FALSE • `S_UnlockGuard` = FALSE • `Error` = FALSE
8000	Guard Closed and Locked	Guard is locked: • `S_GuardLocked` = TRUE • `S_UnlockGuard` = FALSE • `Error` = FALSE
8001	INIT	`Activate` is TRUE. The function block was enabled: • `S_GuardLocked` = FALSE • `S_UnlockGuard` = FALSE • `Error` = FALSE
8003	Wait for Reset 1	Door is closed and locked, waiting for operator `Reset`: • `S_GuardLocked` = FALSE • `S_UnlockGuard` = FALSE • `Error` = FALSE
8011	Wait for Operator	Waiting for operator to either unlock request or reset: • `S_GuardLocked` = FALSE • `S_UnlockGuard` = FALSE • `Error` = FALSE
8012	Guard Open and Unlocked	Lock is released and guard is open: • `S_GuardLocked` = FALSE • `S_UnlockGuard` = TRUE • `Error` = FALSE
8013	Guard Closed but Unlocked	Lock is released but guard is closed: • `S_GuardLocked` = FALSE • `S_UnlockGuard` = TRUE • `Error` = FALSE
8014	Safety Return	Return of `S_SafetyActive` signal, now waiting for operator acknowledge: • `S_GuardLocked` = FALSE • `S_UnlockGuard` = FALSE • `Error` = FALSE

S_GUARD_MONITORING: Guard Lock Monitoring

What's in This Chapter

Description 103

Introduction

This chapter describes the `S_GUARD_MONITORING` block.

Description

Function Description

Use the `S_GUARD_MONITORING` function block to monitor the status of a guard lock with two-state interlocking in your application. This block performs only monitoring, without guard locking.

⚠WARNING

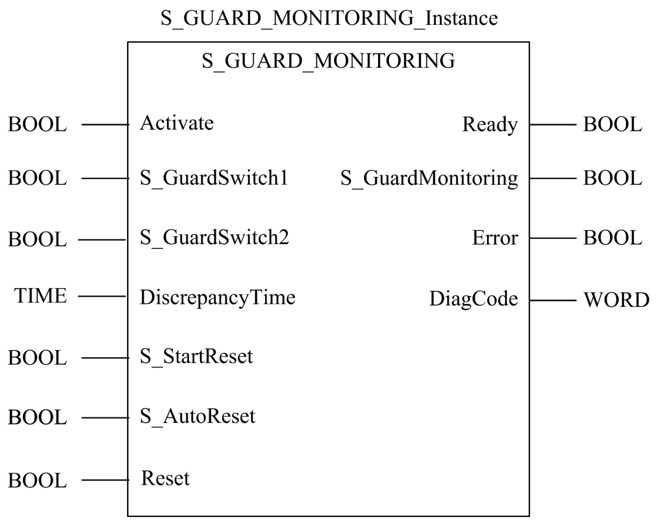
UNINTENDED EQUIPMENT OPERATION

Activate the S_StartReset and S_AutoReset inputs only after you verify that no hazardous situation can occur if the system is started.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Representation in FBD

Representation



Input Parameters

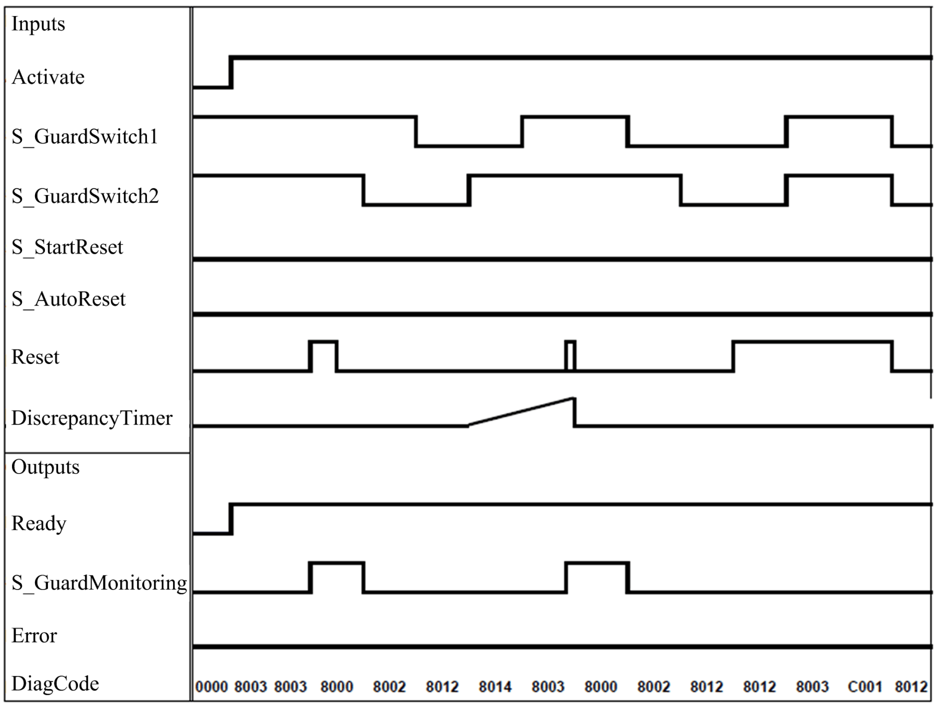
Parameter	Data type	Init Value	Meaning
`Activate`	`BOOL`	FALSE	The variable or constant value indicating activation state of the function block. This parameter can be connected to the variable, which represents the status (Active or Not Active) of the safety-related device. This causes no irrelevant diagnostic information to be generated if a device is disabled: • When FALSE, the output variables are set to the initial values. • Assign a static TRUE signal if no device is connected.
`S_GuardSwitch1`	`BOOL`	FALSE	A variable input signal monitoring the status of guard switch 1: • FALSE: Guard open. • TRUE: Guard closed.
`S_GuardSwitch2`	`BOOL`	FALSE	A variable input signal monitoring the status of guard switch 2: • FALSE: Guard open. • TRUE: Guard closed.

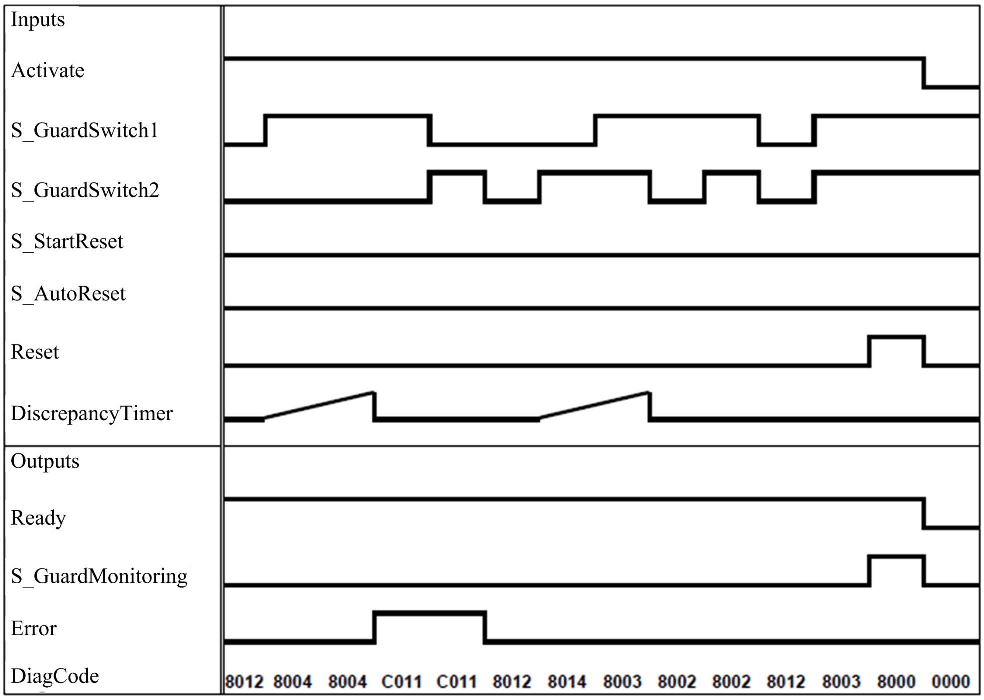
Parameter	Data type	Init Value	Meaning
`DiscrepancyTime`	`TIME`	T#0 ms	A configurable constant value for the monitored synchronous time between `S_GuardSwitch1` and `S_GuardSwitch2`.
`S_StartReset`	`BOOL`	FALSE	A variable or constant value that indicates: • FALSE: Manual reset when system is started (warm or cold). • TRUE: Automatic reset when system is started (warm or cold). **NOTE:** Activate this function only after you have confirmed that no hazard can occur at the start of the controller. Use of an automatic reset feature of the function blocks requires implementation of other system or application measures to guard against unexpected (or unintended) start-up. Refer to the hazard message in the Function Description above.
`S_AutoReset`	`BOOL`	FALSE	A variable or constant value indicating the state of the auto reset function: • FALSE: Manual reset when emergency stop button is released. • TRUE: Automatic reset when emergency stop button is released **NOTE:** Activate this function only after you have confirmed that no hazard can occur at the start of the controller. Use of an automatic reset feature of the function blocks requires implementation of other system or application measures to guard against unexpected (or unintended) start-up. Refer to the hazard message in the Function Description above.
`Reset`	`BOOL`	FALSE	The variable value indicating reset of the state machine, coupled with detected error and status messages as indicated in the `DiagCode` parameter, after the underlying cause of the detected error has been removed. This behavior is designed as a detected error reset. **NOTE:** This function is only active on a signal change from FALSE to TRUE.

Output Parameters

Parameter	Data type	Init Value	Meaning
`Ready`	`BOOL`	FALSE	• TRUE indicates that the function block is activated and the output results are valid. • FALSE indicates that the function block is not active and the program is not executed. **NOTE:** This parameter can be useful in debug mode or to activate/deactivate additional function blocks, as well as for further processing in the functional program.
`S_GuardMonitoring`	`BOOL`	FALSE	State of the interlocking guard: • FALSE: Guard is not active. • TRUE: Guard is active: ◦ `S_GuardSwitch1` and `S_GuardSwitch2` are TRUE, no detected error and acknowledgment. Guard is active ◦ `Error` parameter is FALSE
`Error`	`BOOL`	FALSE	Function block detected error message.
`DiagCode`	`WORD`	0000 hex	Function block diagnostic code.

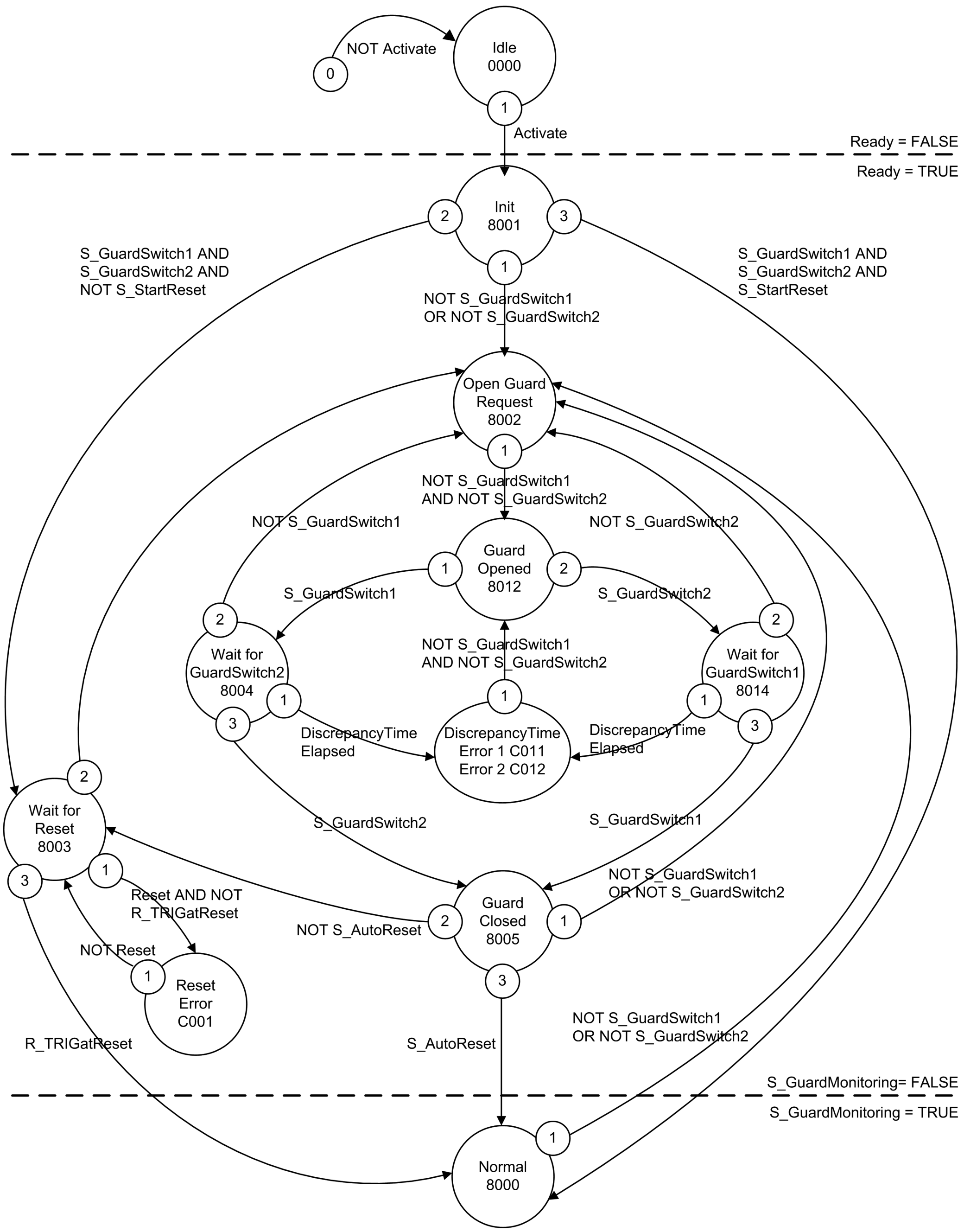
Typical Timing Diagrams




Inputs
Activate
S_GuardSwitch1
S_GuardSwitch2
S_StartReset
S_AutoReset
Reset
DiscrepancyTimer
Outputs
Ready
S_GuardMonitoring
Error
DiagCode
8012 8004 8004 C011 C011 8012 8014 8003 8002 8002 8012 8003 8000 0000

State Diagram

The following diagram describes the state transitions of the `S_GUARD_MONITORING` function block:



Source: *PLCopen - Technical Committee 5, Safety Software, Technical Specification, Part 1: Concepts and Function Blocks, Version 1.0*.

Detected Error Management

In the event an error is detected:

- The `S_GuardMonitoring` output is set to FALSE.
- The `DiagCode` output indicates the relevant error code.
- The `Error` output is set to TRUE.

To leave a detected error state, take the following steps:

- To leave the Reset Error state, set the `Reset` input to FALSE.
- To leave a Discrepancy Time Error state, set both the `S_GuardSwitch1` and `S_GuardSwitch2` inputs to FALSE.

If the `S_GuardSwitch1` and `S_GuardSwitch2` inputs are bridged, no error is detected.

When returning a detected error message, the `DiagCode` parameter can present one of the following detected error values

DiagCode	State Name	State Description and Output Settings
C001	Reset Error	Static `Reset` detected in state 8003: • `S_GuardMonitoring` = FALSE • `Error` = TRUE
C011	Discrepancy Time Error 1	Static `DiscrepancyTimer` elapsed in state 8004: • `S_GuardMonitoring` = FALSE • `Error` = TRUE
C012	Discrepancy Time Error 2	Static `DiscrepancyTimer` elapsed in state 8014: • `S_GuardMonitoring` = FALSE • `Error` = TRUE

Diagnostic Codes Management

When returning a status message, the `Error` parameter is set to FALSE, and the `DiagCode` parameter displays one of the following hexadecimal values:

DiagCode	State Name	State Description and Output Settings
0	IDLE	The function block is not active (initial state): • `S_GuardMonitoring` = FALSE • `Error` = FALSE
8000	Normal	Guard closed and defined safe state acknowledged: • `S_GuardMonitoring` = TRUE • `Error` = FALSE
8001	INIT	`Activate` is TRUE. The function block was enabled: • `S_GuardMonitoring` = FALSE • `Error` = FALSE
8002	Open Guard Request	Complete switching sequence required: • `S_GuardMonitoring` = FALSE • `Error` = FALSE
8003	Wait for Reset	Waiting for rising edge at `Reset`: • `S_GuardMonitoring` = FALSE • `Error` = FALSE
8012	Guard Opened	Guard completely opened: • `S_GuardMonitoring` = FALSE • `Error` = FALSE
8004	Wait for GuardSwitch2	S_GuardWitch1 has been switched to TRUE; waiting for S_ GuardSwitch2; DiscrepancyTimer started: • `S_GuardMonitoring` = FALSE • `Error` = FALSE
8014	Wait for GuardSwitch1	S_GuardWitch2 has been switched to TRUE; waiting for S_ GuardSwitch1; DiscrepancyTimer started: • `S_GuardMonitoring` = FALSE • `Error` = FALSE
8005	Guard Closed	Guard closed; waiting for `Reset` if `S_AutoReset` = FALSE: • `S_GuardMonitoring` = FALSE • `Error` = FALSE

S_MODE_SELECTOR: Safety Mode Switch

What’s in This Chapter

Description 113

Introduction

This chapter describes the `S_MODE_SELECTOR` block.

Description

Function Description

Use the `S_MODE_SELECTOR` function block to support the function of a mode selector switch with up to eight signals in an application.

Connect a mode selector switch to the `S_Mode0` to `S_Mode7` inputs of the function block to specify an operating mode of the application, by issuing a single TRUE input signal from the mode selector switch.

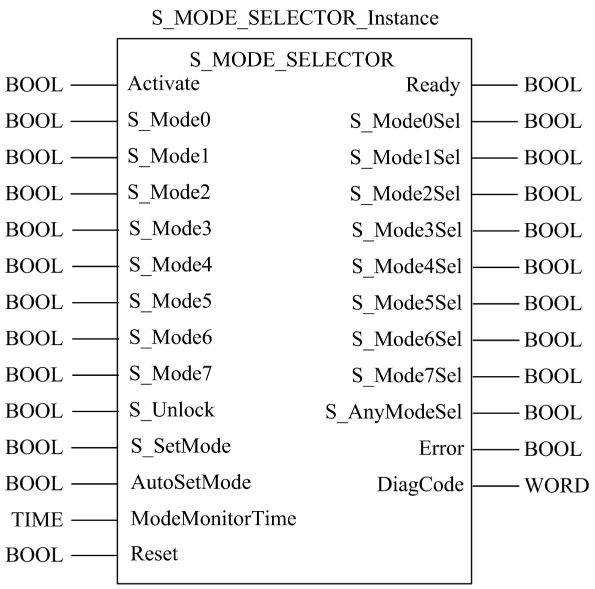
Configure your application to execute the specified operating mode (for example, service mode, cleaning mode, jogging mode, setup mode or automatic mode).

Depending upon the mechanical properties of the mode selector switch, it is possible that more than one signal – or no signal – is set to TRUE when you change the mode selector switch setting. In this case, configure the `ModeMonitorTime` parameter to create a switching time span during which these conditions are permitted. Outside this permitted time span, the function block detects these conditions as errors.

The signal states `S_Mode0` to `S_Mode7` are output at the corresponding `S_Mode0Sel` to `S_Mode7Sel` output parameters, either automatically or after a manually issued operator acknowledgment signal.

Representation in FBD

Representation



Input Parameters

Parameter	Data type	Init Value	Meaning
`Activate`	`BOOL`	FALSE	The variable or constant value indicating activation state of the function block. This parameter can be connected to the variable, which represents the status (Active or Not Active) of the safety-related device. This causes no irrelevant diagnostic information to be generated if a device is disabled: • When FALSE, the output variables are set to the initial values. • Assign a static TRUE signal if no device is connected.

`S_Mode0`
...
`S_Mode7`
	`BOOL`	FALSE	A variable or constant value for the input assigned to a specific position, 0...7, on the mode selector switch:
			• FALSE: The associated mode (0...7) is not selected.
			• TRUE: the associated mode (0...7) is selected.

Parameter	Data type	Init Value	Meaning
S_Unlock	BOOL	FALSE	A variable or constant input signal that locks and unlocks the selected mode: • FALSE: The active S_Mode*X*Sel output is locked. A change to any other S_Mode*X* input does not lead to a change in the active S_Mode*X*Sel output even in the event of a rising edge of S_SetMode. • TRUE: The active S_Mode*X*Sel output is not locked. A change to a different S_Mode*X*Sel output is possible.
S_SetMode	BOOL	FALSE	A variable that, when set to TRUE by the operator, acknowledges the change in the selected mode. **NOTE:** When a mode change occurs by setting an S_Mode*X* to TRUE, both S_AnyModeSel and the associated S_Mode*X*Sel are set to FALSE. Only a rising edge of SetMode causes the associated S_Mode*X*Sel to be set to TRUE.
AutoSetMode	BOOL	FALSE	A constant value that sets the acknowledgment mode: • FALSE: The operator is required to manually acknowledge a change in mode via S_SetMode. • TRUE: A valid change of the S_Mode*X* input to a different S_Mode*X* selection automatically leads to a change in S_Mode*X*Sel without operator acknowledgment via SetMode (provided that the S_Mode*X*Sel setting is not locked via S_Unlock).
ModeMonitorTime	TIME	FALSE	A constant value indicating the maximum time for changing the selection input.
Reset	BOOL	FALSE	The variable value indicating reset of the state machine, coupled with detected error and status messages as indicated in the DiagCode parameter, after the underlying cause of the detected error has been removed. This behavior is designed as a detected error reset. **NOTE:** This function is only active on a signal change from FALSE to TRUE.

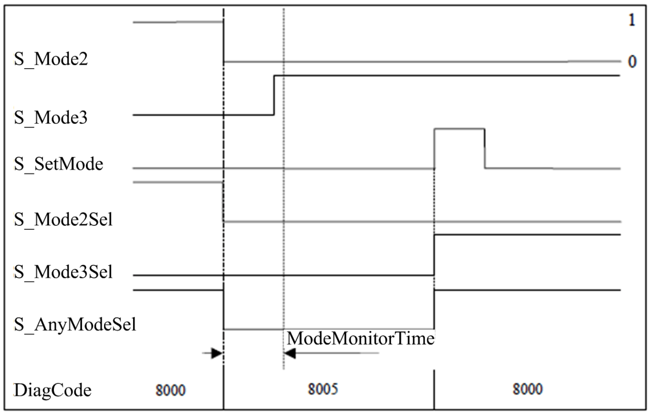
Output Parameters

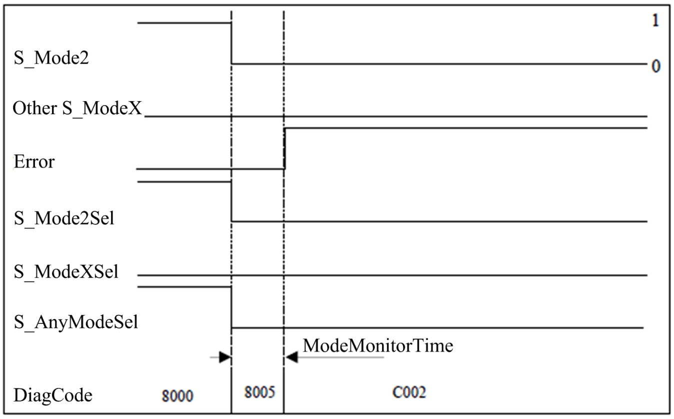
Parameter	Data type	Init Value	Meaning
`Ready`	`BOOL`	FALSE	• TRUE indicates that the function block is activated and the output results are valid. • FALSE indicates that the function block is not active and the program is not executed. **NOTE:** This parameter can be useful in debug mode or to activate/deactivate additional function blocks, as well as for further processing in the functional program.

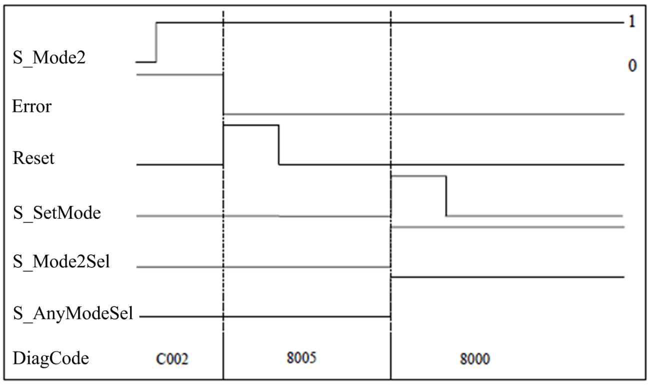
`S_Mode0`
...
`S_Mode7`
	`BOOL`	FALSE	Indicates the selected and acknowledged status of the specified mode (0...7):
			• FALSE: The specified mode (0...7) is not selected or is not active.
			• TRUE: The specified mode (0...7) is selected and active

`S_AnyModeSel`	`BOOL`	FALSE	Indicates if any mode (`S_Mode0`...`S_Mode7`) is selected: • FALSE: No `S_ModeX` is selected. • TRUE: One of the eight modes (`S_Mode0`...`S_Mode7`) is selected.
`Error`	`BOOL`	FALSE	Function block detected error message.
`DiagCode`	`WORD`	0000 hex	Function block diagnostic code.

Typical Timing Diagrams

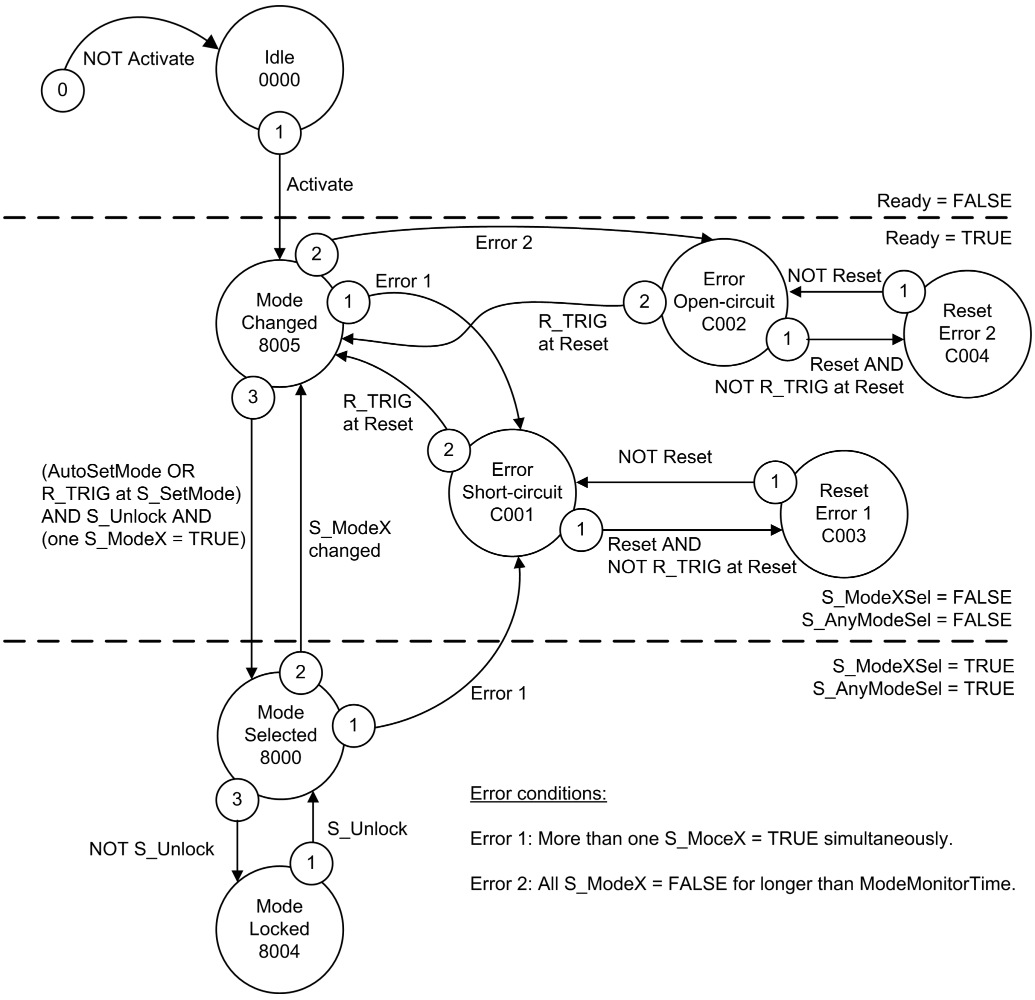



1
0
S_Mode2
Other S_ModeX
Error
S_Mode2Sel
S_ModeXSel
S_AnyModeSel
ModeMonitorTime
DiagCode
8000
8005
C002


1
0
S_Mode2
Error
Reset
S_SetMode
S_Mode2Sel
S_AnyModeSel
DiagCode
C002
8005
8000

State Diagram

The following diagram describes the state transitions of the `S_MODE_SELECTOR` function block:



Source: *PLCopen - Technical Committee 5, Safety Software, Technical Specification, Part 1: Concepts and Function Blocks, Version 1.0*.

NOTE: The transition to the Idle state from any other state, occurring because Activate = FALSE, is not depicted. Such a transition has the highest priority (0).

Error Detection

The `S_MODE_SELECTOR` function block:

- Detects if no input mode is selected. This invalid condition is raised after the `ModeMonitorTime` has elapsed.
- Restarts with every falling edge of an `S_ModeX` input.
- Enters the ModeChanged state on activation.

By contrast, the function block detects whether more than one `S_ModeX` input is simultaneously selected. A static Reset condition is detected when the function block is in either C001 or C002 error state.

Detected Error Management

If an error is detected:

- The `S_ModeXSel` and `S_AnyModeSel` outputs are set to their defined safe state (FALSE).
- The `DiagCode` output indicates the detected error code and the `Error` output is set to TRUE.

Each detected error is acknowledged by means of the rising edge of the `Reset` input. The function block changes from a detected error state to the `ModeChanged` state.

When returning a detected error message, the `DiagCode` parameter can present one of the following detected error values

DiagCode	**State Name**	**State Description and Output Settings**
C001	Error Short-circuit	The function block detected that two or more `S_ModeX` are set to TRUE, indicating a possible cable short-circuit: • `S_AnyModeSel` = FALSE • All `S_ModeXSel` = FALSE • `Error` = TRUE
C002	Error Open-circuit	The function block detected that all `S_ModeX` are FALSE. The period following a falling edge of `S_ModeX` exceeds `ModeMonitorTime`, indicating a possible open-circuit of cables: • `S_AnyModeSel` = FALSE • All `S_ModeXSel` = FALSE • `Error` = TRUE

DiagCode	State Name	State Description and Output Settings
C003	Reset Error 1	Static `Reset` signal detected in state C001: • `S_AnyModeSel` = FALSE • All `S_Mode`*`X`*`Sel` = FALSE • `Error` = TRUE
C004	Reset Error 2	Static `Reset` signal detected in state C002: • `S_AnyModeSel` = FALSE • All `S_Mode`*`X`*`Sel` = FALSE • `Error` = TRUE

Diagnostic Codes Management

When returning a status message, the `Error` parameter is set to FALSE, and the `DiagCode` parameter displays one of the following hexadecimal values:

DiagCode	State Name	State Description and Output Settings
0	IDLE	The function block is not active (initial state): • `S_AnyModeSel` = FALSE • `S_Mode`*`X`*`Sel` = FALSE • `Error` = FALSE
8005	ModeChanged	State after activation or when S_ModeX has changed (unless locked) or after Reset of an error state: • `S_AnyModeSel` = FALSE • `S_Mode`*`X`*`Sel` = FALSE • `Error` = FALSE
8000	ModeSelected	A valid mode is selected, but not locked: • `S_AnyModeSel` = TRUE • The selected `S_Mode`*`X`*`Sel`= TRUE, all others are FALSE. • `Error` = FALSE
8004	ModeLocked	A valid mode is selected and locked. • `S_AnyModeSel` = TRUE • The selected `S_Mode`*`X`*`Sel` = TRUE, all others are FALSE. • `Error` = FALSE

S_OUTCONTROL: Output Driver

What's in This Chapter

Description 121

Introduction

This chapter describes the `S_OUTCONTROL` block.

Description

Function Description

The `S_OUTCONTROL` function block is an output driver for a safety-related output. Use the `S_OUTCONTROL` function block to control the output based on signals from both the functional application (`ProcessControl`, to control the process) and the safety-related application (`S_SafeControl`, to control the safety function).

⚠WARNING

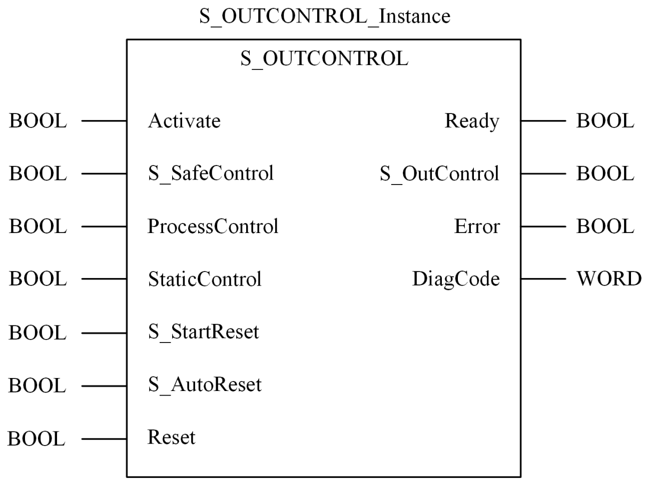
UNINTENDED EQUIPMENT OPERATION

Activate the S_StartReset and S_AutoReset inputs only after you verify that no hazardous situation can occur if the system is started.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Representation in FBD

Representation



Input Parameters

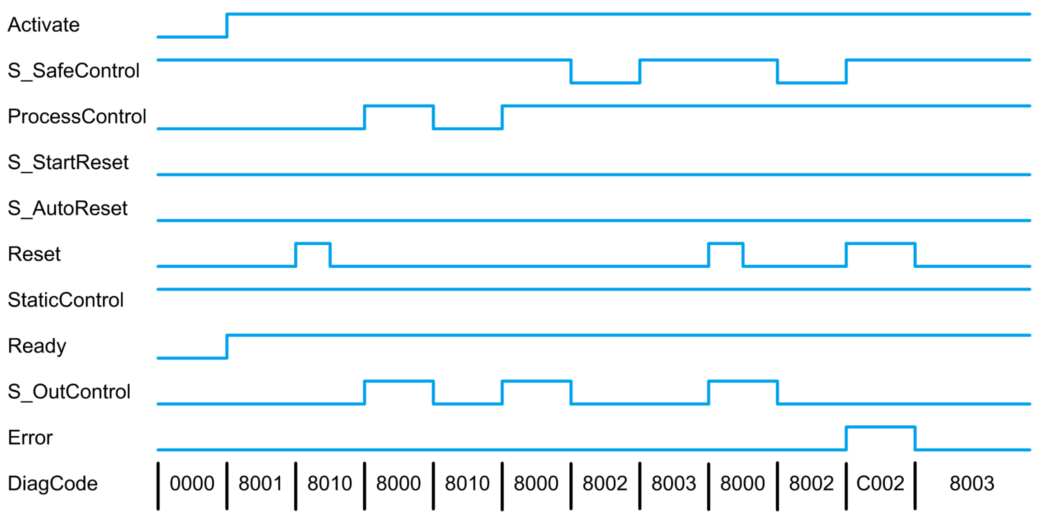
Parameter	Data type	Init Value	Meaning
`Activate`	`BOOL`	FALSE	The variable or constant value indicating activation state of the function block. This parameter can be connected to the variable, which represents the status (Active or Not Active) of the safety-related device. This causes no irrelevant diagnostic information to be generated if a device is disabled: • When FALSE, the output variables are set to the initial values. • Assign a static TRUE signal if no device is connected.
`S_SafeControl`	`BOOL`	FALSE	A variable control signal of the preceding safety-related function block (from the safety library), indicating the state of the safety function: • FALSE: The preceding safety-related function blocks are in the defined safe state, and the associated safety-related function and equipment are disabled. • TRUE: The preceding safety-related function blocks enable functional safety control.

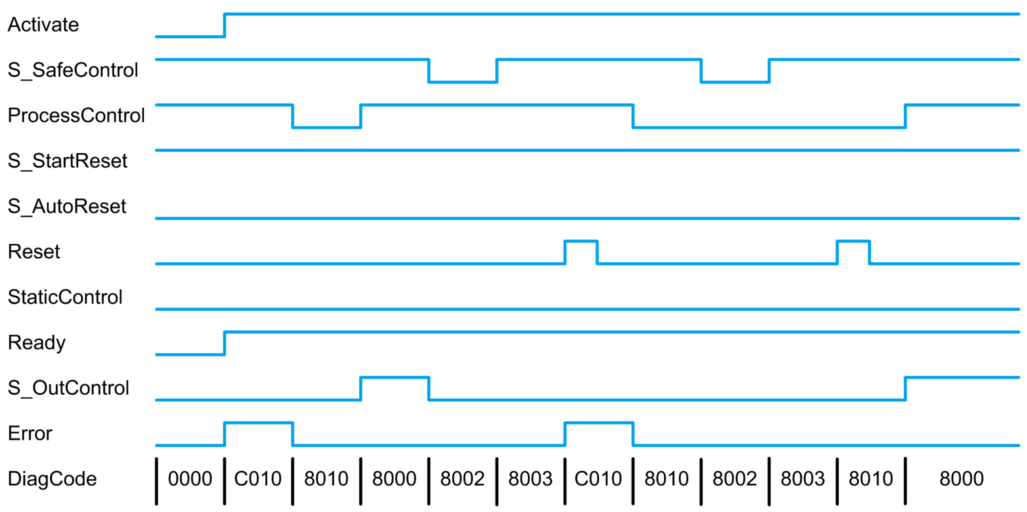
Parameter	Data type	Init Value	Meaning
`ProcessControl`	`BOOL`	FALSE	A variable or constant input signal that from the process control (non-safety) application: • FALSE: Request to set `S_OutControl` to FALSE. • TRUE: Request to set `S_OutControl` to TRUE.
`S_StaticControl`	`BOOL`	FALSE	An optional constant value that indicates additional conditions exist for process control: • FALSE: A dynamic change of the `ProcessControl` parameter (from FALSE to TRUE) is required after block activation or the safety function is triggered. • TRUE: No such dynamic change of the `ProcessControl` parameter is required.
`S_StartReset`	`BOOL`	FALSE	A variable or constant value that indicates: • FALSE: Manual reset when system is started (warm or cold). • TRUE: Automatic reset when system is started (warm or cold). **NOTE:** Activate this function only after you have confirmed that no hazard can occur at the start of the controller. Use of an automatic reset feature of the function blocks requires implementation of other system or application measures to guard against unexpected (or unintended) start-up. Refer to the hazard message in the Function Description above.
`S_AutoReset`	`BOOL`	FALSE	A variable or constant value indicating the state of the auto reset function: • FALSE: Manual reset when emergency stop button is released. • TRUE: Automatic reset when emergency stop button is released **NOTE:** Activate this function only after you have confirmed that no hazard can occur at the start of the controller. Use of an automatic reset feature of the function blocks requires implementation of other system or application measures to guard against unexpected (or unintended) start-up. Refer to the hazard message in the Function Description above.
`Reset`	`BOOL`	FALSE	The variable value indicating reset of the state machine, coupled with detected error and status messages as indicated in the `DiagCode` parameter, after the underlying cause of the detected error has been removed. This behavior is designed as a detected error reset. **NOTE:** This function is only active on a signal change from FALSE to TRUE.

Output Parameters

Parameter	Data type	Init Value	Meaning
`Ready`	`BOOL`	FALSE	• TRUE indicates that the function block is activated and the output results are valid. • FALSE indicates that the function block is not active and the program is not executed. **NOTE:** This parameter can be useful in debug mode or to activate/deactivate additional function blocks, as well as for further processing in the functional program.
`S_OutControl`	`BOOL`	FALSE	Controls the connected actuator(s): • FALSE: Disable connected actuator(s). • TRUE: Enable connected actuator(s).
`Error`	`BOOL`	FALSE	Function block detected error message.
`DiagCode`	`WORD`	0000 hex	Function block diagnostic code.

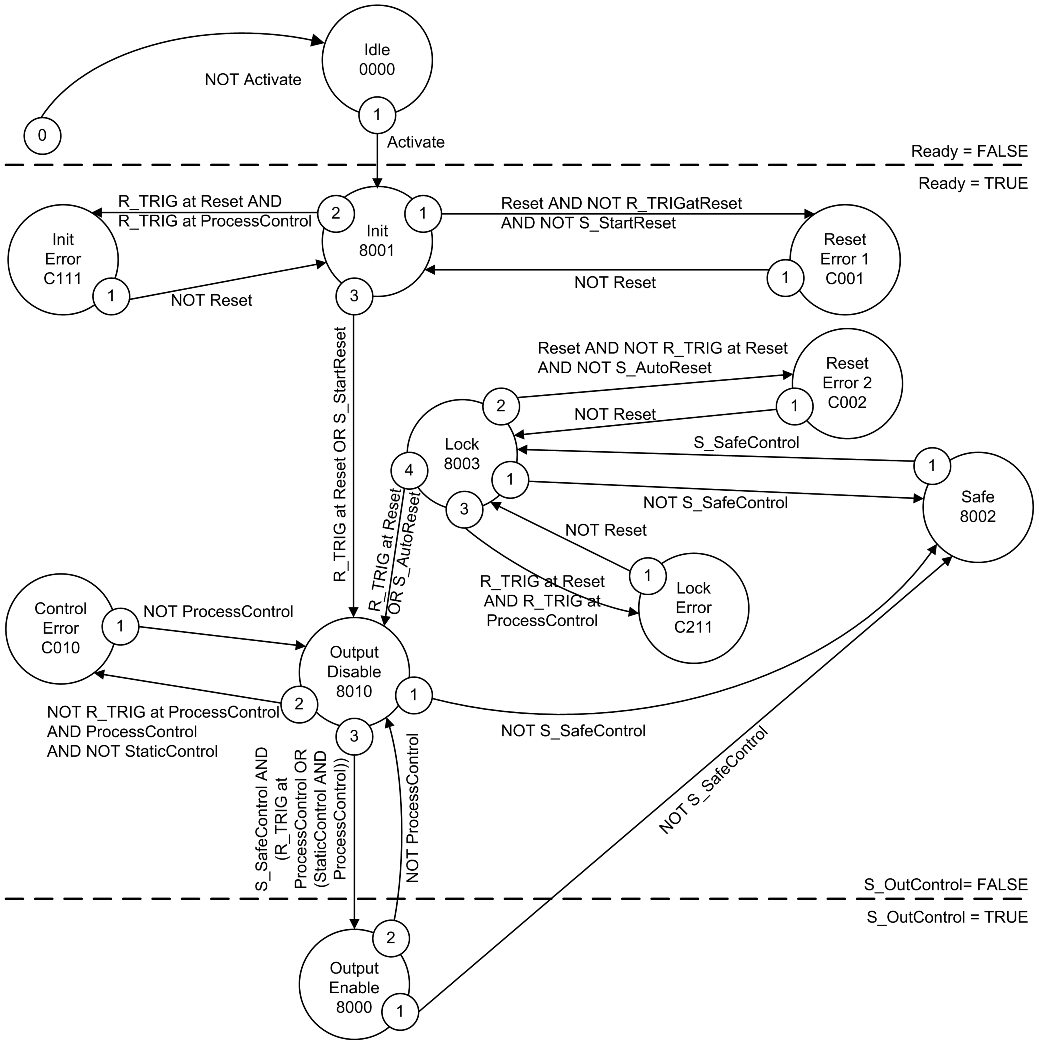
Typical Timing Diagrams




Activate
S_SafeControl
ProcessControl
S_StartReset
S_AutoReset
Reset
StaticControl
Ready
S_OutControl
Error
DiagCode
0000
C010
8010
8000
8002
8003
C010
8010
8002
8003
8010
8000

State Diagram

The following diagram describes the state transitions of the `S_OUTCONTROL` function block:



Source: *PLCopen - Technical Committee 5, Safety Software, Technical Specification, Part 1: Concepts and Function Blocks, Version 1.0*.

NOTE: The transition to the Idle state from any other state, occurring because Activate = FALSE, is not depicted. Such a transition has the highest priority (0).

Error Detection

The following conditions cause a transition to an error state:

- Invalid static `Reset` signal in the process.
- Invalid static `ProcessControl` signal.
- The `ProcessControl` and `Reset` parameters are interconnected due to a programming error.

Detected Error Management

In the event an error is detected, the `S_OutControl` output is set to FALSE and remains in this state.

To leave the `Reset Error`, `Init Error`, or `Lock Error` state, set the `Reset` input to FALSE.

To leave the `Control Error` state, set the `ProcessControl` input to FALSE.

After transition of `S_SafeControl` to TRUE, the optional start-up inhibit can be reset by a rising edge at the `Reset` input. After block activation, the optional start-up inhibit can be reset by a rising edge at the `Reset` input.

When returning a detected error message, the `DiagCode` parameter can present one of the following detected error values

DiagCode	State Name	State Description and Output Settings
C001	Reset Error 1	Static `Reset` signal in state 8001: • `S_OutControl` = FALSE • `Error` = TRUE
C002	Reset Error 2	Static `Reset` signal in state 8003: • `S_OutControl` = FALSE • `Error` = TRUE
C010	Control Error	Static `Reset` signal in state 8010: • `S_OutControl` = FALSE • `Error` = TRUE

DiagCode	State Name	State Description and Output Settings
C111	Init Error	Simultaneous rising edge at `Reset` and `ProcessControl` in state 8001: • `S_OutControl` = FALSE • `Error` = TRUE
C211	Lock Error	Simultaneous rising edge at `Reset` and `ProcessControl` in state 8003: • `S_OutControl` = FALSE • `Error` = TRUE

Diagnostic Codes Management

When returning a status message, the `Error` parameter is set to FALSE, and the `DiagCode` parameter displays one of the following hexadecimal values:

DiagCode	State Name	State Description and Output Settings
0	IDLE	The function block is not active (initial state): • `S_OutControl` = FALSE • `Error` = FALSE
8001	Init	Activation has been detected by the function block, which is now active: • `S_OutControl` = FALSE • `Error` = FALSE
8002	Safe	Triggered safety function: • `S_OutControl` = TRUE • `Error` = FALSE
8003	Lock	Safety function start up inhibit is active. Reset required: • `S_OutControl` = FALSE • `Error` = FALSE
8010	Output Disable	ProcessControl is not active: • `S_OutControl` = FALSE • `Error` = FALSE
8000	Output Enable	ProcessControl is active and preceding safety-related function(s) is (are) enabled: • `S_OutControl` = TRUE • `Error` = FALSE

Sensor

What's in This Part

S_AI_COMP: Analog Input Compare.. 130
S_ANTIVALENT: Compare Antivalent Inputs............................. 137
S_EMERGENCYSTOP: Emergency Stop Monitor.................... 144
S_EQUIVALENT: Compare Equivalent Inputs 152
S_MUTING_PAR: Parallel Muting 158
S_MUTING_SEQ: Sequential Muting 173
S_TWO_HAND_CONTROL_TYPE_II: Two Hand Control 184
S_TWO_HAND_CONTROL_TYPE_III: Two Hand Control with Timer 191

Introduction

This section describes the elementary functions and the elementary function blocks of the `Sensor` family.

S_AI_COMP: Analog Input Compare

What's in This Chapter

Description 130

Introduction

This chapter describes the S_AI_COMP block.

Description

Function Description

Use the S_AI_COMP function block to perform a "one out of two" (1oo2) evaluation of two analog integer values provided by two different sensors:

- If the inputs S_Channel_1 and S_Channel_2 are operating correctly, Health_1 and Health_2 are both set to TRUE. In this case, the block carries out a discrepancy analysis, as follows:
    - If the discrepancy between the inputs S_Channel_1 and S_Channel_2 is greater than the configured difference tolerance (set by Max_Diff) for longer than the configured discrepancy time (set by DiscrepancyTime), a discrepancy error is detected:

      - Error is set to TRUE.

      - DiagCode displays C001, indicating a discrepancy error is detected.

      - Out_Avg is set to 0.

      - Out_Min and Out_Max are maintained.
    - Otherwise:

      - Error is set to FALSE.

      - Out_Avg = (S_Channel_1 + S_Channel_2) / 2.

      - Out_Min = MIN (S_Channel_1;S_Channel_2).

      - Out_Max = MAX (S_Channel_1;S_Channel_2).

- If only one input channel (for example, `S_Channel_1`, but not `S_Channel_2`) is operating correctly, `Health_1` is set to TRUE and `Health_2` is set to FALSE. In this case:
  - `Out_Avg`, `Out_Max`, and `Out_Min` are all set equal to `S_Channel_1`.
  - `Error` is set to TRUE.
- If no input channel (neither `S_Channel_1` nor `S_Channel_2`) is operating correctly, `Health_1` and `Health_2` are both set to FALSE. In this case:
  - `Out_Avg`, `Out_Max` and `Out_Min` are set to 0.
  - `Error` is set to TRUE.

If a discrepancy error is detected, it needs to be acknowledged. This is accomplished after the inputs are again operating within their assigned tolerance, by a rising edge of the `Reset` signal.

The `S_AI_COMP` function block is a modifiable template. You can edit the block structure to meet your application requirements. Therefore, it is not certified by the TÜV, or other agency, because it is not static.

⚠WARNING

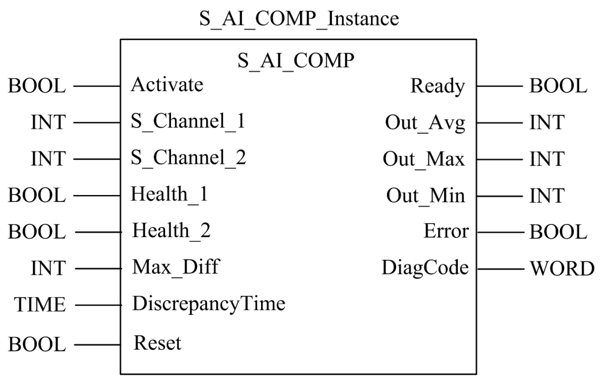
COMPROMISED SAFETY INTEGRITY LEVEL

You must certify the use of the `S_AI_COMP` function block by a recognized agency in accordance with IEC 61508 before it is used in a safety-related application.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Representation in FBD

Representation



Input Parameters

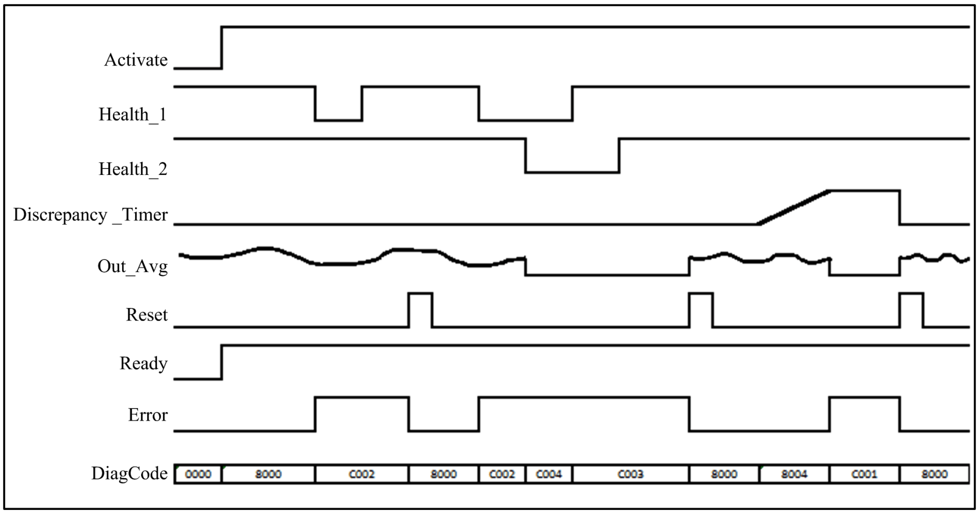
Parameter	Data type	Init Value	Meaning
`Activate`	`BOOL`	FALSE	The variable or constant value indicating activation state of the function block. This parameter can be connected to the variable, which represents the status (Active or Not Active) of the safety-related device. This causes no irrelevant diagnostic information to be generated if a device is disabled: • When FALSE, the output variables are set to the initial values. • Assign a static TRUE signal if no device is connected.
`S_Channel_1`	`INT`	0	Analog input value from channel 1.
`S_Channel_2`	`INT`	0	Analog input value from channel 2.
`Health_1`	`BOOL`	FALSE	Health state of analog signal on channel 1: • FALSE: The value is valid. • TRUE: The value is not valid.
`Health_2`	`BOOL`	FALSE	Health state of analog signal on channel 2: • FALSE: The value is valid. • TRUE: The value is not valid.
`Max_Diff`	`INT`	0	The constant configurable maximum discrepancy value between S_Channel_1 and S_Channel_2.

Parameter	Data type	Init Value	Meaning
`Discrepancy-Time`	`TIME`	T#0ms	The constant configurable maximum monitoring time for comparing S_Channel_1 and S_Channel_2 values.
`Reset`	`BOOL`	FALSE	The variable value indicating reset of the state machine, coupled with detected error and status messages as indicated in the `DiagCode` parameter, after the underlying cause of the detected error has been removed. This behavior is designed as a detected error reset. **NOTE:** This function is only active on a signal change from FALSE to TRUE.

Output Parameters

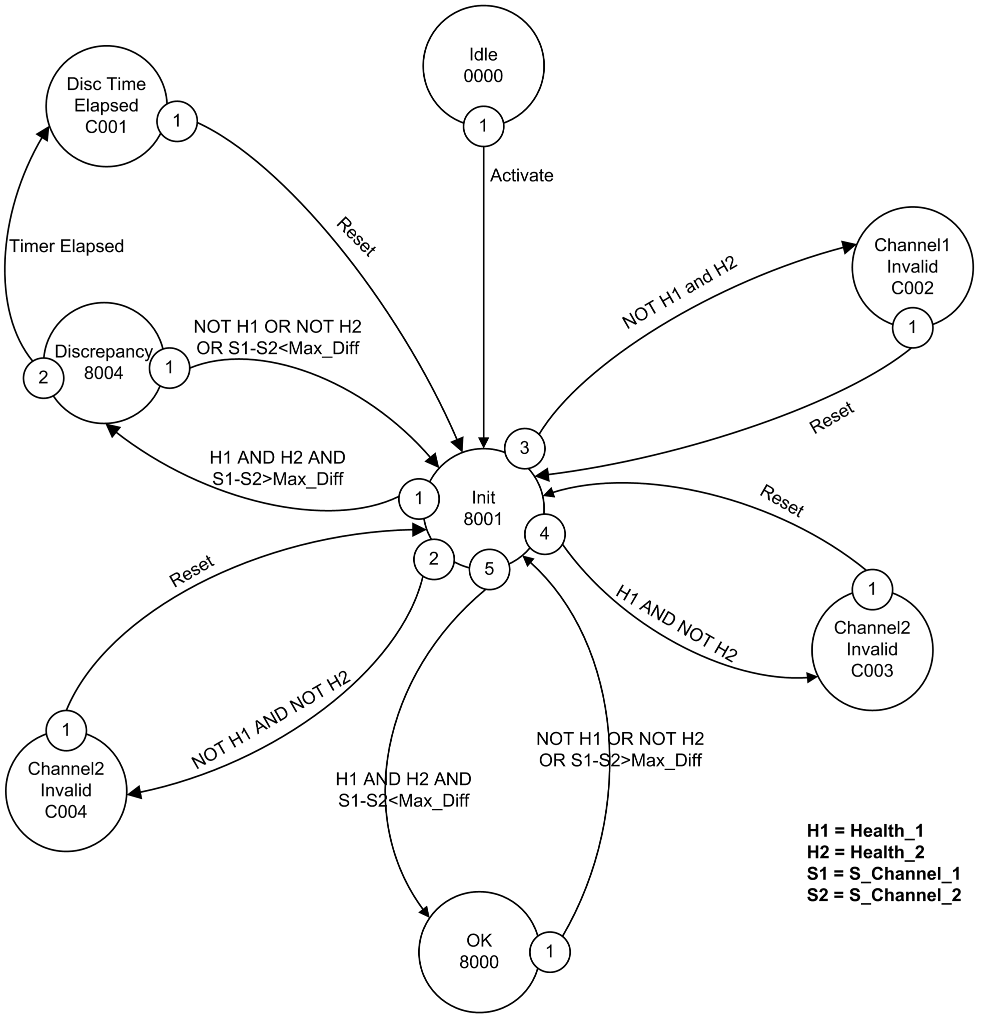
Parameter	Data type	Init Value	Meaning
`Ready`	`BOOL`	FALSE	• TRUE indicates that the function block is activated and the output results are valid. • FALSE indicates that the function block is not active and the program is not executed. **NOTE:** This parameter can be useful in debug mode or to activate/deactivate additional function blocks, as well as for further processing in the functional program.
`Out_Avg`	`INT`	0	Average value of S_Channel_1 and S_Channel_2.
`Out_Max`	`INT`	0	Maximum value between S_Channel_1 and S_Channel_2.
`Out_Min`	`INT`	0	Minimum value between S_Channel_1 and S_Channel_2.
`Error`	`BOOL`	FALSE	Function block detected error message.
`DiagCode`	`WORD`	0000 hex	Function block diagnostic code.

Typical Timing Diagrams



State Diagram

The following diagram describes the state transitions of the `S_EQUIVALENT` function block:



H1 = Health_1
H2 = Health_2
S1 = S_Channel_1
S2 = S_Channel_2

NOTE: The transition to the Idle state from any other state, occurring because Activate = FALSE, is not depicted. Such a transition has the highest priority (0).

Detected Error Management

When an error is detected, `Error` is set to TRUE. The `DiagCode` parameter can present one of the following detected error values:

DiagCode	State Name	State Description and Output Settings
C001	Error 1	`DiscrepancyTime` elapsed in state 8004: • `Error` = TRUE
C002	Error 2	Channel 1 is invalid: • `Error` = TRUE
C003	Error 3	Channel 2 is invalid: • `Error` = TRUE
C004	Error 4	Channel 1 and Channel 2 are invalid: • `Error` = TRUE

Diagnostic Codes Management

When returning a status message, the `Error` parameter is set to FALSE, and the `DiagCode` parameter displays one of the following hexadecimal values:

DiagCode	State Name	State Description and Output Settings
0	IDLE	The function block is not active (initial state): • `Error` = FALSE
8001	INIT	The block detects activation, and is now activated: • `Error` = FALSE
8000	OK	`Out_Avg` contains an average of the two channel values, which is less than the `Max_Diff` setting. The discrepancy timer has not started: • `Error` = FALSE
8003	Discrepancy between channels	The difference between `S_Channel_1` and `S_Channel_2` values exceeds the `Max_Diff` setting. The discrepancy timer is started: • `Error` = FALSE

S_ANTIVALENT: Compare Antivalent Inputs

What's in This Chapter

Description 137

Introduction

This chapter describes the S_ANTIVALENT block.

Description

Function Description

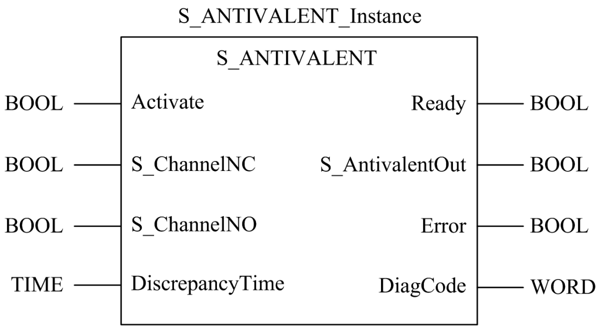
Use the S_ANTIVALENT function block to convert two antivalent BOOL inputs – one input normally closed (NC), the other input normally open (NO) – to a single BOOL output with discrepancy time monitoring.

The two input channels are interdependent. The function block output returns the result of the evaluation of both channels. If S_AntivalentOut = TRUE and the state of one of the safety related inputs changes, the output immediately switches to FALSE.

EN and ENO, page 22 can be configured as additional parameters.

Representation in FBD

Representation



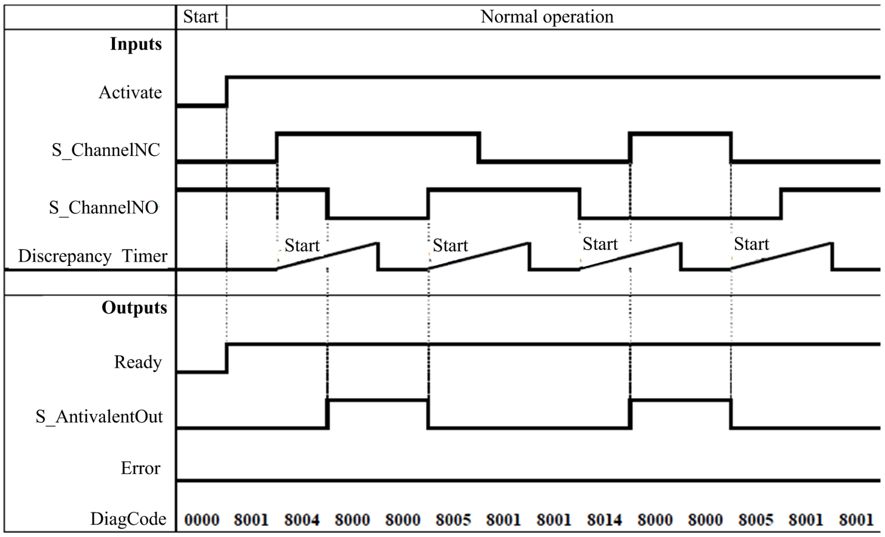
Input Parameters

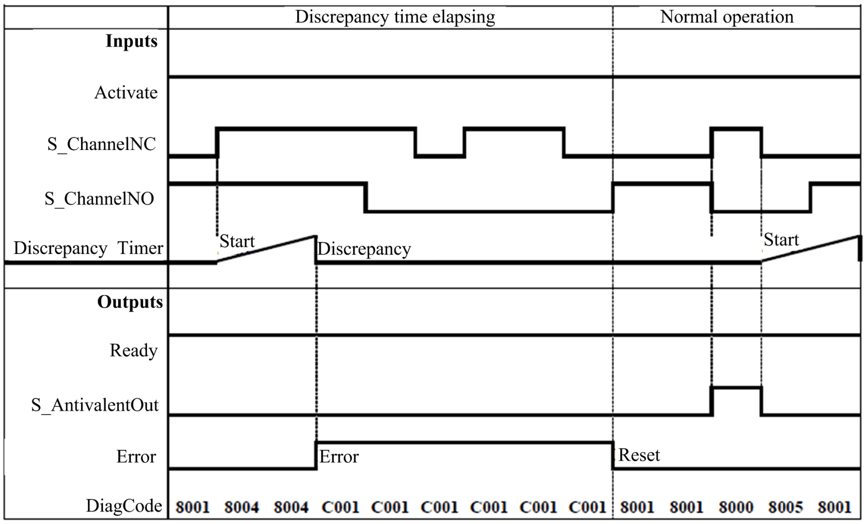
Parameter	Data type	Init Value	Meaning
`Activate`	`BOOL`	FALSE	The variable or constant value indicating activation state of the function block. This parameter can be connected to the variable, which represents the status (Active or Not Active) of the safety-related device. This causes no irrelevant diagnostic information to be generated if a device is disabled: • When FALSE, the output variables are set to the initial values. • Assign a static TRUE signal if no device is connected.
`S_ChannelNC`	`BOOL`	FALSE	The variable value from the normally closed (NC) input channel: • FALSE: NC contact opened. • TRUE: NC contact closed.
`S_ChannelNO`	`BOOL`	TRUE	The variable value from the normally open (NO) input channel: • FALSE: NO contact opened. • TRUE: NO contact closed.
`Discrepancy-Time`	`TIME`	T#0ms	The constant configurable maximum monitoring time for comparing `S_ChannelNC` and `S_ChannelNO` values.

Output Parameters

Parameter	Data type	Init Value	Meaning
`Ready`	`BOOL`	FALSE	• TRUE indicates that the function block is activated and the output results are valid. • FALSE indicates that the function block is not active and the program is not executed. **NOTE:** This parameter can be useful in debug mode or to activate/deactivate additional function blocks, as well as for further processing in the functional program.
`S_ AntivalentOut`	`BOOL`	FALSE	Output value based on a comparison of the two input channel values: • FALSE: One of the following conditions exists: ◦ At least one of the input signals is not active. ◦ A status change has occurred and persisted for longer than the `DiscrepancyTime` setting. • TRUE: Both input signals are active, and any status change is within the `DiscrepancyTime` setting.
`Error`	`BOOL`	FALSE	Function block detected error message.
`DiagCode`	`WORD`	0000 hex	Function block diagnostic code.

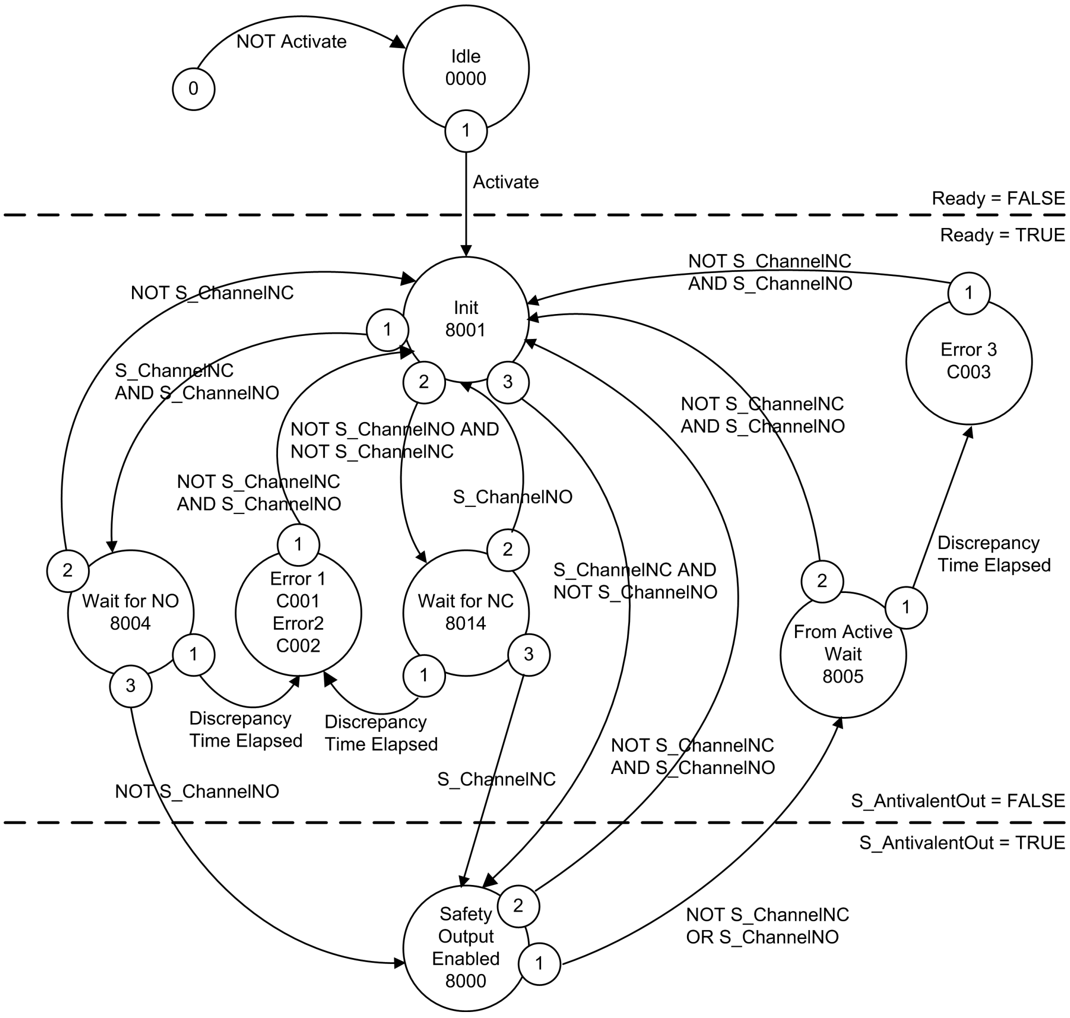
Typical Timing Diagrams




Discrepancy time elapsing
Normal operation
Inputs
Activate
S_ChannelNC
S_ChannelNO
Discrepancy Timer
Start
Discrepancy
Start
Outputs
Ready
S_AntivalentOut
Error
Error
Reset
DiagCode
8001 8004 8004 C001 C001 C001 C001 C001 C001 8001 8001 8000 8005 8001

State Diagram

The following diagram describes the state transitions of the `S_ANTIVALENT` function block:



Source: *PLCopen - Technical Committee 5, Safety Software, Technical Specification, Part 1: Concepts and Function Blocks, Version 1.0*.

NOTE: The transition to the Idle state from any other state, occurring because Activate = FALSE, is not depicted. Such a transition has the highest priority (0).

Error Detection Function

The `S_ANTIVALENT` function block monitors the discrepancy time between the `S_ChannelNC` and `S_ChannelNO` inputs.

Detected Error Management

When an error is detected, `S_AntivalentOut` is set to FALSE, and `Error` is set to TRUE. The `DiagCode` parameter can present one of the following detected error values:

DiagCode	State Name	State Description and Output Settings
C001	Error 1	`DiscrepancyTime` elapsed in state 8004: • `S_AntivalentOut` = FALSE • `Error` = TRUE
C002	Error 2	`DiscrepancyTime` elapsed in state 8014: • `S_AntivalentOut` = FALSE • `Error` = TRUE
C003	Error 3	`DiscrepancyTime` elapsed in state 8005: • `S_AntivalentOut` = FALSE • `Error` = TRUE

Diagnostic Codes Management

When returning a status message, the `Error` parameter is set to FALSE, and the `DiagCode` parameter displays one of the following hexadecimal values:

DiagCode	State Name	State Description and Output Settings
0	IDLE	The function block is not active (initial state): • `S_AntivalentOut` = FALSE • `Error` = FALSE
8001	INIT	The block detects activation, and is now activated: • `S_AntivalentOut` = FALSE • `Error` = FALSE
8000	Safety Output Enabled	The inputs switched to the active state in antivalent mode: • `S_AntivalentOut` = TRUE • `Error` = FALSE

DiagCode	State Name	State Description and Output Settings
8004	Wait for Channel B	`S_ChannelNC` has been switched to TRUE. The function block is now waiting for `S_ChannelNO` to be switched to FALSE. The discrepancy timer has started: • `S_AntivalentOut` = FALSE • `Error` = FALSE
8014	Wait for Channel A	`S_ChannelNO` has been switched to FALSE. The function block is now waiting for `S_ChannelNC` to be switched to TRUE. The discrepancy timer has started: • `S_AntivalentOut` = FALSE • `Error` = FALSE
8005	From Active Wait	One input channel has been switched to inactive. The function block is now waiting for the other input channel also to be switched to inactive: • `S_AntivalentOut` = FALSE • `Error` = FALSE

S_EMERGENCYSTOP: Emergency Stop Monitor

What's in This Chapter

Description 144

Introduction

This chapter describes the `S_EMERGENCYSTOP` block.

Description

Function Description

Use the `S_EMERGENCYSTOP` function block to monitor the state of an emergency stop push button. This function block can be used to activate the emergency switch off functionality.

⚠WARNING

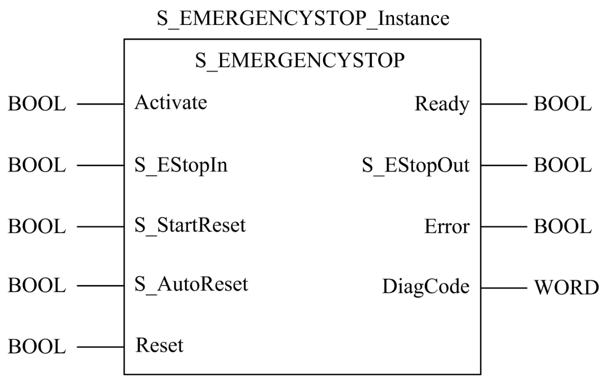
UNINTENDED EQUIPMENT OPERATION

Activate the S_StartReset and S_AutoReset inputs only after you verify that no hazardous situation can occur if the system is started.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Representation in FBD

Representation



Input Parameters

Parameter	Data type	Init Value	Meaning
`Activate`	`BOOL`	FALSE	The variable or constant value indicating activation state of the function block. This parameter can be connected to the variable, which represents the status (Active or Not Active) of the falling edge device. This causes no irrelevant diagnostic information to be generated if a device is disabled: • When FALSE, the output variables are set to the initial values. • Assign a static TRUE signal if no device is connected.
`S_EStopIn`	`BOOL`	FALSE	A variable input signaling the status of an emergency stop button: • FALSE: A request for a safety-related response has been made. The emergency button is engaged. • TRUE: There is no request for a safety-related response. The emergency button is not engaged.

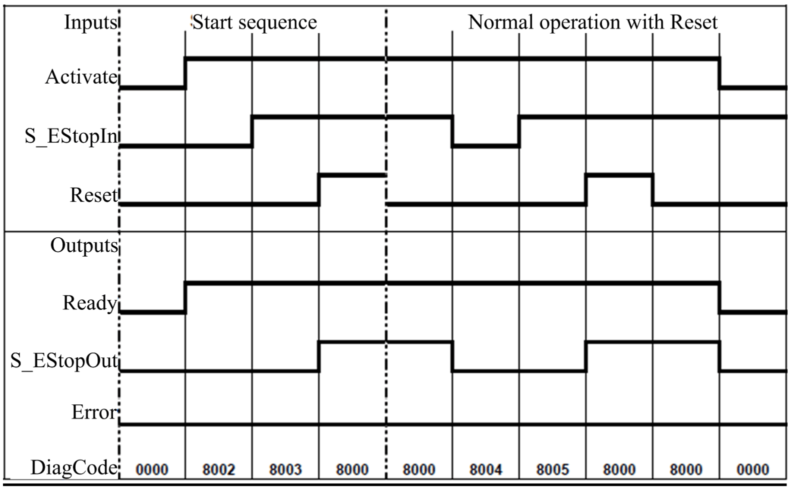
Parameter	Data type	Init Value	Meaning
`S_StartReset`	`BOOL`	FALSE	A variable or constant value that indicates: • FALSE: Manual reset when system is started (warm or cold). • TRUE: Automatic reset when system is started (warm or cold). **NOTE:** Activate this function only after you have confirmed that no hazard can occur at the start of the controller. Use of an automatic reset feature of the function blocks requires implementation of other system or application measures to guard against unexpected (or unintended) start-up. Refer to the hazard message in the Function Description above.
`S_AutoReset`	`BOOL`	FALSE	A variable or constant value indicating the state of the auto reset function: • FALSE: Manual reset when emergency stop button is released. • TRUE: Automatic reset when emergency stop button is released **NOTE:** Activate this function only after you have confirmed that no hazard can occur at the start of the controller. Use of an automatic reset feature of the function blocks requires implementation of other system or application measures to guard against unexpected (or unintended) start-up. Refer to the hazard message in the Function Description above.
`Reset`	`BOOL`	FALSE	The variable value indicating reset of the state machine, coupled with detected error and status messages as indicated in the `DiagCode` parameter, after the underlying cause of the detected error has been removed. This behavior is designed as a detected error reset. **NOTE:** This function is only active on a signal change from FALSE to TRUE.

Output Parameters

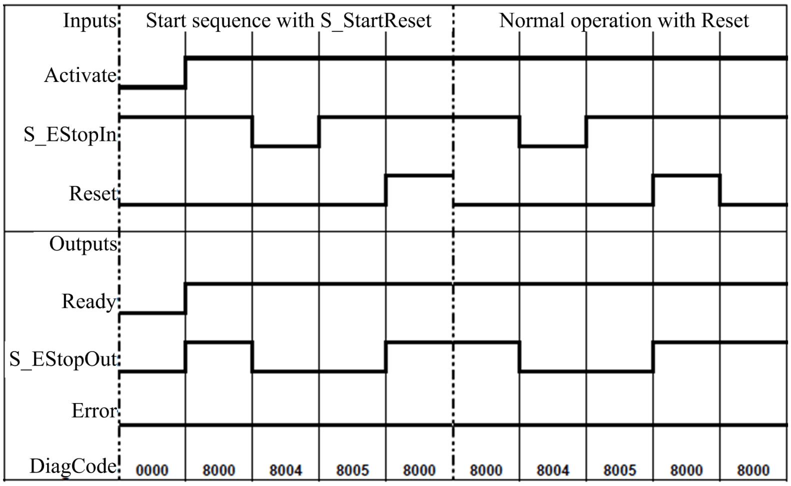
Parameter	Data type	Init Value	Meaning
`Ready`	`BOOL`	FALSE	• TRUE indicates that the function block is activated and the output results are valid. • FALSE indicates that the function block is not active and the program is not executed. **NOTE:** This parameter can be useful in debug mode or to activate/deactivate additional function blocks, as well as for further processing in the functional program.
`S_EStopOut`	`BOOL`	FALSE	A safety related output that indicates: • FALSE: Safety-related output disabled. A request exists for safety-related response (e.g., an emergency stop button is engaged, a reset is required, or internal errors have been detected. • TRUE: Safety-related output enabled. No request for safety-related response (e.g., emergency stop button not engaged, no internal detected errors active).
`Error`	`BOOL`	FALSE	Function block detected error message.
`DiagCode`	`WORD`	0000 hex	Function block diagnostic code.

Typical Timing Diagrams

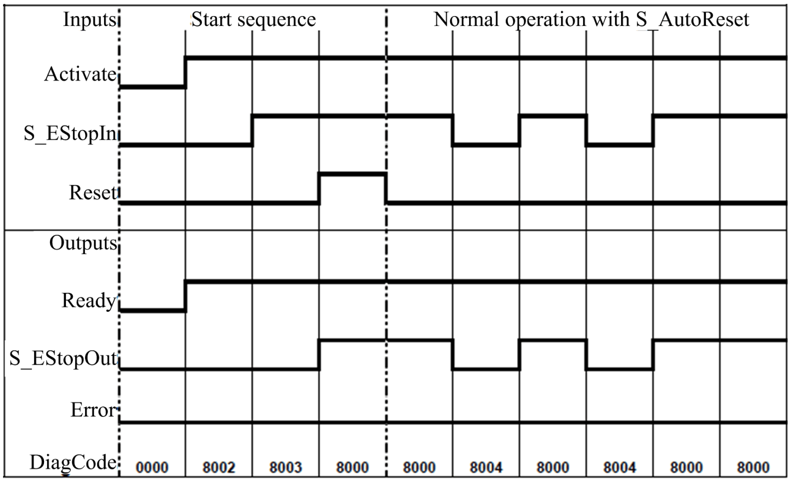
`S_StartReset` = FALSE; `S_AutoReset` = FALSE



`S_StartReset` = TRUE; `S_AutoReset` = FALSE

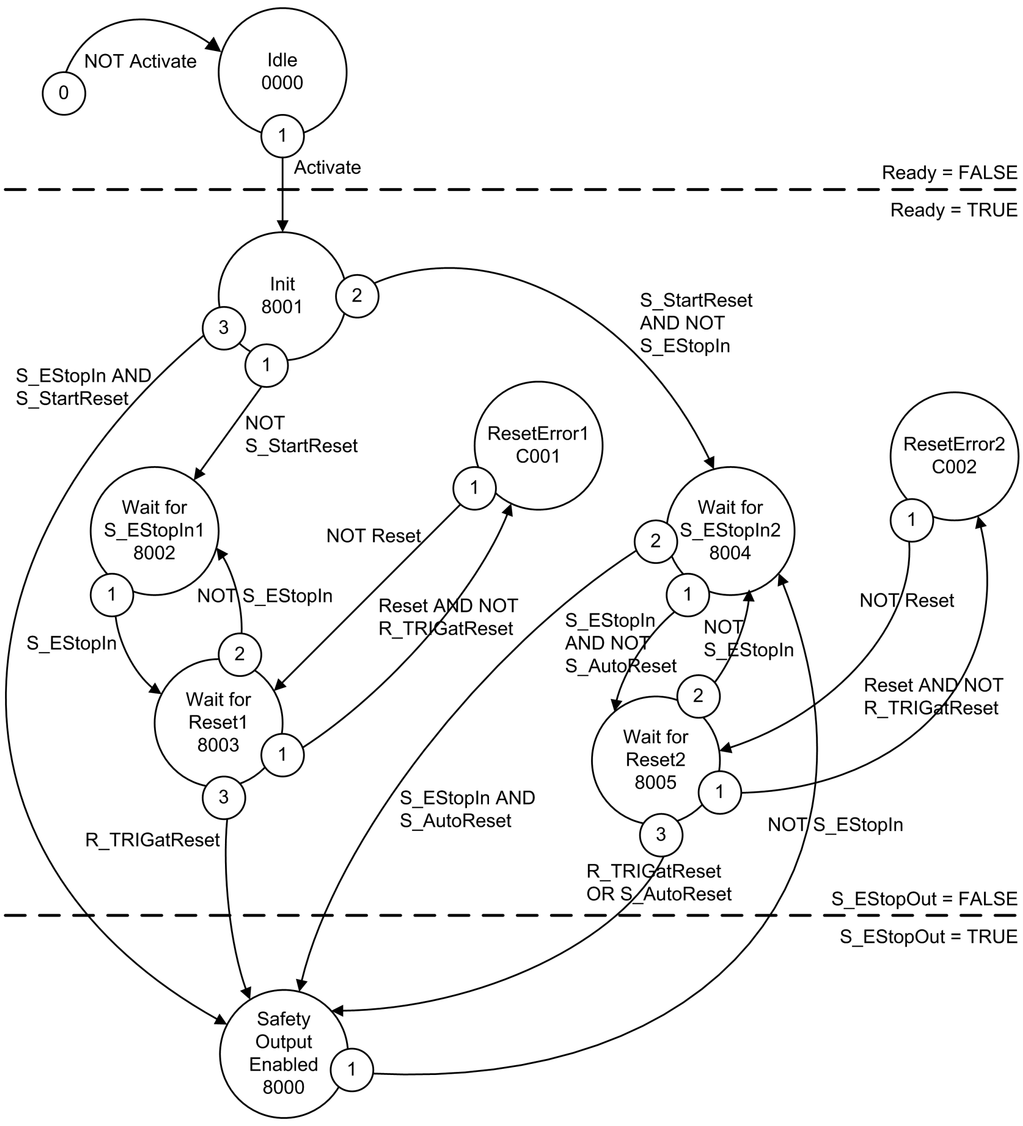


`S_StartReset` = FALSE; `S_AutoReset` = TRUE



State Diagram

The following diagram describes the state transitions of the `S_EMERGENCYSTOP` function block:



Source: *PLCopen - Technical Committee 5, Safety Software, Technical Specification, Part 1: Concepts and Function Blocks, Version 1.0*.

NOTE: The transition to the Idle state from any other state, occurring because Activate = FALSE, is not depicted. Such a transition has the highest priority (0).

Error Detection

The function block detects a static TRUE signal at `Reset` input.

Detected Error Management

`S_EStopOut` is set to an initial value of FALSE.

If a static TRUE signal is received at the `Reset` input, the `DiagCode` output indicates the detected error code and the `Error` output is set to TRUE.

To leave the detected error state, set `Reset` input to FALSE.

When returning a detected error message, the `DiagCode` parameter can present one of the following detected error values

DiagCode	State Name	State Description and Output Settings
C001	Reset Error 1	`Reset` signal is TRUE while waiting for `S_EStopIn` = TRUE: • `S_EStopOut` = FALSE • `Error` = TRUE
C002	Reset Error 2	`Reset` signal is TRUE while waiting for `S_EStopIn` = TRUE: • `S_EStopOut` = FALSE • `Error` = TRUE

Diagnostic Codes Management

When returning a status message, the `Error` parameter is set to FALSE, and the `DiagCode` parameter displays one of the following hexadecimal values:

DiagCode	State Name	State Description and Output Settings
0	IDLE	The function block is not active (initial state): • `S_EStopOut` = FALSE • `Error` = FALSE
8001	INIT	`Activate` is TRUE. The function block was enabled. Check if `S_StartReset` needs to be set: • `S_EStopOut` = FALSE • `Error` = FALSE
8002	Safety Output Enabled	`Activate` is TRUE. Check if `Reset` is FALSE and wait for `S_EStopIn` = TRUE: • `S_EStopIn` = TRUE • `S_EStopOut` = FALSE • `Error` = FALSE
8003	Wait for Channel B	`Activate` is TRUE. `S_EStopIn` = TRUE. Wait for rising edge of `Reset`: • `S_EStopOut` = FALSE • `Error` = FALSE
8004	Wait for Channel A	`Activate` is TRUE. Safety request detected. Check if `Reset` is FALSE and wait for `S_EStopIn` = TRUE: • `S_EStopOut` = FALSE • `Error` = FALSE
8005	From Active Wait	`Activate` is TRUE. S_EStopIn = TRUE. Check for `S_AutoReset` or wait for rising edge of `Reset`: • `S_EStopOut` = FALSE • `Error` = FALSE
8000	From Active Wait	`Activate` is TRUE. S_EStopIn = TRUE. Functional mode with S_EStopOut = TRUE: • `S_EStopOut` = TRUE • `Error` = FALSE

S_EQUIVALENT: Compare Equivalent Inputs

What's in This Chapter

Description 152

Introduction

This chapter describes the `S_EQUVALENT` block.

Description

Function Description

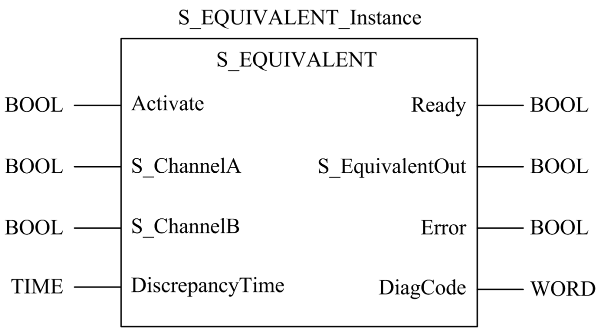
Use the `S_EQUIVALENT` function block to convert two equivalent `BOOL` inputs – both inputs either normally closed (NC) or normally open (NO) – to a single `BOOL` output with discrepancy time monitoring.

NOTE: Because the `S_EQUIVALENT` function block does not include a restart interlock, use it in combination with other safety functions and not as a stand-alone function block.

EN and ENO, page 22 can be configured as additional parameters.

Representation in FBD

Representation



Input Parameters

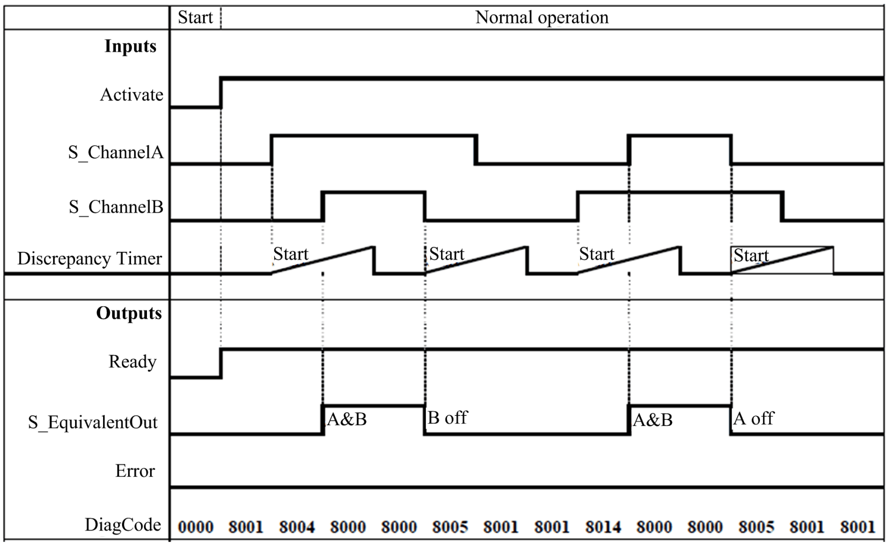
Parameter	Data type	Init Value	Meaning
`Activate`	`BOOL`	FALSE	The variable or constant value indicating activation state of the function block. This parameter can be connected to the variable, which represents the status (Active or Not Active) of the safety-related device. This causes no irrelevant diagnostic information to be generated if a device is disabled: • When FALSE, the output variables are set to the initial values. • Assign a static TRUE signal if no device is connected.
`S_ChannelA`	`BOOL`	FALSE	The variable value from input A: • FALSE: Contact A open. • TRUE: Contact A closed.
`S_ChannelB`	`BOOL`	FALSE	The variable value from input B: • FALSE: Contact B open. • TRUE: Contact B closed.
`Discrepancy-Time`	`TIME`	T#0ms	The constant configurable maximum monitoring time for comparing `S_ChannelA` and `S_ChannelB` values.

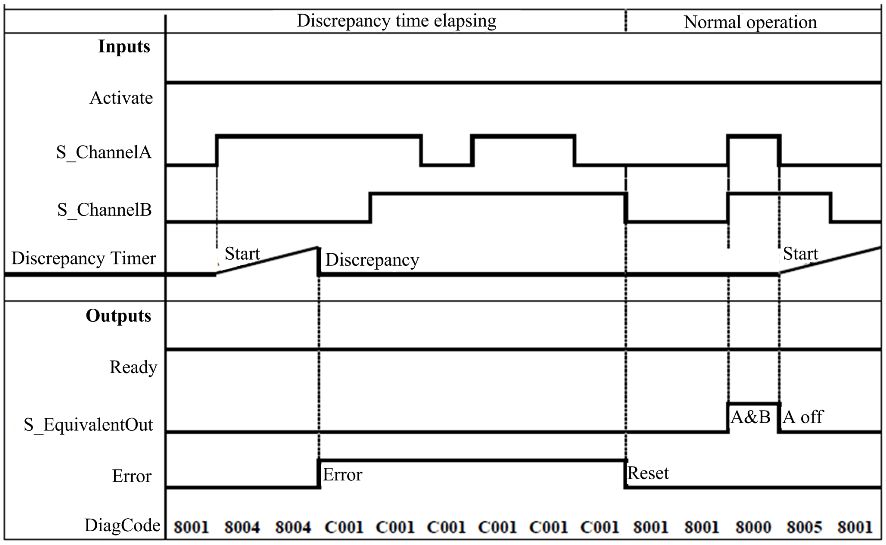
Output Parameters

Parameter	Data type	Init Value	Meaning
`Ready`	`BOOL`	FALSE	• TRUE indicates that the function block is activated and the output results are valid. • FALSE indicates that the function block is not active and the program is not executed. **NOTE:** This parameter can be useful in debug mode or to activate/deactivate additional function blocks, as well as for further processing in the functional program.
`S_EquivalentOut`	`BOOL`	FALSE	Output value based on a comparison of the two input channel values: • FALSE: One of the following conditions exists: ◦ At least one of the input signals is FALSE. ◦ A status change has occurred and persisted for longer than the `DiscrepancyTime` setting. • TRUE: Both input signals are active, and any status change is within the `DiscrepancyTime` setting.

Parameter	Data type	Init Value	Meaning
Error	BOOL	FALSE	Function block detected error message.
DiagCode	WORD	0000 hex	Function block diagnostic code.

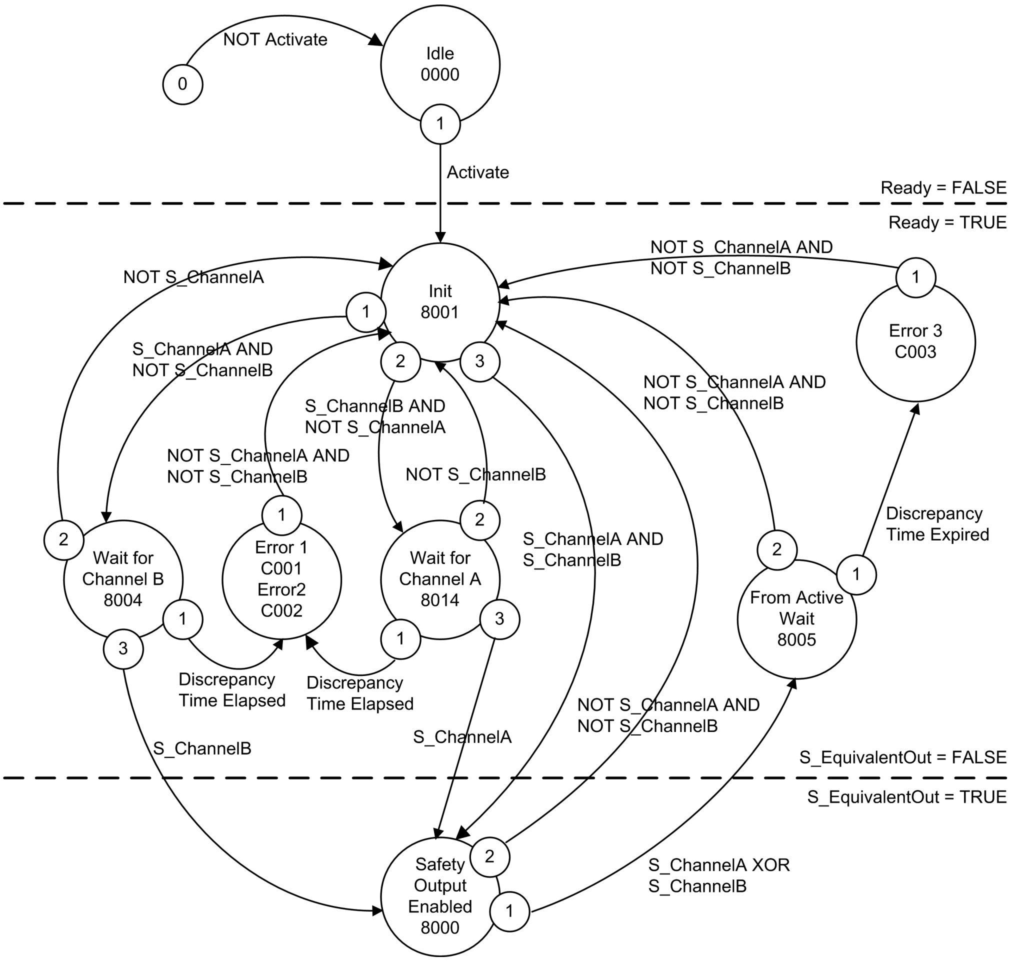
Typical Timing Diagrams





State Diagram

The following diagram describes the state transitions of the `S_EQUIVALENT` function block:



Source: *PLCopen - Technical Committee 5, Safety Software, Technical Specification, Part 1: Concepts and Function Blocks, Version 1.0*.

NOTE: The transition to the Idle state from any other state, occurring because Activate = FALSE, is not depicted. Such a transition has the highest priority (0).

Error Detection Function

The `S_EQUIVALENT` function block monitors the discrepancy time between the `S_ChannelA` and `S_ChannelB` inputs, when switching to TRUE and also when switching to FALSE.

Detected Error Management

When an error is detected, `S_EquivalentOut` is set to FALSE, and `Error` is set to TRUE. The `DiagCode` parameter can present one of the following detected error values:

DiagCode	State Name	State Description and Output Settings
C001	Error 1	`DiscrepancyTime` elapsed in state 8004: • `S_EquivalentOut` = FALSE • `Error` = TRUE
C002	Error 2	`DiscrepancyTime` elapsed in state 8014: • `S_EquivalentOut` = FALSE • `Error` = TRUE
C003	Error 3	`DiscrepancyTime` elapsed in state 8005: • `S_EquivalentOut` = FALSE • `Error` = TRUE

Diagnostic Codes Management

When returning a status message, the `Error` parameter is set to FALSE, and the `DiagCode` parameter displays one of the following hexadecimal values:

DiagCode	State Name	State Description and Output Settings
0	IDLE	The function block is not active (initial state): • `S_EquivalentOut` = FALSE • `Error` = FALSE
8001	INIT	The block detects activation, and is now activated: • `S_EquivalentOut` = FALSE • `Error` = FALSE
8000	Safety Output Enabled	The inputs switched to TRUE in equivalent mode: • `S_EquivalentOut` = TRUE • `Error` = FALSE

DiagCode	State Name	State Description and Output Settings
8004	Wait for Channel B	`S_ChannelA` has been switched to TRUE. The function block is now waiting for `S_ChannelB`. The discrepancy timer has started: • `S_EquivalentOut` = FALSE • `Error` = FALSE
8014	Wait for Channel A	`S_ChannelB` has been switched to TRUE. The function block is now waiting for `S_ChannelA`. The discrepancy timer has started: • `S_EquivalentOut` = FALSE • `Error` = FALSE
8005	From Active Wait	One input channel has been switched to FALSE. The function block is now waiting for the other input channel also to be switched to FALSE. The discrepancy timer has started: • `S_EquivalentOut` = FALSE • `Error` = FALSE

S_MUTING_PAR: Parallel Muting

What's in This Chapter

Description 158

Introduction

This chapter describes the `S_MUTING_PAR` block.

Description

Muting

Muting is the intentional temporary suppression of the safety function implemented by, for example, an Active Opto-electronic Protective Device (AOPD), such as a light curtain, that guards against entry into a restricted zone of operation. Muting is intended to permit necessary materials – but not people – to enter the restricted zone without interrupting the work process. One or two pairs of sensors, properly situated in the production sequence, can be used to trigger muting of the safety function. Each pair of sensors can be situated in parallel positions to operate simultaneously (parallel muting) or can be staggered to operate sequentially (sequential muting).

Muting sensors can be proximity switches, photoelectric barriers, limit switches, and so forth. Indicator lights are applied in the application to indicate that muting is active.

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

Activate the S_StartReset and S_AutoReset inputs only after you verify that no hazardous situation can occur if the system is started.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Function Description

Use the `S_MUTING_PAR` function block to implement parallel muting of a safety-related process. This function block is designed to be used with four muting sensors. You can specify the maximum allowable times for:

- Muting of the safety-related process (`DiscTime11_12` and `DiscTime21_22`) to allow material to enter or exit a detection zone guarded by AOPDs or other guarding devices.
- Completing the entire muting sequence (`MaxMutingTime`), beginning with the initial triggering of muting switches to the exit from the defined zone of operation.

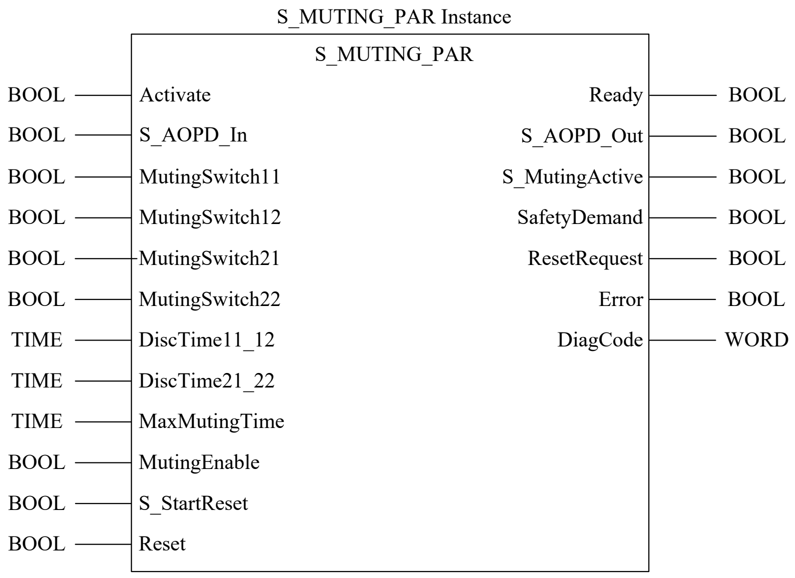
The block supports bi-directional (forward and backward) travel of material.

Muting is enabled by the `MutingEnable` signal, which is issued by the process control system. The `S_MutingActive` signal is set to TRUE to when the muting function is activated.

Upon expiration of the `MaxMutingTime` period, the muting function is canceled.

Representation in FBD

Representation



Input Parameters

Parameter	Data type	Init Value	Meaning
`Activate`	`BOOL`	FALSE	The variable or constant value indicating activation state of the function block. This parameter can be connected to the variable, which represents the status (Active or Not Active) of the safety-related device(s). This causes no irrelevant diagnostic information to be generated if a device is disabled: • When FALSE, the output variables are set to the initial values. • Assign a static TRUE signal if no device is connected.
`S_AOPD_In`	`BOOL`	FALSE	A variable input signal from an output signal switching device (OSSD), typically an output from an AOPD such as a light curtain: • FALSE: The restricted field of the AOPD has been penetrated. • TRUE: The restricted field of the AOPD has not been penetrated.
`MutingSwitch11`	`BOOL`	FALSE	The variable status of muting sensor 11 (i.e. the first sensor in the first pair of sensors): • FALSE: The sensor is not actuated. • TRUE: The sensor is actuated.
`MutingSwitch12`	`BOOL`	FALSE	The variable status of muting sensor 12 (i.e. the second sensor in the first pair of sensors): • FALSE: The sensor is not actuated. • TRUE: The sensor is actuated.
`MutingSwitch21`	`BOOL`	FALSE	The variable status of muting sensor 21 (i.e. the first sensor in the second pair of sensors): • FALSE: The sensor is not actuated. • TRUE: The sensor is actuated.
`MutingSwitch22`	`BOOL`	FALSE	The variable status of muting sensor 22 (i.e. the second sensor in the second pair of sensors): • FALSE: The sensor is not actuated. • TRUE: The sensor is actuated.
`DiscTime11_12`	`TIME`	T#0s	The configurable maximum delay time, from 0...4 seconds, between the triggering of muting switch 11 or 12, and material penetrating the restricted field.
`DiscTime21_22`	`TIME`	T#0s	The configurable maximum delay time, from 0...4 seconds, between the triggering of muting switch 21 or 22, and material penetrating the restricted field.
`MaxMutingTime`	`TIME`	T#0s	The configurable maximum time, from 0 seconds ...10 minutes, for completing a muting sequence, beginning when the first muting sensor is actuated.

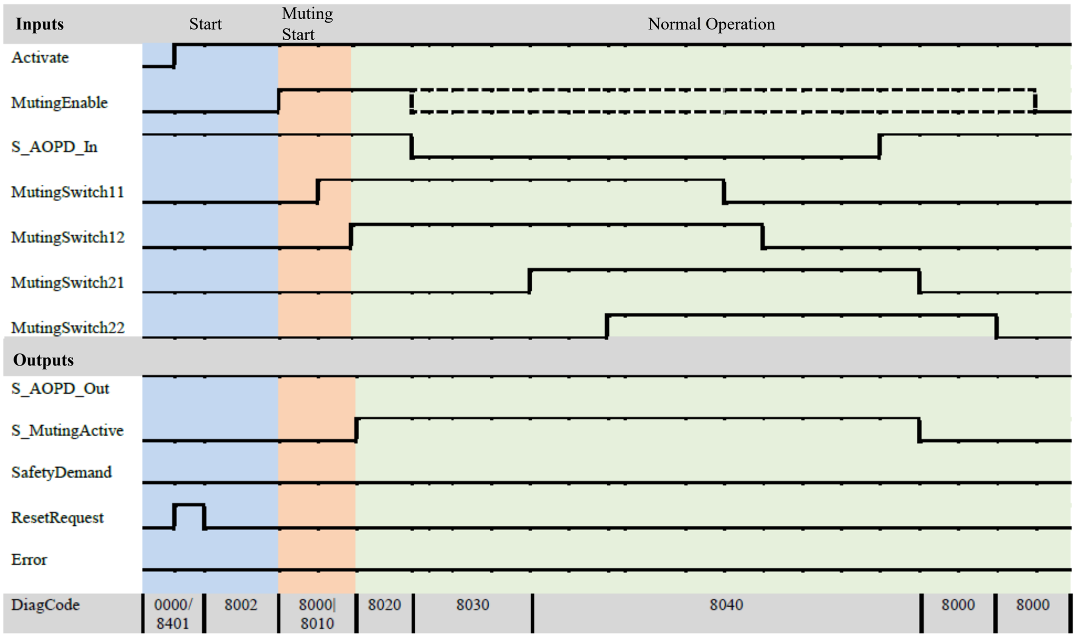
Parameter	Data type	Init Value	Meaning
`MutingEnable`	`BOOL`	FALSE	The variable or constant command, issued by the process control system, that enables or disables the muting function. After the function is enabled, it can be invoked as needed by the process control system: • FALSE: Muting is disabled. • TRUE: Muting is enabled.
`S_StartReset`	`BOOL`	FALSE	A variable or constant value that indicates: • FALSE: Manual reset when system is started (warm or cold). • TRUE: Automatic reset when system is started (warm or cold). **NOTE:** Activate this function only after you have confirmed that no hazard can occur at the start of the controller. Use of an automatic reset feature of the function blocks requires implementation of other system or application measures to guard against unexpected (or unintended) start-up. Refer to the hazard message in the Function Description above.
`Reset`	`BOOL`	FALSE	The variable value indicating a reset of the state machine, coupled with a detected error and status messages as indicated in the `DiagCode` parameter, after the underlying cause of the detected error has been removed. This behavior is designed as a detected error reset. **NOTE:** This function is only active on a signal change from FALSE to TRUE.

Output Parameters

Parameter	Data type	Init Value	Meaning
`Ready`	`BOOL`	FALSE	• TRUE indicates that the function block is activated and the output results are valid. • FALSE indicates that the function block is not active and the program is not executed. **NOTE:** This parameter can be useful in debug mode or to activate/deactivate additional function blocks, as well as for further processing in the functional program.
`S_AOPD_Out`	`BOOL`	FALSE	A safety-related output that indicates the status of the muted output signal switching device (AOPD): • FALSE: The AOPD restricted field has been penetrated and muting is not enabled. • TRUE: The AOPD restricted field has not been penetrated or muting is enabled.
`S_MutingActive`	`BOOL`	FALSE	The status of the muting function: • FALSE: Muting is deactivated. • TRUE: Muting is actived.
`SafetyDemand`	`BOOL`	FALSE	Optional output indicating the function block is active and the primary safety function is requested. Other safety-related input parameters are not considered. The safety-related loop is not closed and the defined safe state is requested for the safety-related output. There is no error detected. • FALSE: No safety-related request is made. • TRUE: A safety-related request is made.
`ResetRequest`	`BOOL`	FALSE	Optional output that can be used to signal the operator to press reset in order to continue. • FALSE: Reset not requested. • TRUE: Reset requested.
`Error`	`BOOL`	FALSE	Detected error flag: • TRUE indicates an error has been detected, and the function block is in error state, as described by the `DiagCode` output. • FALSE indicates no error is detected as described by the `DiagCode` output.
`DiagCode`	`WORD`	0000 hex	Diagnostic code. The states of the function block (Active, Not Active, and Error) are represented in this register. Only the active code is represented at any time. If multiple errors are detected, the `DiagCode` output indicates the first detected error.

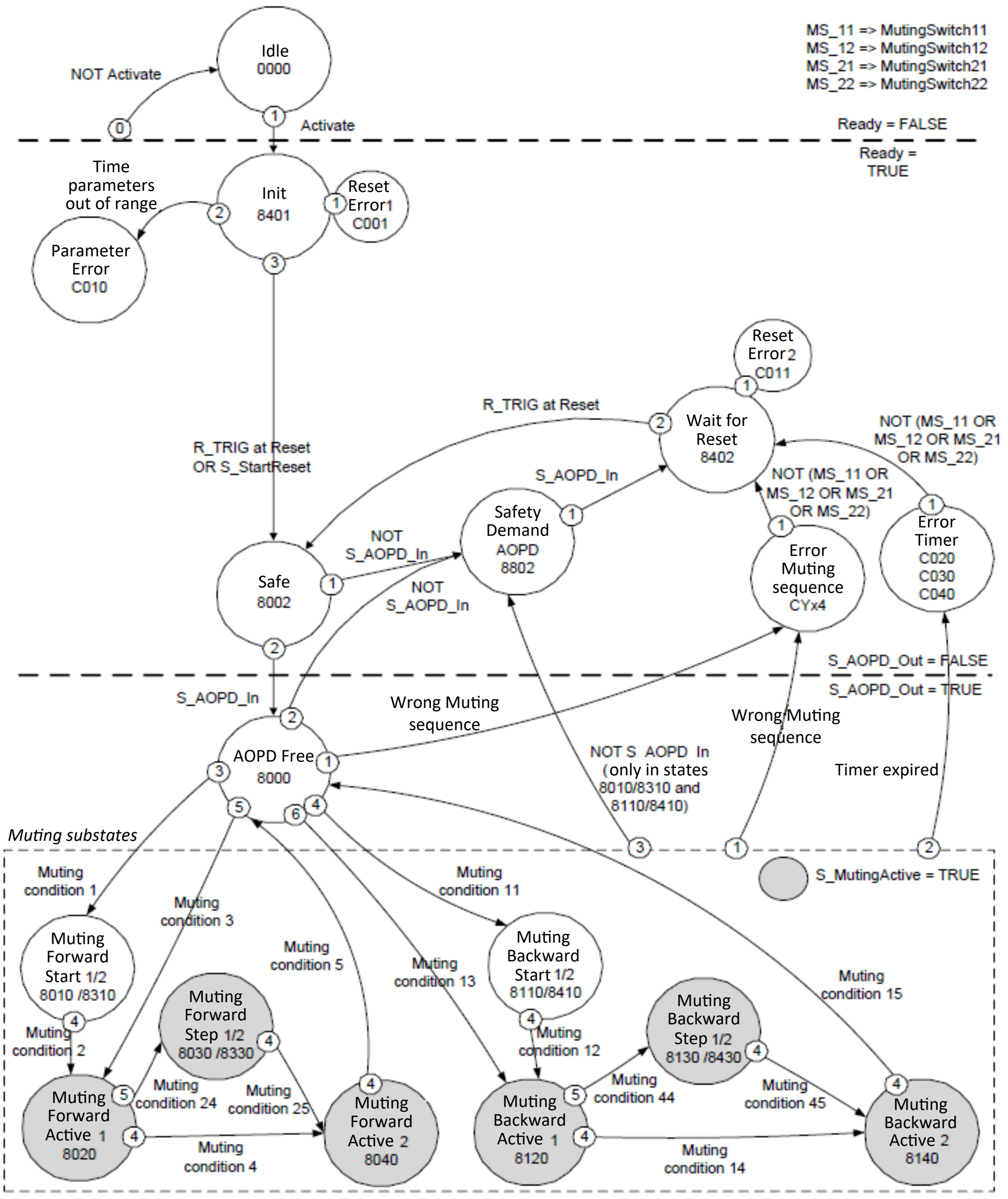
Typical Timing Diagram

Example Timing diagram for S_MUTING_PAR with S_StartReset = TRUE:



State Diagram

The following diagram describes the state transitions of the `S_MUTING_PAR` function block:



Source: *PLCopen - Technical Committee 5, Safety Software, Technical Specification, Part 1: Concepts and Function Blocks, Version 2.01*.

NOTE: The transition to the Idle state from any other state, occurring because Activate = FALSE, is not depicted. Such a transition has the highest priority (0).

Error Detection

The `S_MUTING_PAR` block detects the following error conditions:

- `DiscTime11_12` or `DiscTime21_22` have been set to values less than 0 second or greater than 4 seconds.
- `MaxMutingTime` has been set to a value less than 0 second or greater than 10 minutes.
- The discrepancy time for the `MutingSwitch11/12` or `MutingSwitch21/22` sensor pairs has been exceeded.
- `S_MutingActive` has been equal to TRUE for a period that exceeds the maximum muting time (`MaxMutingTime`) setting.
- Muting sensors `MutingSwitch11`, `MutingSwitch12`, `MutingSwitch21`, and `MutingSwitch22` are activated in an incorrect sequence.
- A muting sequence begins without first being enabled by `MutingEnable`.
- A static `Reset` condition is detected in state 8401 and 8403.

The function block detects a static TRUE signal at the `Reset` input.

Detected Error Management

If an error is detected, the `S_AOPD_Out` and `S_MutingActive` outputs are set to FALSE. The `DiagCode` output indicates the detected error code and the `Error` output is set to TRUE.

NOTE: The `SafetyDemand` output is not impacted by the detected error states. `ResetRequest` is TRUE only in states `Init` and `Wait For Reset`.

A restart is inhibited until the detected error conditions are cleared and the defined safe state is acknowledged using `Reset`.

NOTE: Some **DiagCode** status code values have evolved in the release of *PLCopen - Technical Committee 5, Safety Software, Technical Specification, Part 1: Concepts and Function Blocks, Version 2.01*. Modified **DiagCode** values in the following table are marked with the notation “(*)”.

When returning a detected error message, the `DiagCode` parameter can present one of the following detected error values:

DiagCode	State Name	State Description and Output Settings
C001	Reset Error 1	A static `Reset` condition is detected after the function block activation: • `S_AOPD_Out` = FALSE • `S_MutingActive` = FALSE • `SafetyDemand` = FALSE • `ResetRequest` = FALSE • `Error` = TRUE
C011(*)	Reset Error 2	A static `Reset` condition is detected in state 8003: • `S_AOPD_Out` = FALSE • `S_MutingActive` = FALSE • `SafetyDemand` = FALSE • `ResetRequest` = FALSE • `Error` = TRUE

DiagCode	State Name	State Description and Output Settings
CYx4	Error Muting Sequence	An error is detected in a muting switch in one of the following states: 8000, 8010, 8310, 8012, 8001, 8030, 8330, 8040, 8110, 8410, 8120, 8130, 8430, or 8140: • `S_AOPD_Out` = FALSE • `S_MutingActive` = FALSE • `SafetyDemand` = FALSE • `ResetRequest` = FALSE • `Error` = TRUE **NOTE:** In this `DiagCode`, “Y” and “x” have the following meanings: • Y: Status in the sequence (two states for forward direction & two states for backward direction): ◦ 0: Error detected in state 8000 ◦ 1: Error detected in state Forward 8010 ◦ 2: Error detected in state Forward 8310 ◦ 3: Error detected in state Forward 8020 ◦ 4: Error detected in state Forward 8030 ◦ 5: Error detected in state Forward 8330 ◦ 6: Error detected in state Forward 8040 ◦ 7: Error detected in state Backward 8110 ◦ 8: Error detected in state Backward 8410 ◦ 9: Error detected in state Backward 8120 ◦ A: Error detected in state Backward 8130 ◦ B: Error detected in state Backward 8430 ◦ C: Error detected in state Backward 8140 ◦ F: Muting enable missing • x: Identity of sensor where error detected. One of four bits: ◦ 0: muting switch 11 ◦ 1: muting switch 12 ◦ 2: muting switch 21 ◦ 3: muting switch 22
C010(*)	Parameter Error	`MaxMutingTime` value out of range: • `S_AOPD_Out` = FALSE • `S_MutingActive` = FALSE • `SafetyDemand` = FALSE • `ResetRequest` = FALSE • `Error` = TRUE

DiagCode	State Name	State Description and Output Settings
C020(*)	Error Timer MaxMuting	`S_MutingActive` has been active for longer than the `MaxMutingTime` setting: • `S_AOPD_Out` = FALSE • `S_MutingActive` = FALSE • `SafetyDemand` = FALSE • `ResetRequest` = FALSE • `Error` = TRUE
C030(*)	Error Timer MS11_12	The elapsed time between the triggering of muting switch 11 or 12 and penetrating the restricted field exceeds `DiscTime11_12`: • `S_AOPD_Out` = FALSE • `S_MutingActive` = FALSE • `SafetyDemand` = FALSE • `ResetRequest` = FALSE • `Error` = TRUE
C040(*)	Error Timer MS21_22	The elapsed time between the triggering of muting switch 21 or 22 and penetrating the restricted field exceeds `DiscTime21_22`: • `S_AOPD_Out` = FALSE • `S_MutingActive` = FALSE • `SafetyDemand` = FALSE • `ResetRequest` = FALSE • `Error` = TRUE

Diagnostic Codes Management

When returning a status message, the `Error` parameter is set to FALSE, and the `DiagCode` parameter displays one of the following hexadecimal values:

NOTE: Some **DiagCode** status code values have changed in the release of *PLCopen - Technical Committee 5, Safety Software, Technical Specification, Part 1: Concepts and Function Blocks, Version 2.01*. Changed **DiagCode** values in the following table are marked with the notation “(*)”.

DiagCode	State Name	State Description and Output Settings
0000	IDLE	The function block is not active (initial state): • `S_AOPD_Out` = FALSE • `S_MutingActive` = FALSE • `SafetyDemand` = FALSE • `ResetRequest` = FALSE • `Error` = FALSE
8000	AOPD Free	Muting is not active and no safety request from AOPD. If timers from a prior muting are still running, they are stopped: • `S_AOPD_Out` = TRUE • `S_MutingActive` = FALSE • `SafetyDemand` = FALSE • `ResetRequest` = FALSE • `Error` = FALSE
8401(*)	INIT	`Activate` = TRUE: • `S_AOPD_Out` = FALSE • `S_MutingActive` = FALSE • `SafetyDemand` = FALSE • `ResetRequest` = TRUE • `Error` = FALSE
8802(*)	Safety Demand AOPD	Safety request detected by AOPD and muting is not active: • `S_AOPD_Out` = FALSE • `S_MutingActive` = FALSE • `SafetyDemand` = TRUE • `ResetRequest` = FALSE • `Error` = FALSE
8402(*)	Wait for Reset	Safety request or errors have been detected and have been cleared. Operator acknowledgment by `Reset` required: • `S_AOPD_Out` = FALSE • `S_MutingActive` = FALSE • `SafetyDemand` = FALSE • `ResetRequest` = TRUE • `Error` = FALSE
8002(*)	Safe	Safety function activated Activate = TRUE: • `S_AOPD_Out` = FALSE • `S_MutingActive` = FALSE • `Error` = FALSE

DiagCode	State Name	State Description and Output Settings
8010(*)	Muting Forward Start 1	Muting forward sequence is in starting phase after rising edge of `MutingSwitch_11`. Monitoring of `DiscTime11_12` is activated. Monitoring of `MaxMutingTime` is activated: • `S_AOPD_Out` = TRUE • `S_MutingActive` = FALSE • `Error` = FALSE
8310(*)	Muting Forward Start 2	Muting forward sequence is in starting phase after rising edge of `MutingSwitch_12`. Monitoring of `DiscTime11_12` is activated. Monitoring of `MaxMutingTime` is activated: • `S_AOPD_Out` = TRUE • `S_MutingActive` = FALSE • `Error` = FALSE
8020(*)	Muting Forward Active 1	Muting forward sequence is activated by either: • A rising edge of the second of `MutingSwitch11` or `MutingSwitch12`. • Both `MutingSwitch11` and `MutingSwitch12` being actuated in the same cycle. Monitoring of `DiscTime11_12` is stopped. Monitoring of `MaxMuting_Time` is activated if the transition comes directly from state 8000: • `S_AOPD_Out` = TRUE • `S_MutingActive` = TRUE • `SafetyDemand` = FALSE • `ResetRequest` = FALSE • `Error` = FALSE
8030(*)	Muting Forward Step 1	Muting forward sequence is active. `MutingSwitch21` is the first exit switch activated. Monitoring of `DiscTime21_22` is started: • `S_AOPD_Out` = TRUE • `S_MutingActive` = TRUE • `SafetyDemand` = FALSE • `ResetRequest` = FALSE • `Error` = FALSE
8310(*)	Muting Forward Step 2	Muting forward sequence is active. `MutingSwitch22` is the first exit switch activated. Monitoring of `DiscTime21_22` is started: • `S_AOPD_Out` = TRUE • `S_MutingActive` = TRUE • `SafetyDemand` = FALSE • `ResetRequest` = FALSE • `Error` = FALSE

DiagCode	State Name	State Description and Output Settings
8040(*)	Muting Forward Active 2	Muting forward sequence is still active. Both MutingSwitch21 and MutingSwitch22 are actuated. Monitoring of DiscTime21_22 is stopped: • S_AOPD_Out = TRUE • S_MutingActive = TRUE • SafetyDemand = FALSE • ResetRequest = FALSE • Error = FALSE
8110(*)	Muting Backward Start 1	Muting backward sequence is in starting phase after rising edge of MutingSwitch_21. Monitoring of DiscTime21_22 is activated. Monitoring of MaxMutingTime is activated: • S_AOPD_Out = TRUE • S_MutingActive = FALSE • SafetyDemand = FALSE • ResetRequest = FALSE • Error = FALSE
8410(*)	Muting Backward Start 2	Muting backward sequence is in starting phase after rising edge of MutingSwitch_22. Monitoring of DiscTime21_22 is activated. Monitoring of MaxMutingTime is activated: • S_AOPD_Out = TRUE • S_MutingActive = FALSE • SafetyDemand = FALSE • ResetRequest = FALSE • Error = FALSE
8120(*)	Muting Backward Active 1	Muting backward sequence is activated by either: • A rising edge of the second of MutingSwitch21 or MutingSwitch22. • Both MutingSwitch21 and MutingSwitch22 being actuated in the same cycle. Monitoring of DiscTime21_22 is stopped. Monitoring of MaxMuting_Time is activated if the transition comes directly from state 8000: • S_AOPD_Out = TRUE • S_MutingActive = TRUE • SafetyDemand = FALSE • ResetRequest = FALSE • Error = FALSE
8130(*)	Muting Backward Step 1	Muting backward sequence is active. MutingSwitch11 is the first exit switch activated. Monitoring of DiscTime11_12 is started: • S_AOPD_Out = TRUE • S_MutingActive = TRUE • Error = FALSE

DiagCode	State Name	State Description and Output Settings
8430(*)	Muting Backward Step 2	Muting backward sequence is active. `MutingSwitch12` is the first exit switch activated. Monitoring of `DiscTime11_12` is started: • `S_AOPD_Out` = TRUE • `S_MutingActive` = TRUE • `SafetyDemand` = FALSE • `ResetRequest` = FALSE • `Error` = FALSE
8140(*)	Muting Backward Active 2	Muting backward sequence is active. Both `MutingSwitch11` and `MutingSwitch12` are actuated. Monitoring of `DiscTime11_12` is stopped: • `S_AOPD_Out` = TRUE • `S_MutingActive` = TRUE • `SafetyDemand` = FALSE • `ResetRequest` = FALSE • `Error` = FALSE

S_MUTING_SEQ: Sequential Muting

What's in This Chapter

Description 173

Introduction

This chapter describes the `S_MUTING_SEQ` block.

Description

Muting

Muting is the intentional temporary suppression of the safety function implemented by, for example, an Active Opto-electronic Protective Device (AOPD), such as a light curtain, that guards against entry into a restricted zone of operation. Muting is intended to permit necessary materials – but not people – to enter the restricted zone without interrupting the work process. Two or more sensors, properly situated in the production sequence, can be used to trigger muting of the safety function. Sensors can be situated in parallel positions to operate simultaneously (parallel muting) or can be staggered to operate sequentially (sequential muting).

Muting sensors can be proximity switches, photoelectric barriers, limit switches, and so forth. Indicator lights are applied in the application to indicate that muting is active.

⚠WARNING

UNINTENDED EQUIPMENT OPERATION

Activate the S_StartReset and S_AutoReset inputs only after you verify that no hazardous situation can occur if the system is started.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Function Description

Use the `S_MUTING_SEQ` function block to implement sequential muting of a safety-related process. This function block is designed to be used with four muting sensors. You can specify the maximum allowable time for completing the entire muting sequence (`MaxMutingTime`), beginning with the initial triggering of muting switches to the exit from the defined zone of operation.

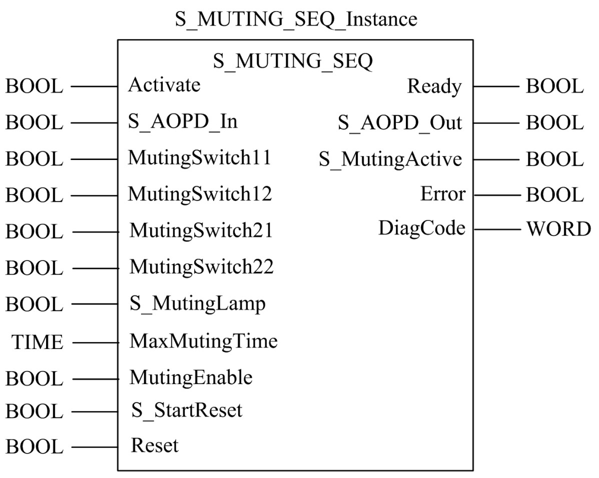
The block supports bi-directional (forward and backward) travel of material.

Muting is enabled by the `MutingEnable` signal, which is issued by the process control system. The `S_MutingActive` signal is set to TRUE to when the muting function is activated. The `S_MutingLamp` signal indicates if indicator lights are functioning while the muting function is active.

Upon expiration of the `MaxMutingTime` period, the muting function is canceled and the S_ MutingLamp signal is set to FALSE to indicate to the operator that the muting function is inactive.

Representation in FBD

Representation



Input Parameters

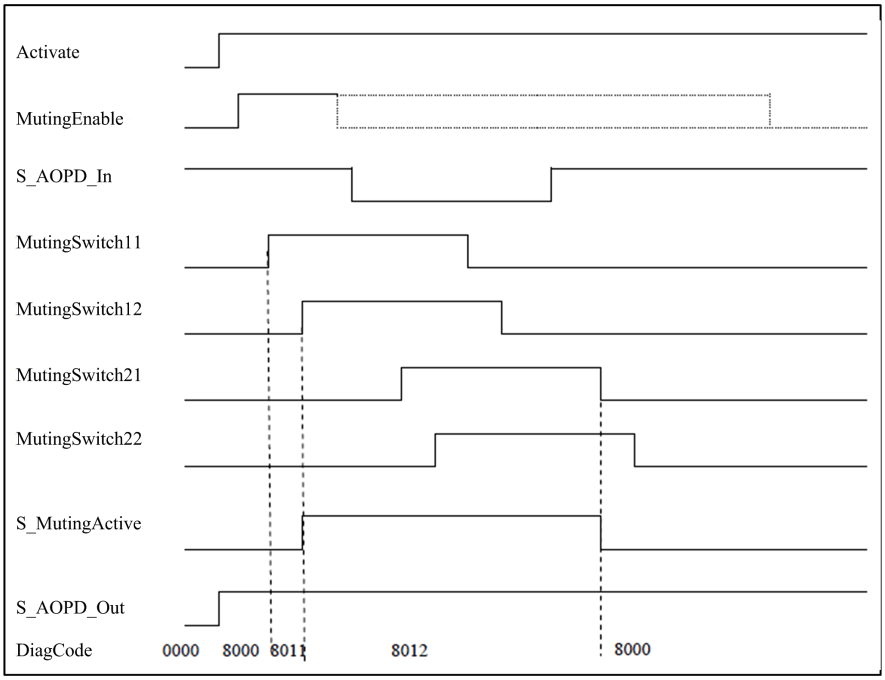
Parameter	Data type	Init Value	Meaning
`Activate`	`BOOL`	FALSE	The variable or constant value indicating activation state of the function block. This parameter can be connected to the variable, which represents the status (Active or Not Active) of the safety-relate device(s). This causes no irrelevant diagnostic information to be generated if a device is disabled: • When FALSE, the output variables are set to the initial values. • Assign a static TRUE signal if no device is connected.
`S_AOPD_In`	`BOOL`	FALSE	A variable input signal from an output signal switching device (OSSD), typically an output from an AOPD such as a light curtain: • FALSE: The protection field of the AOPD has been penetrated. • TRUE: The protection field of the AOPD has not been penetrated.
`MutingSwitch11`	`BOOL`	FALSE	The variable status of muting sensor 11: • FALSE: The sensor is not actuated. • TRUE: The sensor is actuated.
`MutingSwitch12`	`BOOL`	FALSE	The variable status of muting sensor 12: • FALSE: The sensor is not actuated. • TRUE: The sensor is actuated.
`MutingSwitch21`	`BOOL`	FALSE	The variable status of muting sensor 21: • FALSE: The sensor is not actuated. • TRUE: The sensor is actuated.
`MutingSwitch22`	`BOOL`	FALSE	The variable status of muting sensor 22: • FALSE: The sensor is not actuated. • TRUE: The sensor is actuated.
`S_MutingLamp`	`BOOL`	FALSE	The variable or constant value indicating the operational status of the muting lamp: • FALSE: The lamp is not functional. • TRUE: The lamp is operating normally.
`MaxMutingTime`	`TIME`	T#0s	The configurable maximum time, from 0 seconds ...10 minutes, for completing a muting sequence, beginning when the first sensor is actuated.

Parameter	Data type	Init Value	Meaning
MutingEnable	BOOL	FALSE	The variable or constant command, issued by the process control system, that enables or disables the muting function. After the function is enabled, it can be invoked as needed by the process control system: • FALSE: Muting is disabled. • TRUE: Muting is enabled.
S_StartReset	BOOL	FALSE	A variable or constant value that indicates: • FALSE: Manual reset when system is started (warm or cold). • TRUE: Automatic reset when system is started (warm or cold). **NOTE:** Activate this function only after you have confirmed that no hazard can occur at the start of the controller. Use of an automatic reset feature of the function blocks requires implementation of other system or application measures to guard against unexpected (or unintended) start-up. Refer to the hazard message in the Function Description above.
Reset	BOOL	FALSE	The variable value indicating a reset of the state machine, coupled with a detected error and status messages as indicated in the DiagCode parameter, after the underlying cause of the detected error has been removed. This behavior is designed as a detected error reset. **NOTE:** This function is only active on a signal change from FALSE to TRUE.

Output Parameters

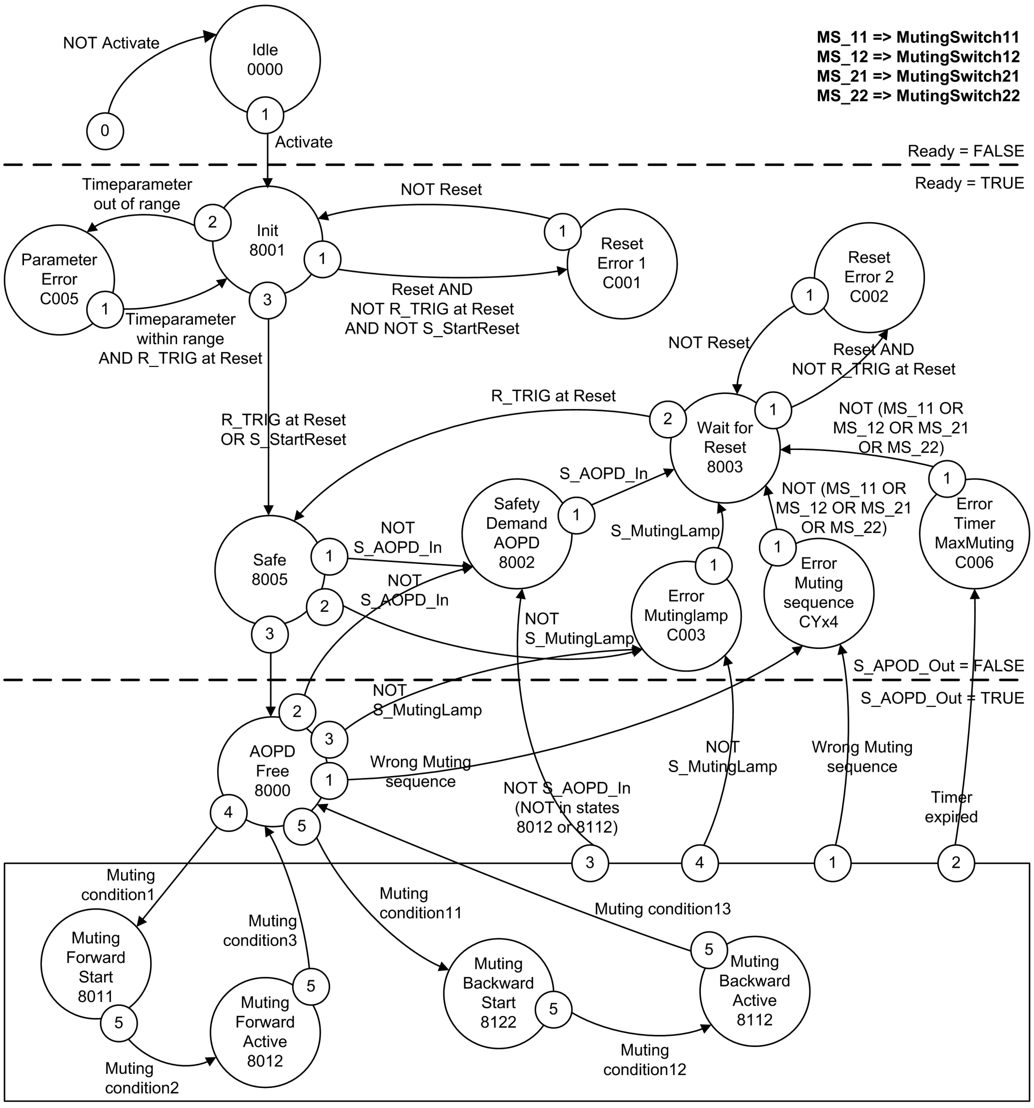
Parameter	Data type	Init Value	Meaning
`Ready`	`BOOL`	FALSE	• TRUE indicates that the function block is activated and the output results are valid (same as the POWER LED of a safety relay). • FALSE indicates that the function block is not active and the program is not executed. **NOTE:** This parameter can be useful in debug mode or to activate/deactivate additional function blocks, as well as for further processing in the functional program.
`S_AOPD_Out`	`BOOL`	FALSE	A safety-related output that indicates the status of the muted output signal switching device (AOPD): • FALSE: The AOPD protection field has been penetrated and muting is not enabled. • TRUE: The AOPD protection field has not been penetrated or muting is enabled.
`S_MutingActive`	`BOOL`	FALSE	The status of the muting function: • FALSE: Muting is disabled. • TRUE: Muting is enabled.
`Error`	`BOOL`	FALSE	Function block detected error message.
`DiagCode`	`WORD`	0000 hex	Function block diagnostic code.

Typical Timing Diagram



State Diagram

The following diagram describes the state transitions of the `S_MUTING_SEQ` function block:



Source: *PLCopen - Technical Committee 5, Safety Software, Technical Specification, Part 1: Concepts and Function Blocks, Version 1.0*.

NOTE:

- The transition to the Idle state from any other state, occurring because Activate = FALSE, is not depicted. Such a transition has the highest priority (0).
- Within muting substates, transitions due to Error Muting sequence (priority 1), Error Timer (priority 2), Safety demand AOPD (priority 3) or Error Muting lamp (priority 4) have higher priority than transitions to Muting substates (priority 5).

Error Detection

The `S_MUTING_SEQ` block detects the following detected error conditions:

- `MaxMutingTime` has been set to a value less than 0 second or greater than 10 minutes.
- `S_MutingActive` has been equal to TRUE for a period that exceeds the maximum muting time (`MaxMutingTime`) setting.
- Muting sensors `MutingSwitch11`, `MutingSwitch12`, `MutingSwitch21`, and `MutingSwitch22` are activated in an incorrect sequence.
- A muting sequence begins without first being enabled by `MutingEnable`.
- A not operational muting lamp is indicated by `S_MutingLamp` = FALSE
- A static `Reset` condition is detected.

Detected Error Management

If an error is detected, the `S_AOPD_Out` and `S_MutingActive` outputs are set to FALSE. The `DiagCode` output indicates the detected error code and the `Error` output is set to TRUE.

A restart is inhibited until the detected error conditions are cleared and the defined safe state is acknowledged using `Reset`.

When returning a detected error message, the `DiagCode` parameter can present one of the following detected error values:

DiagCode	State Name	State Description and Output Settings
C001	Reset Error 1	A static `Reset` condition is detected after the function block activation: • `S_AOPD_Out` = FALSE • `S_MutingActive` = FALSE • `Error` = TRUE
C002	Reset Error 2	A static `Reset` condition is detected in state 8003: • `S_AOPD_Out` = FALSE • `S_MutingActive` = FALSE • `Error` = TRUE
C003	Error Muting Lamp	An error is detected in the muting lamp: • `S_AOPD_Out` = FALSE • `S_MutingActive` = FALSE • `Error` = TRUE
CYx4	Error Muting Sequence	An error is detected in a muting switch in one of the following states: 8000, 8011, 8012, 8112, or 8122: • `S_AOPD_Out` = FALSE • `S_MutingActive` = FALSE • `Error` = TRUE **NOTE:** In this `DiagCode`, “Y” and “x” have the following meanings: • Y: Status in the sequence: ◦ 0:Error detected in state 8000 ◦ 1: Error detected in state Forward 8011 ◦ 2: Error detected in state Forward 8012 ◦ 3: Error detected in state Backward 8122 ◦ 4: Error detected in state Backward 8112 ◦ F: Muting enable missing • x: Identity of sensor where error detected. One of four bits: ◦ 0: muting switch 11 ◦ 1: muting switch 12 ◦ 2: muting switch 21 ◦ 3: muting switch 22

DiagCode	State Name	State Description and Output Settings
C005	Parameter Error	MaxMutingTime value out of range: • S_AOPD_Out = FALSE • S_MutingActive = FALSE • Error = TRUE
C006	Error Timer MaxMuting	S_MutingActive has been active for longer than the MaxMutingTime setting: • S_AOPD_Out = FALSE • S_MutingActive = FALSE • Error = TRUE

Diagnostic Codes Management

When returning a status message, the Error parameter is set to FALSE, and the DiagCode parameter displays one of the following hexadecimal values:

DiagCode	State Name	State Description and Output Settings
0	IDLE	The function block is not active (initial state): • S_AOPD_Out = FALSE • S_MutingActive = FALSE • Error = FALSE
8000	AOPD Free	Muting is not active and no safety-related request from AOPD. If timers from a prior muting are still running, they are stopped: • S_AOPD_Out = TRUE • S_MutingActive = FALSE • Error = FALSE
8001	INIT	Activate = TRUE: • S_AOPD_Out = FALSE • S_MutingActive = FALSE • Error = FALSE
8002	Safety Demand AOPD	safety-related request detected by AOPD and muting is not active: • S_AOPD_Out = FALSE • S_MutingActive = FALSE • Error = FALSE
8003	Wait for Reset	safety-related request or errors have been detected and have been cleared. Operator acknowledgment by Reset required: • S_AOPD_Out = FALSE • S_MutingActive = FALSE • Error = FALSE

DiagCode	State Name	State Description and Output Settings
8005	Safe	Safety function activated (`Activate` = TRUE): • `S_AOPD_Out` = FALSE • `S_MutingActive` = FALSE • `Error` = FALSE
8011	Muting Forward Start	Muting forward sequence is in starting phase and no safety-related request: • `S_AOPD_Out` = TRUE • `S_MutingActive` = FALSE • `Error` = FALSE
8012	Muting Forward Active	Muting forward sequence is active: • `S_AOPD_Out` = TRUE • `S_MutingActive` = TRUE • `Error` = FALSE
8112	Muting Backward Active	Muting backward sequence is active: • `S_AOPD_Out` = TRUE • `S_MutingActive` = TRUE • `Error` = FALSE
8122	Muting Backward Start	Muting backward sequence is in starting phase and no safety-related request: • `S_AOPD_Out` = TRUE • `S_MutingActive` = FALSE • `Error` = FALSE

S_TWO_HAND_CONTROL_TYPE_II: Two Hand Control

What's in This Chapter

Description 184

Introduction

This chapter describes the `S_TWO_HAND_CONTROL_TYPE_II` block.

Description

Function Description

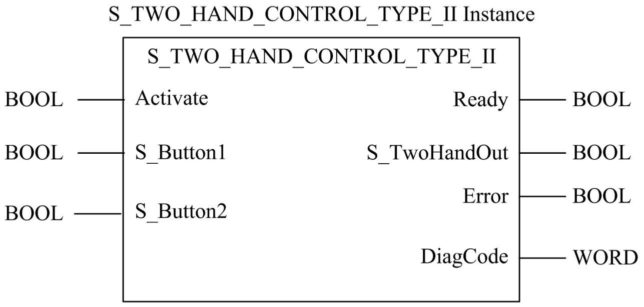
Use the `S_TWO_HAND_CONTROL_TYPE_II` function block to provide two-hand control functionality to a manufacturing process. Two-hand control requires that each hand of the operator be placed on a separate control button, to help prevent potential hazards to the operator.

When both control buttons are actuated (i.e. pressed down), the `S_Button1` and `S_Button2` parameters are set to TRUE. In this state, if the `Error` signal remains FALSE, the `S_TwoHandOut` output is set to TRUE.

This function block also controls the release of both buttons before setting the `S_TwoHandOut` output to TRUE.

Representation in FBD

Representation



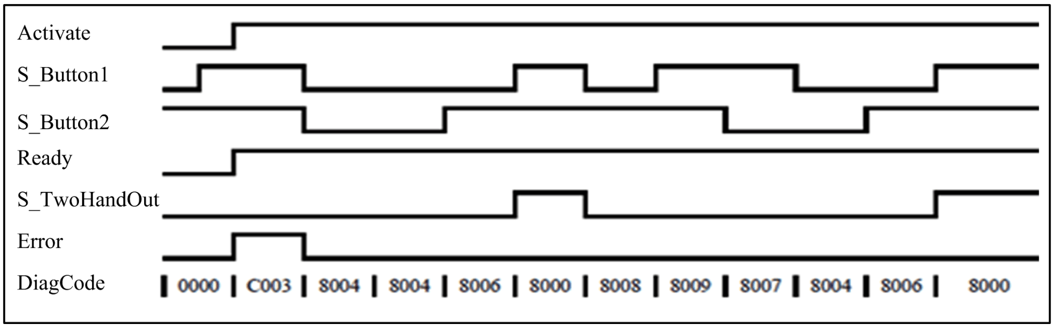
Input Parameters

Parameter	Data type	Init Value	Meaning
`Activate`	`BOOL`	FALSE	The variable or constant value indicating activation state of the function block. This parameter can be connected to the variable, which represents the status (Active or Not Active) of the safety-related device(s). This causes no irrelevant diagnostic information to be generated if a device is disabled: • When FALSE, the output variables are set to the initial values. • Assign a static TRUE signal if no device is connected.
`S_Button1`	`BOOL`	FALSE	The variable value of input button 1 (for SIL level 3 or 4: two antivalent contacts): • FALSE: Button 1 released. • TRUE: Button 1 actuated.
`S_Button2`	`BOOL`	FALSE	The variable value of input button 2 (for SIL level 3 or 4: two antivalent contacts): • FALSE: Button 2 released. • TRUE: Button 2 actuated.

Output Parameters

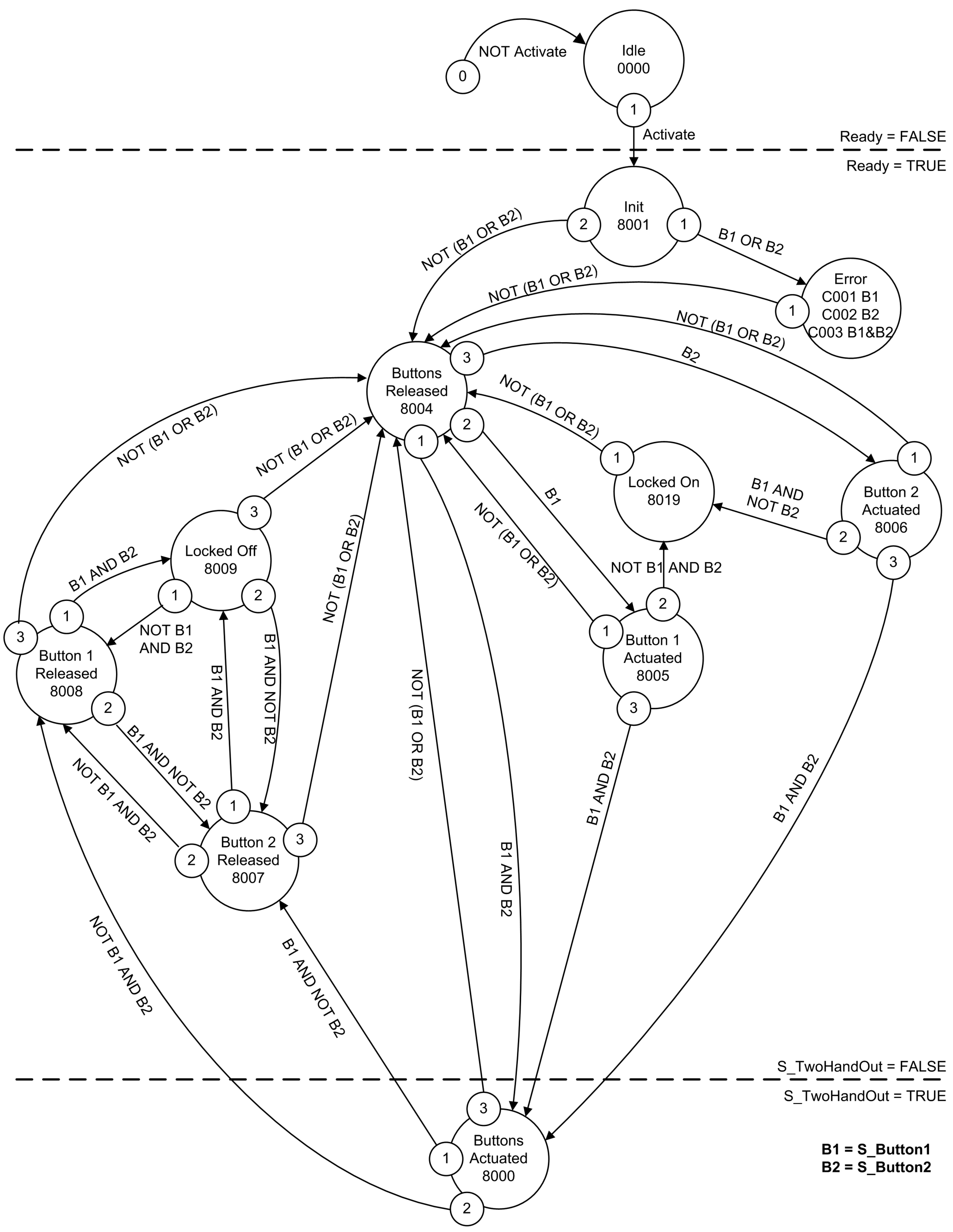
Parameter	Data type	Init Value	Meaning
`Ready`	`BOOL`	FALSE	• TRUE indicates that the function block is activated and the output results are valid. • FALSE indicates that the function block is not active and the program is not executed. **NOTE:** This parameter can be useful in debug mode or to activate/deactivate additional function blocks, as well as for further processing in the functional program.
`S_TwoHandOut`	`BOOL`	FALSE	Safety-related output signal: • FALSE: Two-hand operation is not enabled or performed correctly. • TRUE: Both `S_Button1` and `S_Button2` inputs = TRUE; `Error` = FALSE. Two-hand operation has been performed correctly.
`Error`	`BOOL`	FALSE	Function block detected error message.
`DiagCode`	`WORD`	0000 hex	Function block diagnostic code.

Typical Timing Diagrams



State Diagram

The following diagram describes the state transitions of the `S_TWO_HAND_CONTROL_TYPE_II` function block:



Source: *PLCopen - Technical Committee 5, Safety Software, Technical Specification, Part 1: Concepts and Function Blocks, Version 1.0*.

NOTE: The transition to the Idle state from any other state, occurring because Activate = FALSE, is not depicted. Such a transition has the highest priority (0).

Error Detection

Upon activation of the function block, if either `S_Button1` or `S_Button2` is already set to TRUE, the condition is treated as an invalid input setting that causes an error to be detected.

Detected Error Management

When an error is detected, the output `S_TwoHandOut` is set to FALSE. The `DiagCode` parameter can present one of the following detected error values:

DiagCode	State Name	State Description and Output Settings
C001	Error B1	`S_Button1` was TRUE on activation of the function block: • `S_TwoHandOut` = FALSE • `Error` = TRUE
C002	Error B2	`S_Button2` was TRUE on activation of the function block: • `S_TwoHandOut` = FALSE • `Error` = TRUE
C003	Error B1&B2	Both `S_Button1` and `S_Button2` were TRUE on activation of the function block: • `S_TwoHandOut` = FALSE • `Error` = TRUE

The error state is exited when both buttons are released.

Diagnostic Codes Management

When returning a status message, the `Error` parameter is set to FALSE, and the `DiagCode` parameter displays one of the following hexadecimal values:

DiagCode	State Name	State Description and Output Settings
0	IDLE	The function block is not active (initial state): • `S_TwoHandOut` = FALSE • `Error` = FALSE
8000	Buttons Actuated	Both buttons actuated correctly: • `S_TwoHandOut` = TRUE • `Error` = FALSE
8001	INIT	Function block is active in the INIT state: • `S_TwoHandOut` = FALSE • `Error` = FALSE
8004	Buttons Released	No button is actuated: • `S_TwoHandOut` = FALSE • `Error` = FALSE
8005	Button 1 Actuated	Only Button 1 is actuated: • `S_TwoHandOut` = FALSE • `Error` = FALSE
8006	Button 2 Actuated	Only Button 2 is actuated: • `S_TwoHandOut` = FALSE • `Error` = FALSE
8007	Button 2 Released	The safety related output was enabled, but is now disabled. Both `S_Button1` and `S_Button2` were not set to FALSE after disabling the safety related output. In this state, `S_Button1` is TRUE and `S_Button2` is FALSE after disabling the safety related output: • `S_TwoHandOut` = FALSE • `Error` = FALSE
8008	Button 1 Released	The safety related output was enabled, but is now disabled. Both `S_Button1` and `S_Button2` were not set to FALSE after disabling the safety related output. In this state, `S_Button1` is FALSE and `S_Button2` is TRUE after disabling the safety related output: • `S_TwoHandOut` = FALSE • `Error` = FALSE
8009	Locked Off	The safety related output was enabled and is disabled again. Both `S_Button1` and `S_Button2` were not set to FALSE after disabling the safety related output. In this state, `S_Button1` is TRUE and `S_Button2` is TRUE after disabling the safety related output. • `S_TwoHandOut` = FALSE • `Error` = FALSE
8019	Locked On	Incorrect actuation of the buttons. Waiting for release of both buttons. • `S_TwoHandOut` = FALSE • `Error` = FALSE

S_TWO_HAND_CONTROL_TYPE_III: Two Hand Control with Timer

What's in This Chapter

Description 191

Introduction

This chapter describes the `S_TWO_HAND_CONTROL_TYPE_III` block.

Description

Function Description

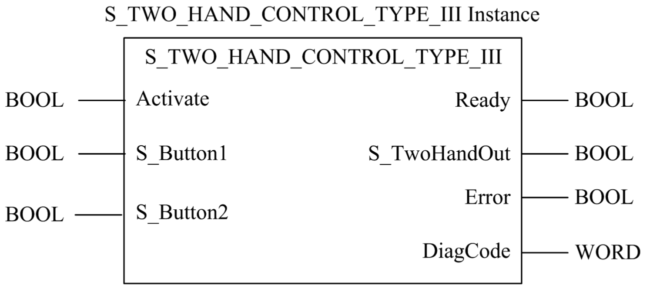
Use the `S_TWO_HAND_CONTROL_TYPE_III` function block to provide two-hand control functionality to a manufacturing process. Two-hand control requires that each hand of the operator be placed on a separate control button, to help prevent potential hazards to the operator.

When both control buttons are actuated (i.e. pressed down), the `S_Button1` and `S_Button2` parameters are set to TRUE. If `S_Button1` and `S_Button2` parameters are set to TRUE within 500 ms of each other and if the `Error` signal remains FALSE, the `S_TwoHandOut` output is set to TRUE.

This function block also controls the release of both buttons before setting the `S_TwoHandOut` output to TRUE.

Representation in FBD

Representation



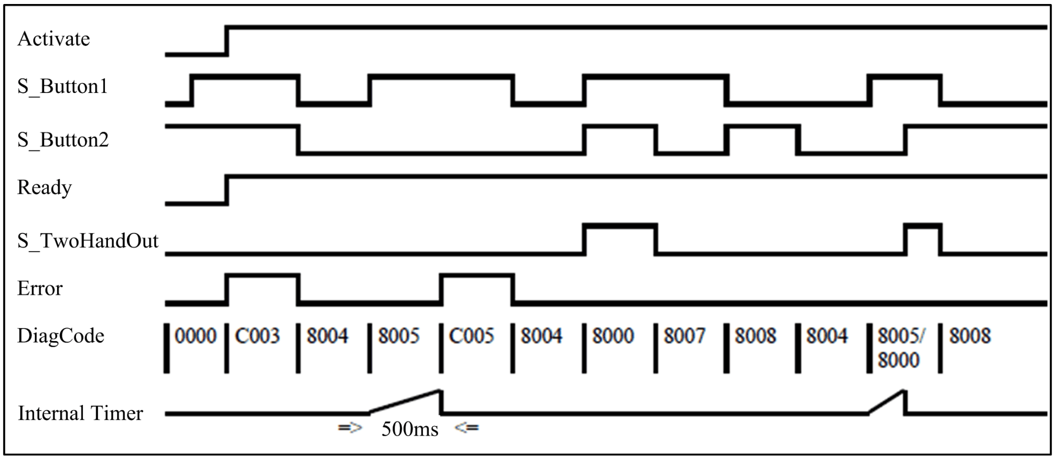
Input Parameters

Parameter	Data type	Init Value	Meaning
`Activate`	`BOOL`	FALSE	The variable or constant value indicating activation state of the function block. This parameter can be connected to the variable, which represents the status (Active or Not Active) of the safety-related device. This causes no irrelevant diagnostic information to be generated if a device is disabled: • When FALSE, the output variables are set to the initial values. • Assign a static TRUE signal if no device is connected.
`S_Button1`	`BOOL`	FALSE	The variable value of input button 1 (for SIL level 3 or 4: two antivalent contacts): • FALSE: Button 1 released. • TRUE: Button 1 actuated.
`S_Button2`	`BOOL`	FALSE	The variable value of input button 2 (for SIL level 3 or 4: two antivalent contacts): • FALSE: Button 2 released. • TRUE: Button 2 actuated.

Output Parameters

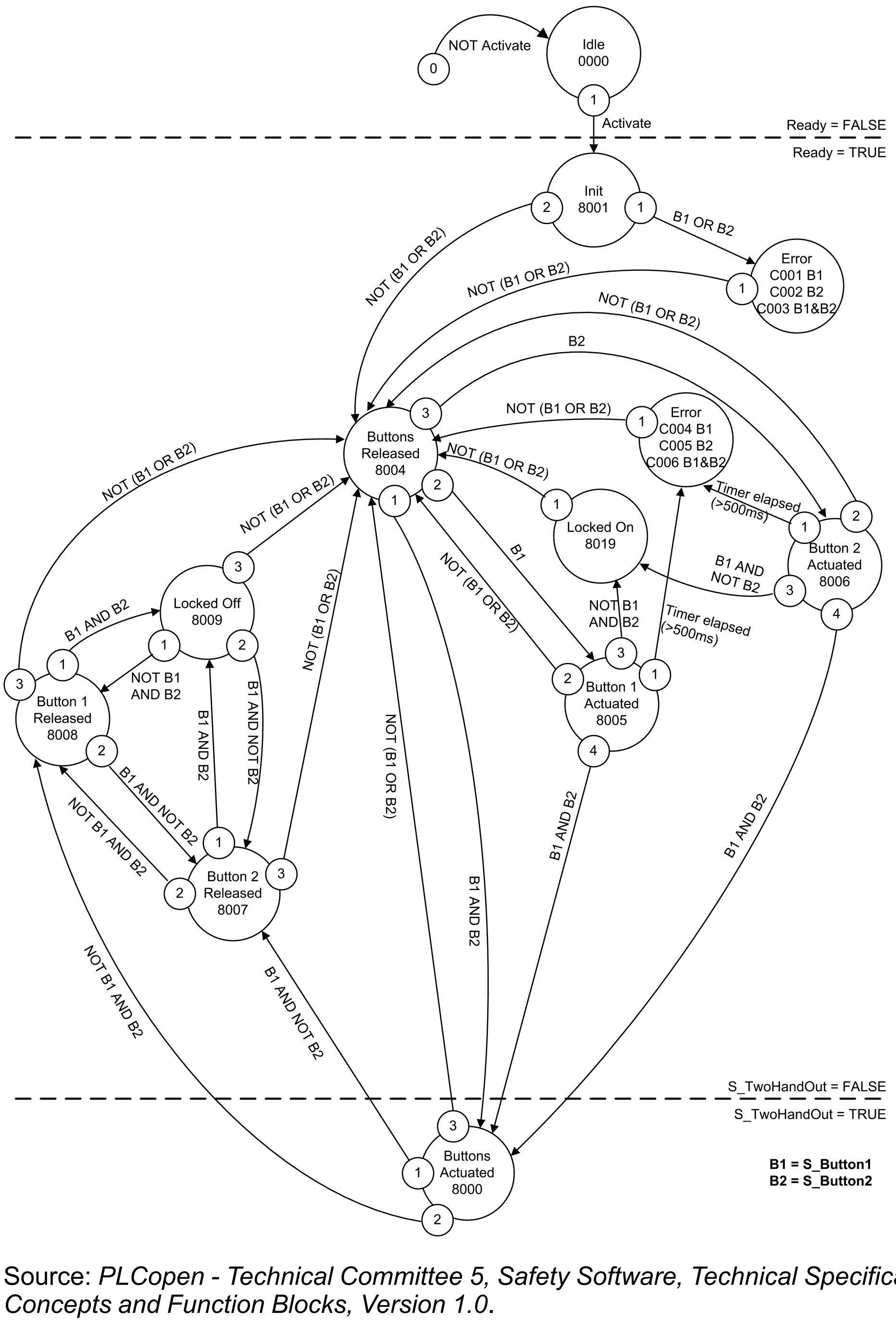
Parameter	Data type	Init Value	Meaning
`Ready`	`BOOL`	FALSE	• TRUE indicates that the function block is activated and the output results are valid. • FALSE indicates that the function block is not active and the program is not executed. **NOTE:** This parameter can be useful in debug mode or to activate/deactivate additional function blocks, as well as for further processing in the functional program.
`S_TwoHandOut`	`BOOL`	FALSE	Safety-related output signal: • FALSE: Two-hand operation is not enabled or performed correctly. • TRUE: Both `S_Button1` and `S_Button2` inputs = TRUE within 500 ms; `Error` = FALSE. Two-hand operation has been performed correctly.
`Error`	`BOOL`	FALSE	Function block detected error message.
`DiagCode`	`WORD`	0000 hex	Function block diagnostic code.

Typical Timing Diagrams



State Diagram

The following diagram describes the state transitions of the `S_TWO_HAND_CONTROL_TYPE_III` function block:



Source: *PLCopen - Technical Committee 5, Safety Software, Technical Specification, Part 1: Concepts and Function Blocks, Version 1.0*.

NOTE: The transition to the Idle state from any other state, occurring because Activate = FALSE, is not depicted. Such a transition has the highest priority (0).

Error Detection

Upon activation of the function block:

- If either S_Button1 or S_Button2 is set to TRUE on block activation, the condition is treated as an invalid input that causes an error to be detected.
- If the elapsed time between the actuation of S_Button1 and S_Button2 is ≥ 500 ms, an error is detected.

Detected Error Management

When an error is detected, the output S_TwoHandOut is set to FALSE. The DiagCode parameter can present one of the following detected error values:

DiagCode	State Name	State Description and Output Settings
C001	Error B1	S_Button1 was TRUE on activation of the function block: • S_TwoHandOut = FALSE • Error = TRUE
C002	Error B2	S_Button2 was TRUE on activation of the function block: • S_TwoHandOut = FALSE • Error = TRUE
C003	Error B1&B2	Both S_Button1 and S_Button2 were TRUE on activation of the function block: • S_TwoHandOut = FALSE • Error = TRUE
C004	Error 2 B1	S_Button1 was FALSE and S_Button2 was TRUE after 500 ms in state 8005: • S_TwoHandOut = FALSE • Error = TRUE

DiagCode	State Name	State Description and Output Settings
C005	Error 2 B2	`S_Button1` was TRUE and `S_Button2` was FALSE after 500 ms in state 8005: • `S_TwoHandOut` = FALSE • `Error` = TRUE
C006	Error 2 B1&B2	Both `S_Button1` and `S_Button2` were TRUE after 500 ms in state 8005 or 8006. This state is possible only if the states of the inputs (`S_Button1` and `S_Button2`) change from divergent to convergent (both TRUE) simultaneously when the timer elapses (500 ms) at the same cycle: • `S_TwoHandOut` = FALSE • `Error` = TRUE

The error state is exited when both buttons are released.

Diagnostic Codes Management

When returning a status message, the `Error` parameter is set to FALSE, and the `DiagCode` parameter displays one of the following hexadecimal values:

DiagCode	State Name	State Description and Output Settings
0	IDLE	The function block is not active (initial state): • `S_TwoHandOut` = FALSE • `Error` = FALSE
8000	Buttons Actuated	Both buttons actuated correctly: • `S_TwoHandOut` = TRUE • `Error` = FALSE
8001	INIT	Function block is active in the INIT state: • `S_TwoHandOut` = FALSE • `Error` = FALSE
8004	Buttons Released	No button is actuated: • `S_TwoHandOut` = FALSE • `Error` = FALSE
8005	Button 1 Actuated	Only Button 1 is actuated. Internal timer starts: • `S_TwoHandOut` = FALSE • `Error` = FALSE
8006	Button 2 Actuated	Only Button 2 is actuated. Internal timer starts: • `S_TwoHandOut` = FALSE • `Error` = FALSE

DiagCode	State Name	State Description and Output Settings
8007	Button 2 Released	The safety related output was enabled, but is now disabled. Both `S_Button1` and `S_Button2` were not set to FALSE after disabling the safety related output. In this state, `S_Button1` is TRUE and `S_Button2` is FALSE after disabling the safety related output: • `S_TwoHandOut` = FALSE • `Error` = FALSE
8008	Button 1 Released	The safety related output was enabled, but is now disabled. Both `S_Button1` and `S_Button2` were not set to FALSE after disabling the safety related output. In this state, `S_Button1` is FALSE and `S_Button2` is TRUE after disabling the safety related output: • `S_TwoHandOut` = FALSE • `Error` = FALSE
8009	Locked Off	The safety related output was enabled and is disabled again. Both `S_Button1` and `S_Button2` were not set to FALSE after disabling the safety related output. In this state, `S_Button1` is TRUE and `S_Button2` is TRUE after disabling the safety related output. • `S_TwoHandOut` = FALSE • `Error` = FALSE
8019	Locked On	Incorrect actuation of the buttons. Waiting for release of both buttons. • `S_TwoHandOut` = FALSE • `Error` = FALSE

High Availability

What’s in This Part

S_AIHA: High Availability for Mx80 Safety Analog Inputs........ 200
S_DIHA: High Availability for Mx80 Safety Digital Inputs 204

Introduction

This section describes the elementary functions and elementary function blocks of the `High Availability` family.

S_AIHA: High Availability for Mx80 Safety Analog Inputs

What's in This Chapter

Description 200

Introduction

This chapter describes the `S_AIHA` block.

Description

Function Description

Use the `S_AIHA` function block in high-availability architectures with redundant BMXSAI0410 safety analog input modules. It continuously compares the integrity of the two channels stemming from the two safety analog input modules and selects the data to be retrieved based on that comparison.
The `S_AIHA` function block is a modifiable template. You can edit the block structure to meet your application requirements. Therefore, it is not certified by the TÜV, or other agency, because it is not static.

⚠WARNING

COMPROMISED SAFETY INTEGRITY LEVEL

You must certify the use of the `S_AIHA` function block by a recognized agency in accordance with IEC 61508 before it is used in a safety-related application.

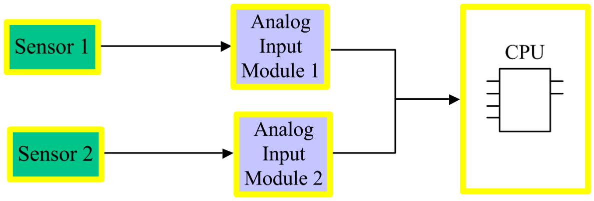
Failure to follow these instructions can result in death, serious injury, or equipment damage.

High-Availability Architecture

To improve availability by using the `S_AIHA` function block, follow these design rules:

- Use two sensors.
- Connect each sensor to a separate input point.
- Each input point should be located on a different analog input module.

Architecture example:

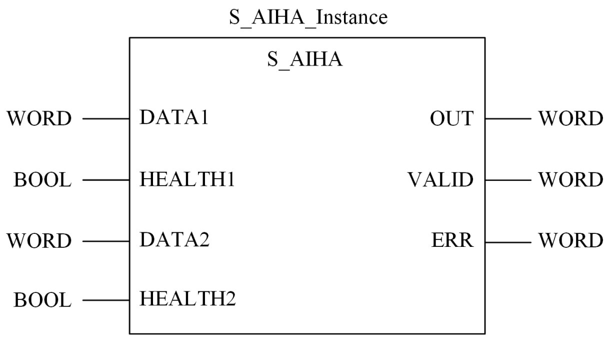


In this design, one safety analog input module processes the signal from Sensor 1, while a second safety analog input module processes the signal from Sensor 2. The controller uses the `S_AIHA` block to select between the data provided by the two channels.

`EN` and `ENO` can be configured as additional parameters.

Representation in FBD

Representation



Input Parameters

The function block retrieves the data and the health information out of each safety analog input module.

Input parameters:

Parameter	Data Type	Meaning
`DATA1`	`WORD`	Data of analog input module 1.
`HEALTH1`	`BOOL`	The health state of the channel to module 1: • 1: Channel is operational. • 0: Channel is not operational.
`DATA2`	`WORD`	Data of analog input module 2.
`HEALTH2`	`BOOL`	The health state of the channel to module 2: • 1: Channel is operational. • 0: Channel is not operational.

Output Parameters

Output parameters:

Parameter	Data Type	Meaning
`OUT`	`WORD`	The output data returned by the `S_AIHA` function block. It will contain the following: • `DATA1` data, if `HEALTH1` = 1. • `DATA2` data, if `HEALTH1` = 0 and `HEALTH2` = 1. • 0 (the defined safe state) if both `HEALTH1` and `HEALTH2` = 0.
`VALID`	`WORD`	Validity of the output data provided by the `OUT` parameter: • 1: Output data is valid. • 0: Output data is not valid. Refer to the State Table, below.
`ERR`	`WORD`	The detected error state of the two safety input modules: • 0: Both modules are OK. • 1: Module 2 is OK; module 1 has a detected error. • 2: Module 1 is OK; module 2 has a detected error. • 3: Errors have been detected in both module 1 and module 2.

State Table

The combinations of parameters of the `S_AIHA` block are explained, below:

HEALTH1	**HEALTH2**	**OUT**	**VALID**	**ERR**	**Remark**
0	0	0	0	Module 1 channel detected error. Module 2 channel detected error.	Defined safe state
0	1	DATA2	1	Module 1 channel detected error.	Use value of module 2 channel.
1	0	DATA1	1	Module 2 channel detected error.	Use value of module 1 channel.
1	1	DATA1	1	Channels on both modules are OK.	Use value of module 1 channel.

S_DIHA: High Availability for Mx80 Safety Digital Inputs

What's in This Chapter

Description 204

Introduction

This chapter describes the `S_DIHA` block.

Description

Function Description

Use the `S_DIHA` function block in high-availability architectures with redundant BMXSDI1602 safety digital input modules. It continuously compares the integrity of the two channels stemming from the two safety digital input modules and selects the data to be retrieved based on that comparison.

The `S_DIHA` function block is a modifiable template. You can edit the block structure to meet your application requirements. Therefore, it is not certified by the TÜV, or other agency, because it is not static.

⚠WARNING

COMPROMISED SAFETY INTEGRITY LEVEL

You must certify the use of the `S_DIHA` function block by a recognized agency in accordance with IEC 61508 before it is used in a safety-related application.

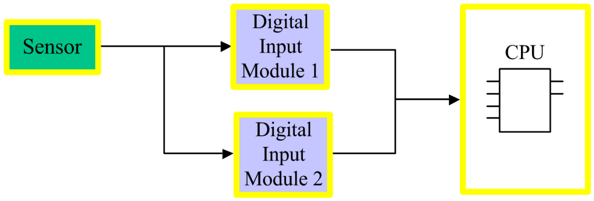
Failure to follow these instructions can result in death, serious injury, or equipment damage.

High-Availability Architecture

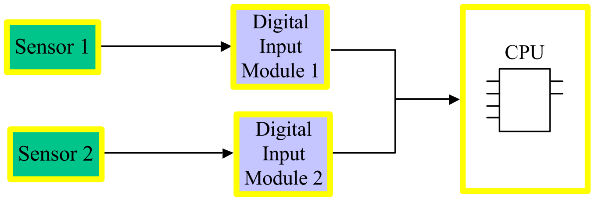
To improve availability by using the `S_DIHA` function block, follow these design rules:

- Use either one or two sensors.
- Use two separate input points:
  - If using one sensor, connect the sensor to both input points.
  - If using two sensors, connect each sensor to a different input point.
- Each input point should be located on a different digital input module.

Architecture example using a single sensor:



Architecture example using two sensors:



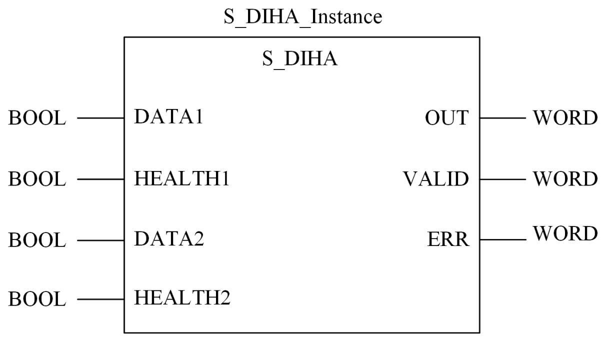
In these designs, either one or two sensors is used. The two input points are located on different digital input modules, which processes the signal received from the connected sensor. The controller uses the `S_DIHA` block to select between the data provided by the two channels.

> **NOTE:** If a single sensor is used, the two digital input modules can share the same process power supply.

`EN` and `ENO` can be configured as additional parameters.

Representation in FBD

Representation



Input Parameters

The function block retrieves the data and the health information out of each safety digital input module.

Input parameters:

Parameter	Data Type	Meaning
DATA1	BOOL	Data of digital input module 1.
HEALTH1	BOOL	The health state of the channel to module 1: • 1: Channel is operational. • 0: Channel is not operational.
DATA2	BOOL	Data of digital input module 2.
HEALTH2	BOOL	The health state of the channel to module 2: • 1: Channel is operational. • 0: Channel is not operational.

Output Parameters

Output parameters:

Parameter	Data Type	Meaning
OUT	BOOL	The output data returned by the S_DIHA function block. It will contain the following: • DATA1 data, if HEALTH1 = 1. • DATA2 data, if HEALTH1 = 0 and HEALTH2 = 1. • 0 (the defined safe state) if both HEALTH1 and HEALTH2 = 0.
VALID	WORD	Validity of the output data provided by the OUT parameter: • 1: Output data is valid. • 0: Output data is not valid. Refer to the State Table, below.
ERR	WORD	The detected error state of the two safety input modules: • 0: Both modules are OK. • 1: Module 2 is OK; module 1 has a detected error. • 2: Module 1 is OK; module 2 has a detected error. • 3: Errors have been detected in both module 1 and module 2. • 4: A discrepancy of input data exists. • 5: A discrepancy of input data exists, and module 1 has a detected error. • 6: A discrepancy of input data exists, and module 2 has a detected error. • 7: A discrepancy of input data exists, and both module 1 and module 2 have detected errors.

State Table

The combinations of parameters of the S_DIHA block are explained, below:

DATA1	HEALTH1	DATA2	HEALTH2	OUT	VALID	ERR	Remark
0	0	0	0	0	0	Module 1 + Module 2 detected error.	Defined safe state.(1)
0	0	0	1	0	1	Module 1 detected error.	Use Module 2 value.(2)
0	0	1	0	0	0	Module 1 + Module 2 detected error.	Defined safe state.(1)
0	0	1	1	1	1	Module 1 detected error.	Use Module 2 value.(2)
0	1	0	0	0	1	Module 2 detected error.	Use Module 1 value.(2)
0	1	0	1	0	1	OK	Consistent values.(3)

DATA1	HEALTH1	DATA2	HEALTH2	OUT	VALID	ERR	Remark
0	1	1	0	0	1	Module 2 detected error.	Use Module 1 value.(2)
0	1	1	1	0	1	Discrepancy	Use Module 1 value.(3)
1	0	0	0	0	0	Module 1 + Module 2 detected error.	Defined safe state.(1)
1	0	0	1	0	1	Module 1 detected error.	Use Module 2 value.(2)
1	0	1	0	0	0	Module 1 + Module 2 detected error.	Defined safe state.(1)
1	0	1	1	1	1	Module 1 detected error.	Use Module 2 value.(2)
1	1	0	0	1	1	Module 2 detected error.	Use Module 1 value.(2)
1	1	0	1	1	1	Discrepancy	Use Module 1 value.(3)
1	1	1	0	1	1	Module 2 detected error.	Use Module 1 value.(2)
1	1	1	1	1	1	OK	Consistent. (3)

(1) The value of the input is set to 0 (defined safe state) because both modules have a detected error. The controller is still running using 0 for the input. Repair or replace the modules.

(2) An error has been detected in one of the modules. The value from the other module is used. Repair or replace the module with the detected error.

(3) Both modules are healthy. If a discrepancy is detected, this condition can be handled programmatically, if desired.

Logic

What's in This Part

S_AND_***: AND Function 210
S_F_TRIG: Falling Edge Detection 212
S_NOT_***: Negation 214
S_OR_***: OR Function 216
S_R_TRIG: Rising Edge Detection 218
S_ROL_***: Rotate Left 220
S_ROR_***: Rotate Right 223
S_RS: Bistable Function Block, Reset Dominant 226
S_SHL_***: Shift Left 228
S_SHR_***: Shift Right 231
S_SR: Bistable Function Block, Set Dominant 234
S_XOR_***: Exclusive OR Function 236

Introduction

This section describes the elementary functions and elementary function blocks of the `Logic` family.

S_AND_***: AND Function

What's in This Chapter

Description 210

Introduction

This chapter describes the `S_AND_***` block.

Description

Function Description

The function for a bit-by-bit AND link of the bit sequences at the inputs and assigns the result to the output.

Ensure that the data types of all input values and output values are identical.

The number of inputs can be increased to a maximum of 32.

`EN` and `ENO` can be configured as additional parameters.

Formula

`OUT` = `IN1` & `IN2` & ... & `INn`

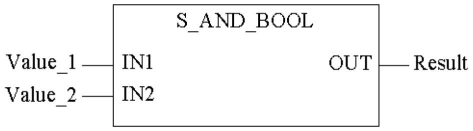
Available Functions

List of available functions

- S_AND_BOOL
- S_AND_BYTE
- S_AND_WORD
- S_AND_DWORD

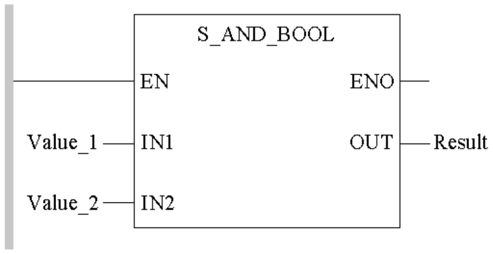
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	**Data Type**	**Meaning**
`IN1`	`BOOL, BYTE, WORD, DWORD`	input bit sequence
`IN2`	`BOOL, BYTE, WORD, DWORD`	input bit sequence
`INn`	`BOOL, BYTE, WORD, DWORD`	input bit sequence (n = maximum 32)

Description of the output parameter

Parameter	**Data Type**	**Meaning**
`OUT`	`BOOL, BYTE, WORD, DWORD`	output bit sequence

S_F_TRIG: Falling Edge Detection

What's in This Chapter

Description 212

Introduction

This chapter describes the `S_F_TRIG` block.

Description

Function Description

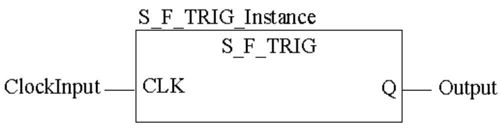
This function block is used for the detection of falling edges 1 -> 0.

Output `Q` becomes 1 if there is a transition from 1 to 0 at the `CLK` input. The output will remain at 1 for one function block execution cycle; the output subsequently returns to 0.

`EN` and `ENO` can be configured as additional parameters.

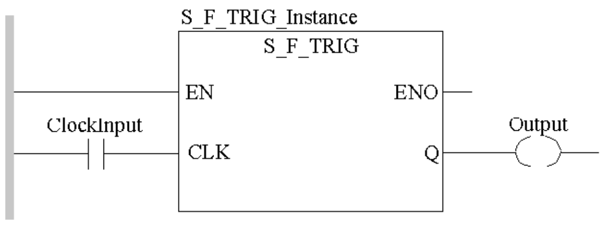
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
CLK	BOOL	clock input

Description of the output parameter

Parameter	Data Type	Meaning
Q	BOOL	output

S_NOT_***: Negation

What's in This Chapter

Description 214

Introduction

This chapter describes the `S_NOT_***` block.

Description

Function Description

The function negates the input bit sequence bit-by-bit and assigns the result to the output.

Ensure that the data types of all input values and output values are identical.

`EN` and `ENO` can be configured as additional parameters.

Formula

`OUT` = NOT `IN`

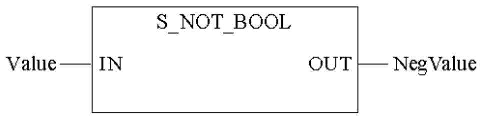
Available Functions

List of available functions

- S_NOT_BOOL
- S_NOT_BYTE
- S_NOT_WORD
- S_NOT_DWORD

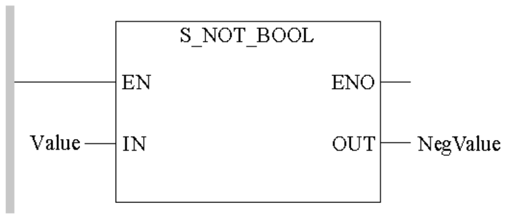
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	**Data Type**	**Meaning**
`IN`	`BOOL`, `BYTE`, `WORD`, `DWORD`	input bit sequence

Description of the output parameter

Parameter	**Data Type**	**Meaning**
`OUT`	`BOOL`, `BYTE`, `WORD`, `DWORD`	negated bit sequence

S_OR_***: OR Function

What’s in This Chapter

Description 216

Introduction

This chapter describes the `S_OR_***` block.

Description

Function Description

The function performs a logical OR operation between each input and the next input until all inputs have been combined using the OR operation. .

Ensure that the data types of all input values and output values are identical.

The number of inputs can be increased to a maximum of 32.

`EN` and `ENO` can be configured as additional parameters.

Formula

`OUT` = `IN1` OR `IN2` OR ... OR `INn`

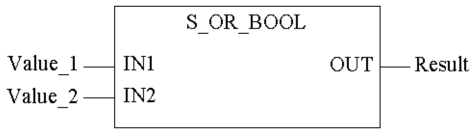
Available Functions

List of available functions

- S_OR_BOOL
- S_OR_BYTE
- S_OR_WORD
- S_OR_DWORD

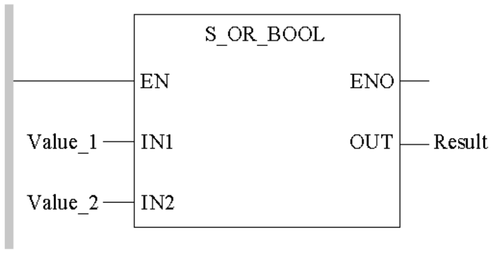
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
IN1	BOOL, BYTE, WORD, DWORD	input bit sequence
IN2	BOOL, BYTE, WORD, DWORD	input bit sequence
INn	BOOL, BYTE, WORD, DWORD	input bit sequence n = maximum 32

Description of the output parameter

Parameter	Data Type	Meaning
OUT	BOOL, BYTE, WORD, DWORD	output bit sequence

S_R_TRIG: Rising Edge Detection

What's in This Chapter

Description 218

Introduction

This chapter describes the `S_R_TRIG` block.

Description

Function Description

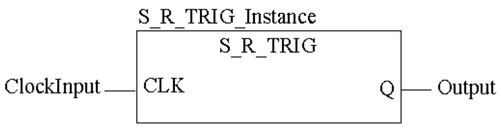
This function block is used for the detection of rising edges 0 -> 1.

Output `Q` becomes 1 if there is a transition from 0 to 1 at the `CLK` input. The output remains at 1 for one function block execution cycle; the output subsequently returns to 0.

`EN` and `ENO` can be configured as additional parameters.

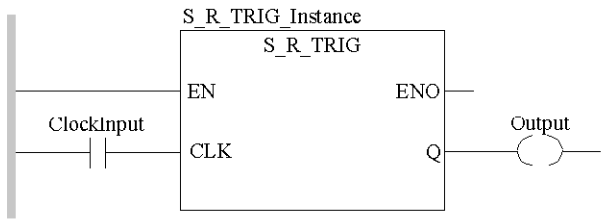
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
`CLK`	`BOOL`	clock input

Description of the output parameter

Parameter	Data Type	Meaning
`Q`	`BOOL`	output

S_ROL_***: Rotate Left

What's in This Chapter

Description 220

Introduction

This chapter describes the S_ROL_*** block.

Description

Function Description

This function rotates the bit pattern at the IN input circularly to the left by n bits (value at input Number).

System bit %S17 is used as CARRY bit; this means that the status of the bit that is shifted out is stored there.

Ensure that the data types of the IN input and OUT output are identical.

EN and ENO can be configured as additional parameters.

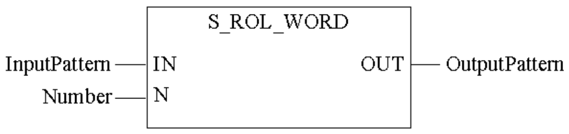
Available Functions

List of available functions

- S_ROL_BOOL
- S_ROL_BYTE
- S_ROL_WORD
- S_ROL_DWORD

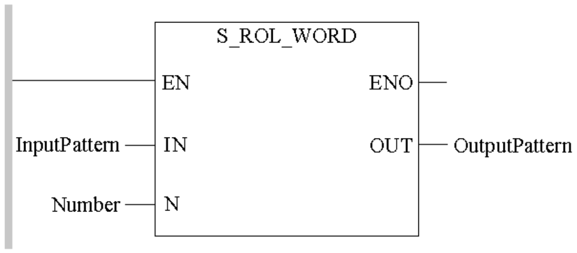
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
IN	BOOL, BYTE, WORD, DWORD	This is the bit pattern to be rotated. For example: InputPattern=2#0100000011110001
N	UINT	This is the number of spaces to be rotated. Example: number = 4

Description of the output parameter

Parameter	Data Type	Meaning
OUT	BOOL, BYTE, WORD, DWORD	This is the rotated bit pattern. For example: With the data from the previous table, the result is: RotatedPattern=2#0000111100010100

Function Block Behavior

The maximum number of rotation should be less than or equal to the size of the operand:

- for BYTE, the maximum number of rotations is 8.
- for WORD and INT, the maximum number of rotations is 16.
- for DWORD and DINT, the maximum number of rotations is 32.

The table below gives the output value of the rotate function block depending on the number of rotations and the size of the operand:

Type	Number of rotations	Output value	%S17
BYTE	0	= Input value	0
	1...8	= Rotated input value	MSB
	9...32	= Incorrect value	- (1)
	>32	= 0	0
WORD/INT	0	= Input value	0
	1...16	= Rotated input value	MSB
	17...31	= Incorrect value	- (1)
	>32	= 0	0
DWORD/DINT	0	= Input value	0
	1...32	= Rotated input value	MSB
	>32	= 0	0
(1) undefined			

S_ROR_***: Rotate Right

What's in This Chapter

Description 223

Introduction

This chapter describes the `S_ROR_***` block.

Description

Function Description

This function rotates the bit pattern at the `In` input circularly to the right by n bits (value at input `Number`).

System bit %S17 is used as CARRY bit, this means that the status of the bit that is shifted out is stored there.

Ensure that the data types of the `IN` input and `OUT` output are identical.

`EN` and `ENO` can be configured as additional parameters.

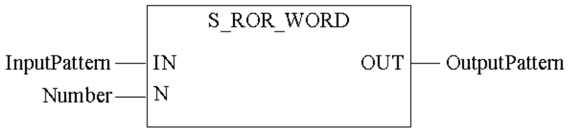
Available Functions

List of available functions

- S_ROR_BOOL
- S_ROR_BYTE
- S_ROR_WORD
- S_ROR_DWORD

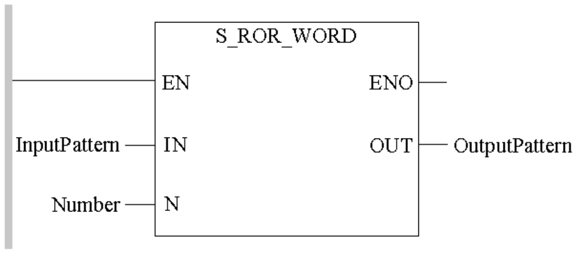
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
IN	BOOL, BYTE, WORD, DWORD	This is the bit pattern to be rotated. For example: InputPattern=2#0100000011110001
N	UINT	This is the number of spaces to be rotated. Example: number = 4

Description of the output parameter

Parameter	Data Type	Meaning
`OUT`	`BOOL`, `BYTE`, `WORD`, `DWORD`	This is the rotated bit pattern. For example: With the data from the previous table, the result is RotatedPattern=2#0001010000001111

S_RS: Bistable Function Block, Reset Dominant

What's in This Chapter

Description 226

Introduction

This chapter describes the `S_RS` block.

Description

Function Description

The function block is used as `RS` memory with the property "Reset dominant".

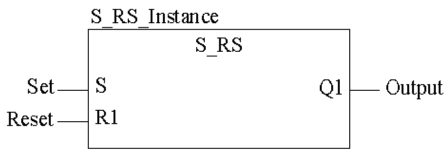
Output `Q1` becomes 1 when the `S` input becomes 1. This state remains even if input `S` reverts back to 0. Output `Q1` changes back to 0 when input `R1` becomes 1. If the inputs `S` and `R1` are 1 simultaneously, the dominating input `R1` will set the output `Q1` to 0.

When the function block is called for the first time, the initial state of `Q1` is 0.

`EN` and `ENO` can be configured as additional parameters.

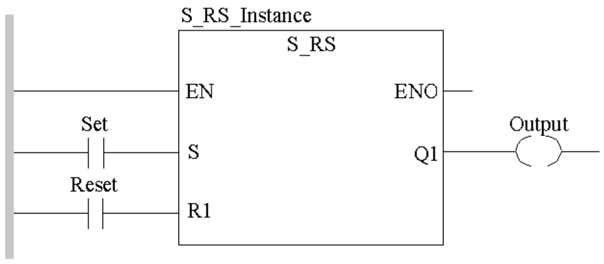
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	**Data Type**	**Meaning**
S	BOOL	set
R1	BOOL	reset (dominant)

Description of the output parameter

Parameter	**Data Type**	**Meaning**
Q1	BOOL	output

S_SHL_***: Shift Left

What's in This Chapter

Description 228

Introduction

This chapter describes the `S_SHL_***` block.

Description

Function Description

This function shifts the bit pattern at the `IN` input to the left by n bits (value at input `N`).

System bit %S17 is used as CARRY bit, this means that the status of the bit that is shifted out is stored there.

Zeros are filled in from the right.

Ensure that the data types of the `IN` input and `OUT` output are identical.

`EN` and `ENO` can be configured as additional parameters.

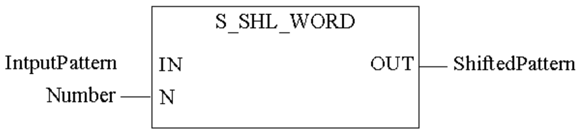
Available Functions

List of available functions

- S_SHL_BOOL
- S_SHL_BYTE
- S_SHL_WORD
- S_SHL_DWORD

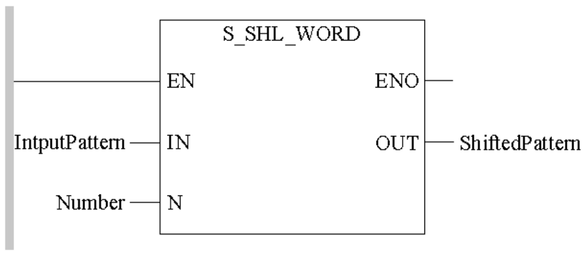
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
`IN`	`BOOL`, `BYTE`, `WORD`, `DWORD`	This is the bit pattern to be shifted. **For example** `IntputPattern` = 2#010000001111**0001**.
`N`	`UINT`	This is the number of spaces to be shifted. **For example** Number = 4.

Description of the output parameter

Parameter	Data Type	Meaning
`OUT`	`BOOL`, `BYTE`, `WORD`, `DWORD`	This is the bit pattern to be shifted. **For example** With the data from the previous table, the result is ShiftedPattern = 2#000011110001**0000**

S_SHR_***: Shift Right

What's in This Chapter

Description 231

Introduction

This chapter describes the S_SHR_*** block.

Description

Function Description

This function shifts the bit pattern at the IN input to the right by n bits (value at input N).

System bit %S17 is used as CARRY bit, this means that the status of the bit that is shifted out is stored there.

Zeros are filled in from the left.

Ensure that the data types of the IN input and OUT output are identical.

EN and ENO can be configured as additional parameters.

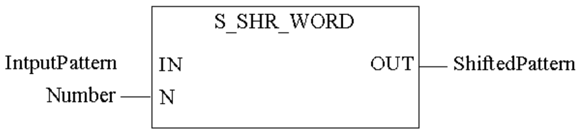
Available Functions

List of available functions

- S_SHR_BOOL
- S_SHR_BYTE
- S_SHR_WORD
- S_SHR_DWORD

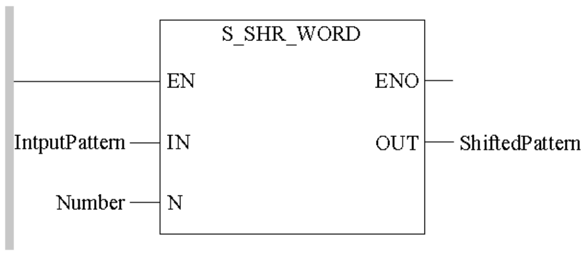
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
`IN`	`BOOL`, `BYTE`, `WORD`, `DWORD`	This is the bit pattern to be shifted. **For example** `IntputPattern` = 2#010000000111**0001**.
`N`	`UINT`	This is the number of spaces to be shifted. **Example:** number = 4.

Description of the output parameter

Parameter	Data Type	Meaning
OUT	BOOL, BYTE, WORD, DWORD	This is the bit pattern to be shifted. **For example** With the data from the previous table, the result is ShiftedPattern = 2#**0000**010000001111

S_SR: Bistable Function Block, Set Dominant

What's in This Chapter

Description 234

Introduction

This chapter describes the S_SR block.

Description

Function Description

The function block is used as SR memory with the property "Set dominant".

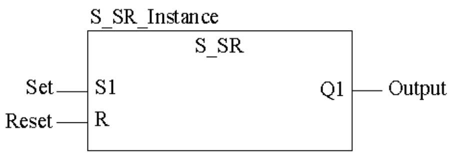
Output Q1 becomes 1 when the S1 input becomes 1. This state remains even if input S1 reverts back to 0. Output Q1 changes back to 0 when input R becomes 1. If the inputs S1 and R are both 1 simultaneously, the dominating input S1 will set the output Q1 to 1.

When the function block is called for the first time, the initial state of Q1 is 0.

EN and ENO can be configured as additional parameters.

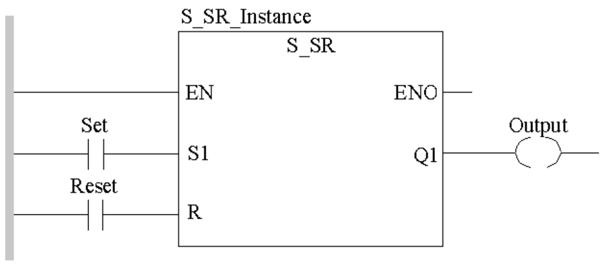
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
S1	BOOL	set (dominant)
R	BOOL	reset

Description of the output parameter

Parameter	Data Type	Meaning
Q1	BOOL	output

S_XOR_***: Exclusive OR Function

What's in This Chapter

Description 236

Introduction

This chapter describes the S_XOR_*** block.

Description

Function Description

The function for a bit XOR is to sequence the bits at the inputs and return the result at the output.

Ensure that the data types of all input values and output values are identical.

The number of inputs can be increased to a maximum of 32.

EN and ENO can be configured as additional parameters.

Available Functions

List of available functions

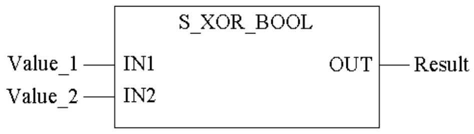
- S_XOR_BOOL
- S_XOR_BYTE
- S_XOR_WORD
- S_XOR_DWORD

Formula

OUT = IN1 XOR IN2 XOR ... XOR INn

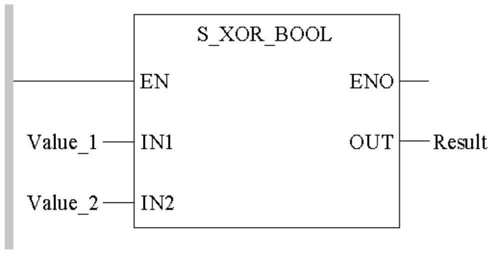
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
IN1	BOOL, BYTE, WORD, DWORD	input bit sequence
IN2	BOOL, BYTE, WORD, DWORD	input bit sequence
INn	BOOL, BYTE, WORD, DWORD	input bit sequence n = maximum 32

Description of the output parameter

Parameter	Data Type	Meaning
OUT	BOOL, BYTE, WORD, DWORD	output bit sequence

Mathematics

What's in This Part

S_ABS_***: Absolute Value Computation....239
S_ADD_***: Addition....242
S_DIV_***: Division....245
S_MUL_***: Multiplication....248
S_MOVE: Assignment....251
S_NEG_***: Negation....253
S_SQRT_REAL: Safety Square Root....256
S_SIGN_***: Sign Evaluation....258
S_SUB_***: Subtraction....261

Introduction

This section describes the elementary functions and elementary function blocks of the `Mathematics` family.

S_ABS_***: Absolute Value Computation

What's in This Chapter

Description 239

Introduction

This chapter describes the `S_ABS_***` block.

Description

Function Description

The function computes the absolute value of the input value and assigns the result to the output.

Ensure that the data types of the input and output values are identical.

`EN` and `ENO` can be configured as additional parameters.

Formula

$$OUT = |IN|$$

Available Functions

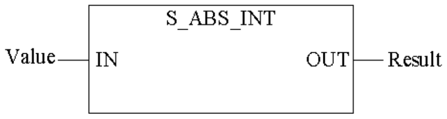
List of available functions

- S_ABS_INT
- S_ABS_DINT
- S_ABS_UINT
- S_ABS_UDINT

- S_ABS_REAL

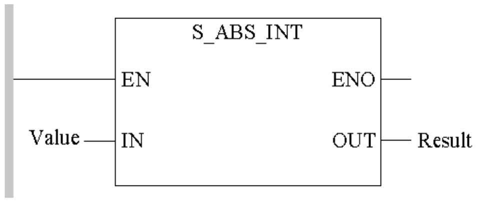
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
IN	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	input value

Description of the output parameter

Parameter	Data Type	Meaning
`OUT`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	output value

Runtime Error

The system bit %S18, page 352 is set to 1, if a value is below a limit value (data types `INT` and `DINT`).

S_ADD_***: Addition

What's in This Chapter

Description 242

Introduction

This chapter describes the `S_ADD_***` block.

Description

Function Description

The function adds the input values and assigns the result to the output.

Ensure that the data types of all input values and output values are identical.

The number of inputs can be increased to a maximum of 32 for all functions.

`EN` and `ENO` can be configured as additional parameters.

Formula

`INT`, `DINT`, `UINT`, `UDINT`, `REAL`:

`OUT = IN1 + IN2 + ... + INn`

Available Functions

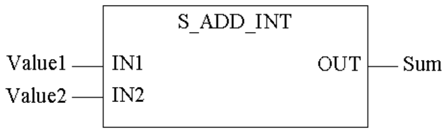
List of available functions

- S_ADD_INT
- S_ADD_DINT
- S_ADD_UINT
- S_ADD_UDINT

- S_ADD_REAL

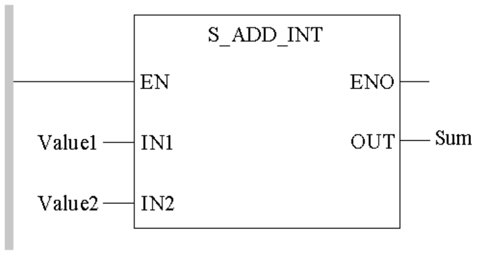
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
IN1	INT, DINT, UINT, UDINT, REAL	summand
IN2	INT, DINT, UINT, UDINT, REAL	summand
INn	INT, DINT, UINT, UDINT, REAL	summand n = maximum 32

Description of the output parameter

Parameter	Data Type	Meaning
OUT	INT, DINT, UINT, UDINT, REAL	sum

Runtime Error

The system bit %S18, page 352 is set to 1, if the value range on the output is exceeded (all available data types).

NOTE: The system word %SW17 (see EcoStruxure™ Control Expert, System Bits and Words, Reference Manual) gives the detected error status on S_ADD_REAL.

S_DIV_***: Division

What's in This Chapter

Description 245

Introduction

This chapter describes the `S_DIV_***` block.

Description

Function Description

The function divides the value at the `Dividend` with the value at the `Divisor` input and assigns the result to the output.

The data types of the input values and the output values must be identical.

When dividing `INT`, `DINT`, `UINT` and `UDINT` data types, any decimal places in the result are omitted, however, when dividing `REAL` data type any decimal places in the result appear.

$$7 \div 3 = 2$$

$$(-7) \div 3 = -2$$

`EN` and `ENO` can be configured as additional parameters.

Formula

$$OUT = ((IN1) \div (IN2))$$

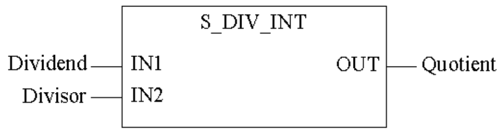
Available Functions

List of available functions

- S_DIV_INT
- S_DIV_DINT
- S_DIV_UINT
- S_DIV_UDINT
- S_DIV_REAL

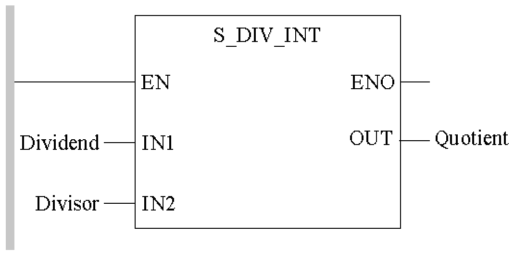
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
IN1	INT, DINT, UINT, UDINT, REAL	dividend
IN2	INT, DINT, UINT, UDINT, REAL	divisor

Description of the output parameter

Parameter	Data Type	Meaning
OUT	INT, DINT, UINT, UDINT, REAL	quotient

Runtime Error

The system bit %S18, page 352 is set to 1, if an invalid division by 0 is executed (all available data types).

NOTE: The system word %SW17 (see EcoStruxure™ Control Expert, System Bits and Words, Reference Manual) gives the detected error status on S_DIV_REAL.

S_MUL_***: Multiplication

What's in This Chapter

Description 248

Introduction

This chapter describes the S_MUL_*** block.

Description

Function Description

The function multiplies the input values and assigns the result to the output.

Ensure that the data types of all input values and output values are identical.

The number of inputs can be increased to a maximum of 32.

EN and ENO can be configured as additional parameters.

Formula

OUT = IN1 x IN2 x ... x IN$_n$

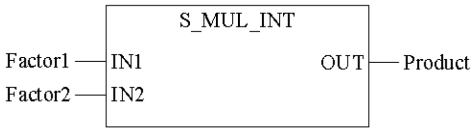
Available Functions

List of available functions

- S_MUL_INT
- S_MUL_DINT
- S_MUL_UINT
- S_MUL_UDINT
- S_MUL_REAL

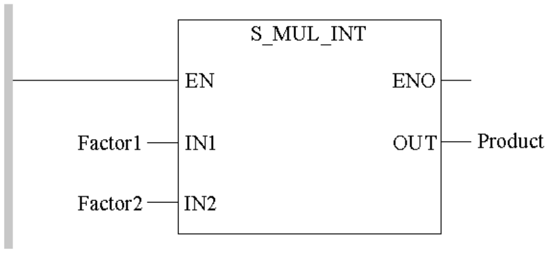
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	**Data Type**	**Meaning**
`IN1`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	multiplicand (factor)
`IN2`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	multiplier (factor)
`INn`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	multiplier (factor) n = maximum 32

Description of the output parameter

Parameter	**Data Type**	**Meaning**
`OUT`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	product

Runtime Error

The system bit %S18, page 352 is set to 1, if the value range at the output has been exceeded (all available data types).

NOTE: The system word %SW17 (see EcoStruxure™ Control Expert, System Bits and Words, Reference Manual) gives the detected error status on S_MUL_REAL.

S_MOVE: Assignment

What's in This Chapter

Description 251

Introduction

This chapter describes the `S_MOVE` block.

Description

Function Description

The function assigns the input value to the output.

This is a generic function. The data type to be processed will be determined by the input to the function.

If a direct address of a variable is to be assigned or vice versa, assign the variable to the function first. A direct address at input and output of the function is not authorized since this does not allow a clear definition of the data type.

Ensure that the data types of the input and output values are identical.

`EN` and `ENO` can be configured as additional parameters.

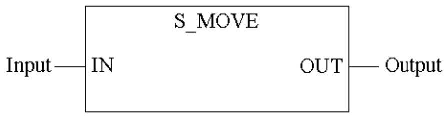
NOTE: It is not possible to copy an array of `EBOOL` elements using the `S_MOVE` function, since the `S_MOVE` function does not update the assignment history of the array elements.

Formula

`OUT` = `IN`

Representation in FBD

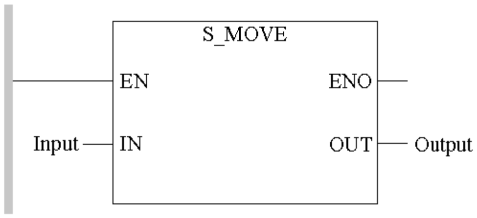
Representation



Representation in LD

This function cannot be used in the LD (Ladder Diagram) programming language with the `BOOL` data type, since the same functionality can be achieved there with contacts and coils.

Representation



Parameter Description

Description of the input parameter:

Parameter	Data Type	Meaning
`IN`	`ANY`	input value

Description of the output parameter:

Parameter	Data Type	Meaning
`OUT`	`ANY`	output value

S_NEG_***: Negation

What's in This Chapter

Description 253

Introduction

This chapter describes the `S_NEG_***` block.

Description

Function Description

The function negates the input value and delivers the result at the `OUT` output.

The negation causes a sign reversal, for example:

6 becomes -6

-4 becomes 4

Ensure that the data types of the input and output values are identical.

`EN` and `ENO` can be configured as additional parameters.

NOTE: When the `INT` and `DINT` data types are processed, it is not possible to convert long negative values into positive ones. However, the `ENO` output is not set to 0 when this error is detected.

NOTE: When the `UINT` and `UDINT` data types are processed, a detected error message is returned in a system bit (%S18, page 352).

Available Functions

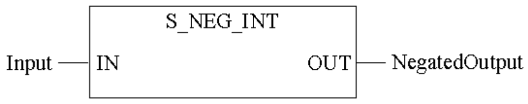
List of available functions

- S_NEG_INT
- S_NEG_DINT
- S_NEG_UINT

- S_NEG_UDINT
- S_NEG_REAL

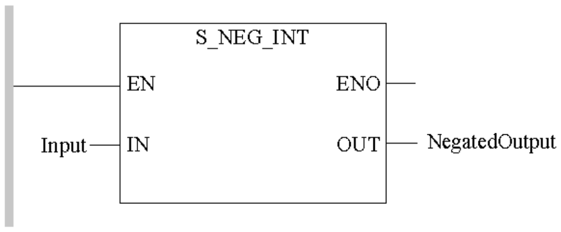
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of input parameter:

Parameter	Data Type	Meaning
`IN`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	input value (`Input`)

Description of output parameters

Parameter	Data Type	Meaning
`OUT`	`INT`, `DINT`, `UINT`, `UDINT`, `REAL`	negated output value (`NegatedOutput`)

Runtime Error

The system bit %S18, page 352 is set to 1 if

- a violation of the value range at the input occurs during the execution of the function (data types `INT` and `DINT`)

  or
- an input value of the data type `UINT` or `UDINT` is to be converted.

In case of %S18, page 352 = 1 the system word %SW17 (see EcoStruxure™ Control Expert, System Bits and Words, Reference Manual) indicates the type of detected error.

S_SQRT_REAL: Safety Square Root

What's in This Chapter

Description 256

Introduction

This chapter describes the `S_SQRT_REAL` block

Description

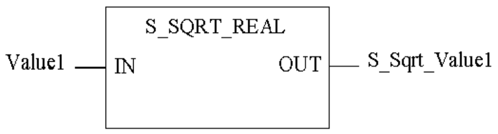
Function description

The S_SQRT_REAL function extracts the square root from a variable. This function can be called using its generic name.

The additional parameters `EN` and `ENO` can be configured.

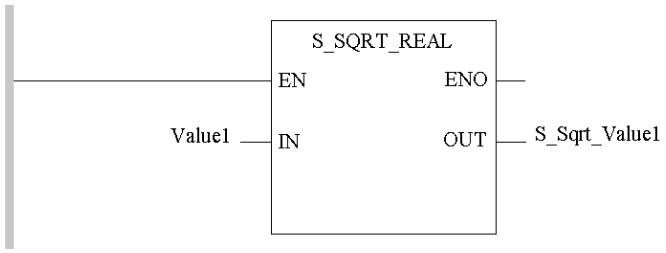
Representation in FBD

Representation applied to a `REAL`:



Representation in LD

Representation applied to a `REAL`:



Description of parameters

The following table describes the input parameter:

Parameter	Type	Comment
`IN`	`REAL`.	Variable whose square root you want to extract. The input value must be greater than or equal to zero.

The following table describes the output parameter:

Parameter	Type	Comment
`OUT`	`REAL`.	`S_Sqrt_Value1` contains the square root of `Value1`. `S_Sqrt_Value1` is of the same type as `Value1`.

Runtime errors

When `Value1` is of `REAL` type and negative, the result of the function contains NAN and bit %S18, page 352 = 1.

In case of %S18, page 352 = 1 the system word %SW17 (see EcoStruxure™ Control Expert, System Bits and Words, Reference Manual) indicates the type of detected error.

S_SIGN_***: Sign Evaluation

What's in This Chapter

Description 258

Introduction

This chapter describes the S_SIGN_*** block.

Description

Function Description

The function is used for the detection of negative signs.

With a value ≥ 0 at the input, the output becomes 0. With a value < 0 at the input, the output becomes 1.

NOTE: Because of IEC 61131-3 conformity, this function also works with the UINT and UDINT data types. This is not significant since these functions return a 0 result.

EN and ENO can be configured as additional parameters.

Formula

Block formula

OUT = 1, if IN < 0

OUT = 0, if IN ≥ 0

NOTE: Be aware of the following behavior results for signed 0 (+/-0):

- -0 -> SIGN_INT/DINT -> 0
- +0 -> SIGN_INT/DINT -> 0

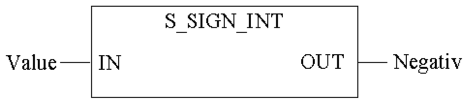
Available Functions

List of available functions

- S_SIGN_INT
- S_SIGN_DINT
- S_SIGN_UINT
- S_SIGN_UDINT
- S_SIGN_REAL

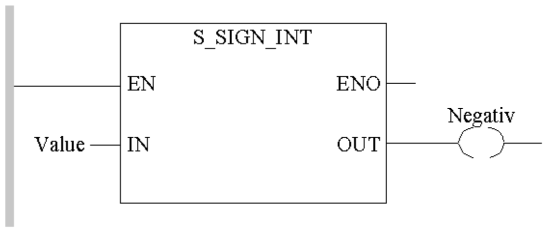
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of input parameter

Parameter	Data Type	Meaning
IN	INT, DINT, UINT, UDINT, REAL	signed input

Description of output parameter

Parameter	Data Type	Meaning
OUT	BOOL	sign evaluation

Runtime Error

The system bit %S18, page 352 is set to 1 and ENO to 0 if an input value of the data type UINT or UDINT is given.

In case of %S18, page 352 = 1 the system word %SW17 (see EcoStruxure™ Control Expert, System Bits and Words, Reference Manual) indicates the type of detected error.

S_SUB_***: Subtraction

What's in This Chapter

Description 261

Introduction

This chapter describes the `S_SUB_***` block.

Description

Function Description

The function subtracts the value at the `Value2` input from the value at the `Value1` input and assigns the result to the output.

Ensure that the data types of the input values and the output values are identical.

`EN` and `ENO` can be configured as additional parameters.

Formula

`Difference` = `Value1` - `Value2`

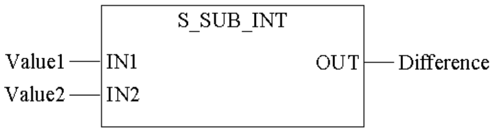
Available Functions

List of available functions

- S_SUB_INT
- S_SUB_DINT
- S_SUB_UINT
- S_SUB_UDINT
- S_SUB_REAL

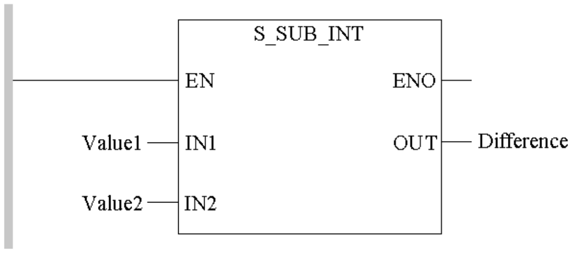
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
IN1	INT, DINT, UINT, UDINT, REAL	minuend
IN2	INT, DINT, UINT, UDINT, REAL	subtrahend

Description of the output parameter

Parameter	Data Type	Meaning
OUT	INT, DINT, UINT, UDINT, REAL	difference

Runtime Error

The system word %SW17 (see EcoStruxure™ Control Expert, System Bits and Words, Reference Manual) gives the detected error status on S_SUB_REAL.

In case of %S18, page 352 = 1 the system word %SW17 (see EcoStruxure™ Control Expert, System Bits and Words, Reference Manual) indicates the type of detected error.

Statistical

What’s in This Part

S_LIMIT_***: Limit 265
S_MAX_***: Maximum Value Function 268
S_MIN_***: Minimum Value Function 271
S_MUX_***: Multiplexer 274
S_SEL: Binary Selection 277

Introduction

This section describes the elementary functions and elementary function blocks of the `Statistical` group.

S_LIMIT_***: Limit

What's in This Chapter

Description 265

Introduction

This chapter describes the S_LIMIT_*** block.

Description

Function Description

This function transfers the unchanged input value (Input) to the output if the input value is not less than the minimum value (LowerLimit) and does not exceed the maximum value (UpperLimit). If the input value (Input) is less than the minimum value (LowerLimit), the minimum value will be transferred to the output. If the input value (Input) exceeds the maximum value (UpperLimit), the maximum value will be transferred to the output.

Ensure that the data types of all input values and output values are identical.

EN and ENO can be configured as additional parameters.

Formula

OUT = IN, if (IN ≥ MN) & (IN ≤ MX)

OUT = MN, if (IN < MN)

OUT = MX, if (IN > MX)

Available Functions

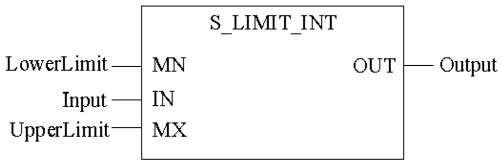
List of available functions

- S_LIMIT_BOOL

- S_LIMIT_BYTE
- S_LIMIT_WORD
- S_LIMIT_DWORD
- S_LIMIT_INT
- S_LIMIT_DINT
- S_LIMIT_UINT
- S_LIMIT_UDINT

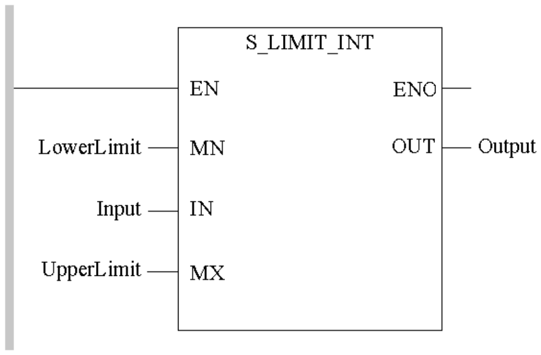
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
MN	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	lower limit
IN	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	input
MX	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	upper limit

Description of the output parameter

Parameter	Data Type	Meaning
OUT	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	output

S_MAX_***: Maximum Value Function

What's in This Chapter

Description 268

Introduction

This chapter describes the `S_MAX_***` block.

Description

Function Description

The function assigns the largest input value to the output.

Ensure that the data types of all input values and output values are identical.

The number of inputs can be increased to maximum 32.

`EN` and `ENO` can be configured as additional parameters.

Formula

`OUT` = `MAX` {`IN1, IN2, ..., INn`}

Available Functions

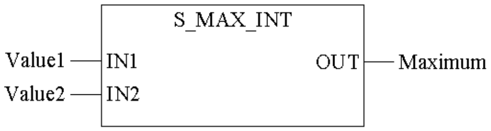
List of available functions

- S_MAX_BOOL
- S_MAX_BYTE
- S_MAX_WORD
- S_MAX_DWORD
- S_MAX_INT

- S_MAX_DINT
- S_MAX_UINT
- S_MAX_UDINT

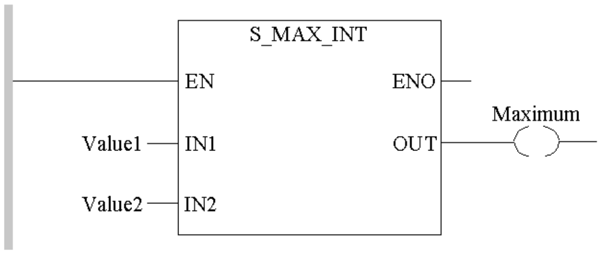
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of input parameters

Parameter	Data Type	Meaning
IN1	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	Input 1 value
IN2	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	Input 2 value
INn	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	Input n value n = maximum 32

Description of output parameter

Parameter	Data Type	Meaning
OUT	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	maximum value

S_MIN_***: Minimum Value Function

What’s in This Chapter

Description 271

Introduction

This chapter describes the `S_MIN_***` block.

Description

Function Description

The function assigns the smallest input value to the output.

Ensure that the data types of all input values and output values are identical.

The number of inputs can be increased to maximum 32.

`EN` and `ENO` can be configured as additional parameters.

Formula

`OUT` = `MIN` {`IN1, IN2, ..., INn`}

Available Functions

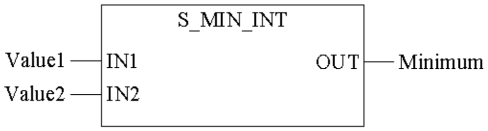
List of available functions

- S_MIN_BOOL
- S_MIN_BYTE
- S_MIN_WORD
- S_MIN_DWORD
- S_MIN_INT

- S_MIN_DINT
- S_MIN_UINT
- S_MIN_UDINT

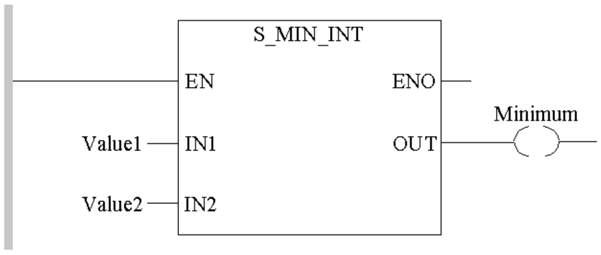
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of input parameters

Parameter	Data Type	Meaning
IN1	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	Input 1 value
IN2	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	Input 2 value
INn	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	Input n value n = maximum 32

Description of output parameter

Parameter	Data Type	Meaning
Minimum	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	minimum value

S_MUX_***: Multiplexer

What's in This Chapter

Description 274

Introduction

This chapter describes the S_MUX_*** block.

Description

Function Description

This function transfers the respective input to the output depending on the value at the K input.

The number of inputs can be increased to maximum 31.

EN and ENO can be configured as additional parameters.

Example

K = 0: Input IN0 is transferred to the output

K = 1: Input IN1 is transferred to the output

K = 5: Input IN5 is transferred to the output

K= n: Input INn is transferred to the output

Data Types

Ensure that the data types at the inputs Input0 to Inputn and at the output are identical.

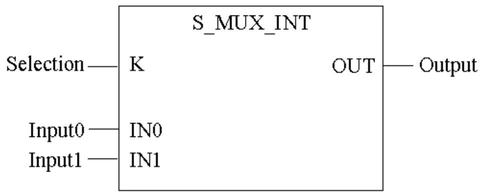
Available Functions

List of available functions

- S_MUX_INT
- S_MUX_DINT
- S_MUX_UINT
- S_MUX_UDINT

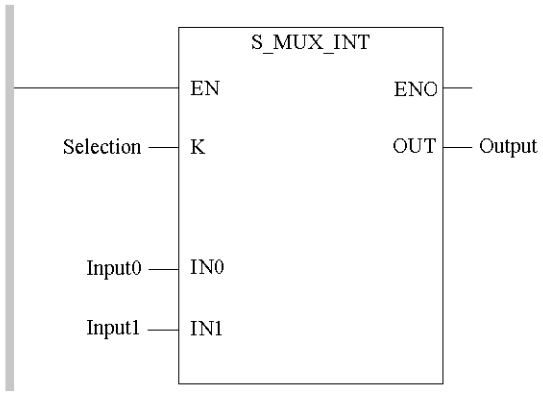
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of input parameters

Parameter	Data Type	Meaning
K	INT, DINT, UINT, UDINT	selection input K = 0...30
IN0	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	Input 1 value
IN1	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	Input 2 value
IN2	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	Input 3 value
INn	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	Input n value, n = maximum 30

Description of output parameter

Parameter	Data Type	Meaning
OUT	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	output

S_SEL: Binary Selection

What's in This Chapter

Description 277

Introduction

This chapter describes the `S_SEL` block.

Description

Function Description

The function is used for binary selection between 2 input values.

Depending on the state of the `Selection` input, either the `Input0` input or `Input1` input is transferred to the `Output` output.

`Selection` = 0 -> `Output` = `Input0`

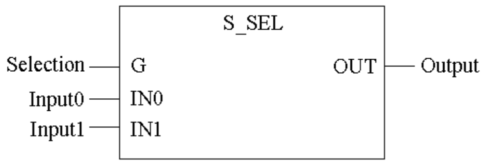
`Selection` = 1 -> `Output` = `Input1`

Ensure that the data types of the `Input0` and `Input1` input values and the `Output` output values are identical.

`EN` and `ENO` can be configured as additional parameters.

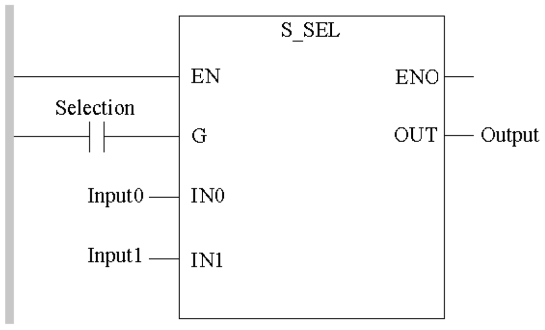
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
G	BOOL	selection input
IN0	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	Input 0 value
IN1	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	Input 1 value

Description of the output parameter

Parameter	Data Type	Meaning
OUT	BOOL, BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	output

System

What's in This Part

S_SYST_CLOCK_MX: System Clock.....280
S_SYST_READ_TASK_BIT_MX: System Task Bit.....283
S_SYST_RESET_TASK_BIT_MX: Reset Task.....285
S_SYST_STAT_MX: System State.....287
S_SYST_TIME_MX: System Time.....289

Introduction

This section describes the elementary functions and elementary function blocks of the `System` family.

S_SYST_CLOCK_MX: System Clock

What's in This Chapter

Description 280

Introduction

This chapter describes the `S_SYST_CLOCK_MX` block.

Description

Function Description

The `S_SYST_CLOCK_MX` function is only available for Safety controller with firmware 3.10 or earlier.

Use the `S_SYST_CLOCK_MX` function to return the value of safe time clock.

The safe time clock value is frozen during the execution of the SAFE task cycle.

The safe time clock can be optionally synchronized by an external NTP server. In this case, an NTP status is provided. If no NTP server is identified in the configuration, the safe time is synchronized by the RTC.

NOTE: The time received from the NTP server, or from the RTC, is used to synchronize the safe time clock under certain conditions. If the difference between the safe time clock and the NTP server or RTC is:

- Less than or equal to 2 seconds: a progressive update of the safe time clock is performed at the rate of 1ms per second (max 2000 s of catch-up time).
- Greater than 2 seconds: the safe time clock is maintained locally and a status of the detected synchronization error is returned.

The application can trigger a non-conditional update of safe time clock by applying a fixed value to system word %SW128.

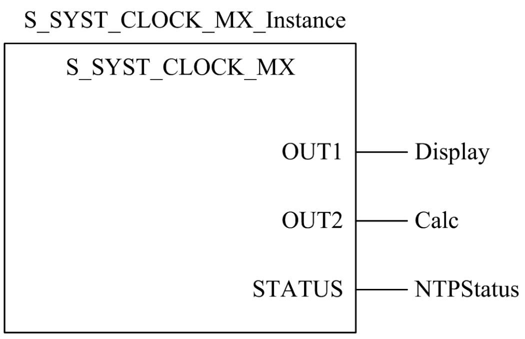
The clock value is given in 2 different formats (`S_Calc_Time` and `S_Display_Time`, as with other system clock functions.

The function has no input parameter.

`EN` and `ENO` can be configured as additional parameters.

Representation in FBD

Representation



Parameter Description

The `S_SYST_CLOCK_MX` function consists of the following output parameters:

- `OUT1`
- `OUT2`
- `STATUS`

The `S_SYST_CLOCK_MX` function includes no input parameters.

Output Parameters

Parameter	Data Type	Meaning
`OUT1`	`S_Display_Time`[1]	Structure containing a DT element and a millisecond counter
`OUT2`	`S_Calc_Time`[2]	Structure containing a clock second counter, elapsed since January 1, 1980, at 00:00, and a millisecond counter.

Parameter	Data Type	Meaning
STATUS	INT	BIT0 : is set to: • 1 if safe time is valid. • 0 if safe time is not valid. **NOTE:** If safe time is not valid, it may be due to an invalid NTP status (if controller is configured as NTP client) and/or to the fact that controller time has changed by more than 2 seconds. Use %SW128 to force the synchronization of the safe time with internal controller time if required.

1. S_Display_Time is a structure comprising a DT type element and an INT type element:
   - DT_value containing the date.
   - Milisecond containing the number of milliseconds of this date.
2. S_Calc_Time is a predefined structure comprising a UDINT type element and an INT type element:
   - Calc.Seconds containing the number of seconds elapsed since January 1, 1900, at 00:00.
   - Calc.Fraction_Second containing the number of milliseconds to add for the precision of the result to be around a millisecond.

S_SYST_READ_TASK_BIT_MX: System Task Bit

What's in This Chapter

Description 283

Introduction

This chapter describes the `S_SYST_READ_TASK BIT_MX` block.

Description

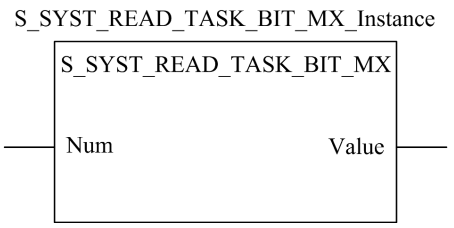
Function Description

Use the `S_SYST_READ_TASK_BIT_MX` function to read and return the value of one of the following specified system task bits: %S17, %S18, or %S21.

`EN` and `ENO` can be configured as additional parameters.

Representation in FBD

Representation



Input Parameters

Parameter	Data Type	Meaning
`Num`	`INT`	Number of the system bit for the specified task: %S17, %S18, or % S21.

Output Parameters

Parameter	Data Type	Meaning
`Value`	`BOOL`	Value of the system bit.
`ENO`	`BOOL`	If mapped, displays false (0) if an invalid `Num` value is configured.

S_SYST_RESET_TASK_BIT_MX: Reset Task

What's in This Chapter

Description 285

Introduction

This chapter describes the `S_SYST_RESET_TASK_BIT_MX` block.

Description

Function Description

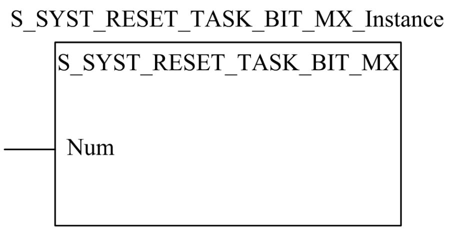
Use the `S_SYST_RESET_TASK_BIT_MX` function to set to a value of 0 to one of the following specified system task bits: %S17, %S18, or %S21.

`EN` and `ENO` can be configured as additional parameters.

NOTE: Enable the `EN, page 22` parameter only when it is required to acknowledge the information set by the corresponding %S, which has been set to a value of 1 by the system.

Representation in FBD

Representation



Input Parameters

Parameter	Data Type	Meaning
`Num`	`INT`	Number of the system bit for the specified task to be reset: %S17, %S18, or %S21.

Output Parameters

Parameter	Data Type	Meaning
`ENO`	`BOOL`	If mapped, displays false (0) if an invalid `Num` value is configured.

S_SYST_STAT_MX: System State

What's in This Chapter

Description 287

Introduction

This chapter describes the `S_SYST_STAT_MX` block.

Description

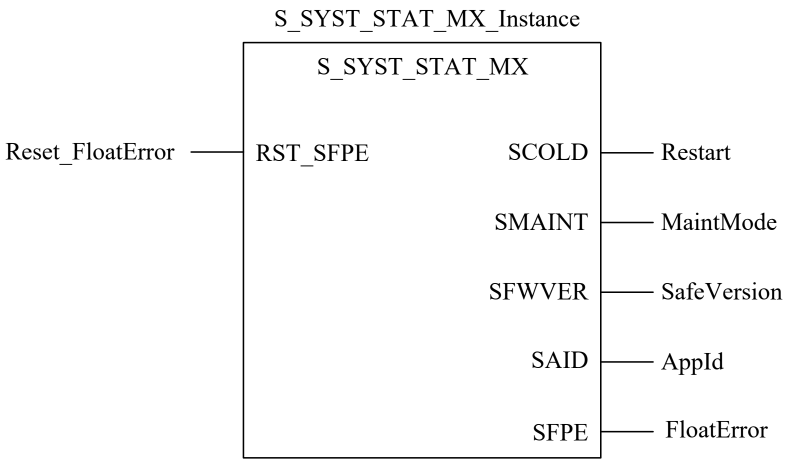
Function Description

Use the `S_SYST_STAT_MX` function to read and return information about the state of the safety-related application.

`EN` and `ENO` can be configured as additional parameters.

Representation in FBD

Representation



Input Parameters

Parameter	Data Type	Meaning
RST_SFPE	BOOL	If the value is set to: • 1: The value for the last floating point error detected by the system on safety-related sections is written to the SFPE output parameter, then cleared by resetting the detected error to 0. • 0: The value for the last floating point error detected by the system on safety-related sections is not written to the SFPE output parameter, and the detected error is not reset to 0.

Output Parameters

Parameter	Data Type	Meaning
SCOLD	BOOL	Restart. Value set to 1 during first SAFE Task cycle after initialization of safe variables.
SMAINT	BOOL	MaintMode. Value set to 1 when the controller is in maintenance mode.
SFWVER	INT	SafeVersion. Contains in binary coded decimal (BCD) format the Safety CoPro firmware version, for example, 16#0102 for V1.2.
SAID	INT	AppId. Contains an ID of the safety portion of the application: %SW169. If you use the **Build** menu command **Build Changes** or **Rebuild All Project**, this ID will be modified only if the safety portion of the application has been changed.
SFPE	INT	Stores the System Floating Point Error (SFPE) value that was most recently detected in a SAFE Task. When a system floating point error is detected, system bit %S18 is set to 1. The SFPE parameter can contain the following values: • Bit 0: Invalid operation detected. The result is not a number. • Bit 1: Denormalized operand detected. Result is acceptable. • Bit 2: Division by 0. Result = infinity. • Bit 3: Overflow. Result = infinity. • Bit 4: Underflow. Result = 0. • All other bits are reserved.

S_SYST_TIME_MX: System Time

What's in This Chapter

Description 289

Introduction

This chapter describes the `S_SYST_TIME_MX` block.

Description

Function Description

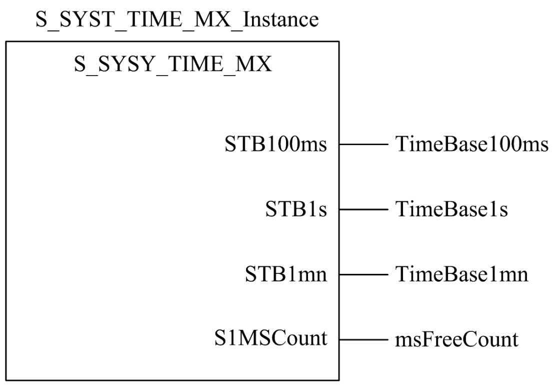
Use the `S_SYST_TIME_MX` function to return the time elapsed since the completion of the most recent SAFE Task cycle.

NOTE: The time values are frozen during execution of the SAFE Task cycle. Calls to this function made at different times during execution of the same SAFE task cycle return the same values.

`EN` and `ENO` can be configured as additional parameters.

Representation in FBD

Representation



Parameter Description

The `S_SYST_TIME_MX` function consists of the following parameters:

Output parameters:

- `STB100ms`
- `STB1s`
- `STB1mn`
- `S1MSCount`

This function has no input parameters.

Output Parameters

Parameter	Data Type	Meaning
`STB100ms`	`BOOL`	TimeBase100ms. This bit toggles every 50 ms.
`STB1s`	`BOOL`	TimeBase1s. This bit toggles every 500 ms.
`STB1mn`	`BOOL`	TimeBase1mn. This bit toggles every 30 s.
`S1MSCount`	`DINT`	msFreeCount. Free running counter with 1 ms increment.

Timers & Counters

What's in This Part

S_CTD_***: Down Counter 292
S_CTU_***: Up Counter 295
S_CTUD_***: Up/Down Counter 298
S_TOF: Off Delay 301
S_TON: On Delay 304
S_TP: Pulse 307

Introduction

This section describes the elementary functions and elementary function blocks of the `Timers & Counters` family.

S_CTD_***: Down Counter

What's in This Chapter

Description 292

Introduction

This chapter describes the `S_CTD_***` blocks.

Description

Function Description

The function block is used for downwards counting.

A 1 signal at the `LD` input causes the value of the `PV` input to be allocated to the `CV` output. With each transition from 0 to 1 at the `CD` input, the value of `CV` is reduced by 1.

When `CV` ≤ 0, the `Q` output becomes 1.

NOTE: The counter only works to the minimum values of the data type being used. No overflow occurs.

`EN` and `ENO` can be configured as additional parameters.

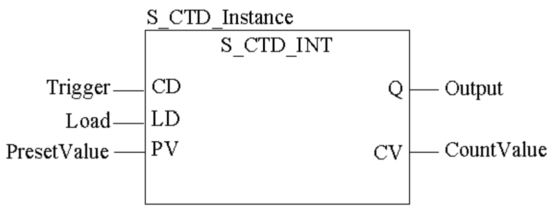
Available Functions

List of available functions

- S_CTD_INT
- S_CTD_DINT
- S_CTD_UINT
- S_CTD_UDINT

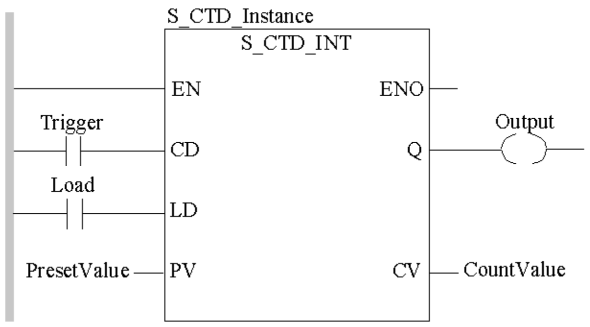
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
CD	BOOL	trigger input
LD	BOOL	load data
PV	INT, DINT, UINT, UDINT	preset value

Description of the output parameter

Parameter	Data Type	Meaning
Q	BOOL	output
CV	INT, DINT, UINT, UDINT	count value (actual value)

S_CTU_***: Up Counter

What's in This Chapter

Description 295

Introduction

This chapter describes the `S_CTU_***` block.

Description

Function Description

The function block is used for upwards counting.

A 1 signal at the `R` input causes the value 0 to be assigned to the `CV` output. With each transition from 0 to 1 at the `CU` input, the value of `CV` is incremented by 1. When `CV` ≥ `PV`, the `Q` output is set to 1.

> **NOTE:** The counter only works to the maximum values of the data type being used. No overflow occurs.

`EN` and `ENO` can be configured as additional parameters.

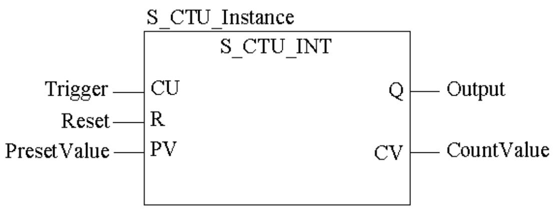
Available Functions

List of available functions

- S_CTU_INT
- S_CTU_DINT
- S_CTU_UINT
- S_CTU_UDINT

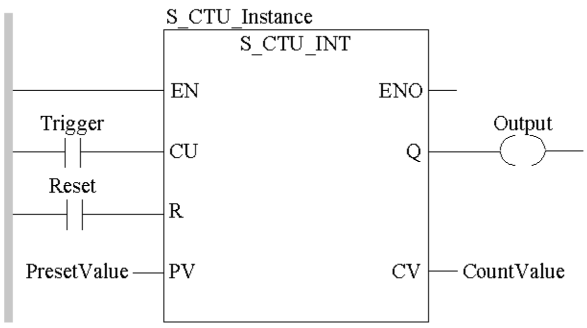
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
CU	BOOL	trigger input
R	BOOL	reset
PV	INT, DINT, UINT, UDINT	preset value

Description of the output parameter

Parameter	Data Type	Meaning
Q	BOOL	output
CV	INT, DINT, UINT, UDINT	count value (actual value)

S_CTUD_***: Up/Down Counter

What's in This Chapter

Description 298

Introduction

This chapter describes the S_CTUD_*** block.

Description

Function Description

The function block is used for upwards and downwards counting.

A 1 signal at the R input causes the value 0 to be assigned to the CV output. A 1 signal at the LD input causes the value of the PV input to be allocated to the CV output. With each transition from 0 to 1 at the CU input, the value of CV is incremented by 1. With each transition from 0 to 1 at the CD input, the value of CV is reduced by 1.

If there is a simultaneous 1 signal at inputs R and LD, input R has precedence.

When CV ≥ PV, output QU becomes 1.

When CV ≤ 0, the QD output becomes 1.

NOTE: The down counter only works to the minimum values of the data type being used, and the up counter only to the maximum values of the data type being used. No overflow occurs.

EN and ENO can be configured as additional parameters.

Available Functions

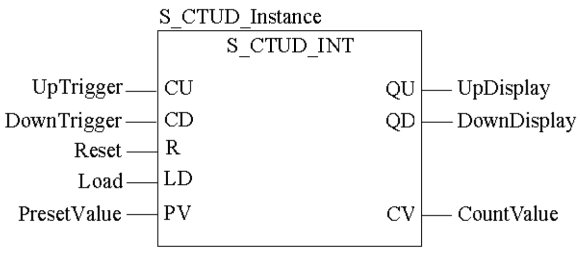
List of available functions

- S_CTUD_INT
- S_CTUD_DINT
- S_CTUD_UINT

- S_CTUD_UDINT

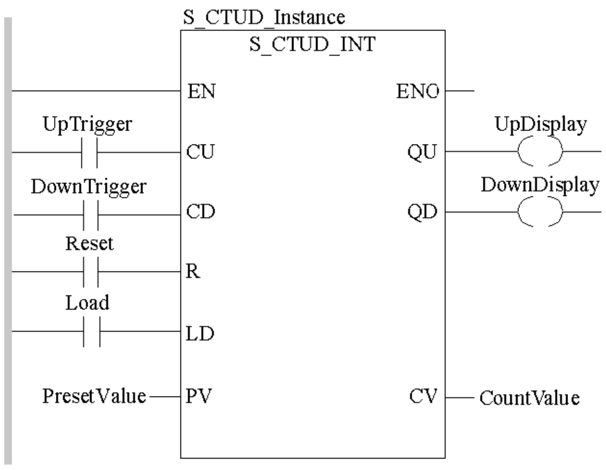
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
CU	BOOL	up counter trigger input
CD	BOOL	down counter trigger input
R	BOOL	reset
LD	BOOL	load data
PV	INT, DINT, UINT, UDINT	preset value

Description of the output parameter

Parameter	Data Type	Meaning
QU	BOOL	up display
QD	BOOL	down display
CV	INT, DINT, UINT, UDINT	count value (actual value)

S_TOF: Off Delay

What's in This Chapter

Description 301

Introduction

This chapter describes the `S_TOF` block.

Description

Function Description

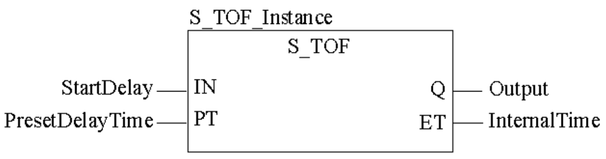
The function block is used as the Off delay.

When the function block is called for the first time, the initial state of `ET` is 0.

`EN` and `ENO` can be configured as additional parameters.

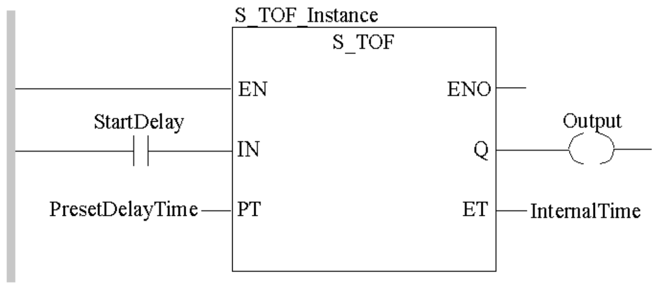
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

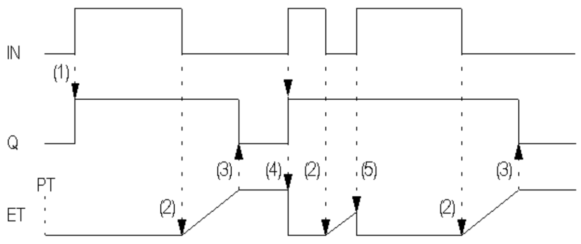
Parameter	Data Type	Meaning
IN	BOOL	start delay
PT	TIME	preset delay time

Description of the output parameter

Parameter	Data Type	Meaning
Q	BOOL	output
ET	TIME	internal time

Timing Diagram

Representation of the OFF delay `S_TOF`



(1) If IN becomes 1, Q becomes 1.

(2) If IN becomes 0, the internal time (ET) is started.

(3) If the internal time reaches the value of PT, Q becomes 0.

(4) If IN becomes 1, Q becomes 1, and the internal time is stopped/reset.

(5) If IN becomes 1 before the internal time has reached the value of PT, the internal time is stopped/reset without Q being set back to 0.

S_TON: On Delay

What's in This Chapter

Description 304

Introduction

This chapter describes the S_TON block.

Description

Function Description

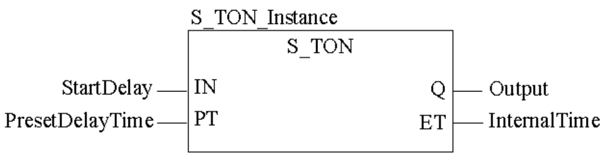
The function block is used as the On delay.

When the function block is called for the first time, the initial state of ET is 0.

EN and ENO can be configured as additional parameters.

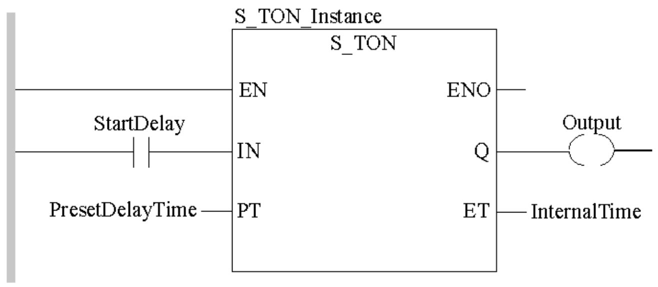
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

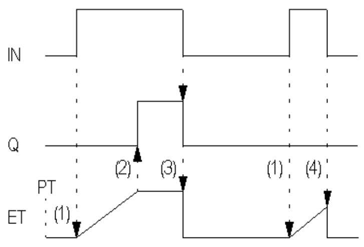
Parameter	Data Type	Meaning
IN	BOOL	start delay
PT	TIME	preset delay time

Description of the output parameter

Parameter	Data Type	Meaning
Q	BOOL	output
ET	TIME	internal time

Timing Diagram

Representation of the ON delay `S_TON`



(1) If IN becomes 1, the internal time (ET) starts.

(2) If the internal time reaches the value of PT, Q becomes 1.

(3) If IN becomes 0, Q becomes 0 and the internal time is stopped/reset.

(4) If IN becomes 0 before the internal time has reached the value of PT, the internal time stops/resets without Q going to 1.

S_TP: Pulse

What's in This Chapter

Description 307

Introduction

This chapter describes the `S_TP` block.

Description

Function Description

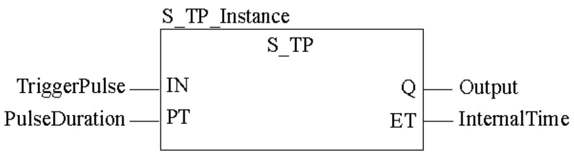
The function block is used for the generation of a pulse with defined duration.

When the function block is called for the first time, the initial state of `ET` is 0.

`EN` and `ENO` can be configured as additional parameters.

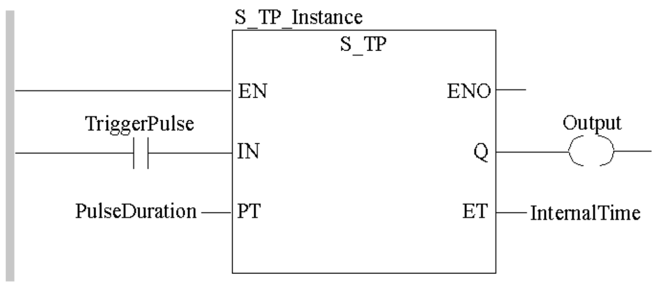
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

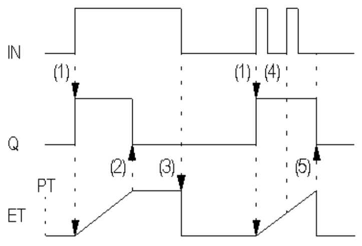
Parameter	Data Type	Meaning
IN	BOOL	trigger pulse
PT	TIME	preset pulse duration

Description of the output parameters:

Parameter	Data Type	Meaning
Q	BOOL	output
ET	TIME	internal time

Timing Diagram

Representation of the `S_TP` pulse



(1) If IN becomes 1, Q becomes 1 and the internal time (ET) starts.

(2) If the internal time reaches the value of PT, Q becomes 0 (independent of IN).

(3) The internal time stops/is reset if IN becomes 0.

(4) If the internal time has not reached the value of PT yet, the internal time is not affected by a clock at IN.

(5) If the internal time has reached the value of PT and IN is 0, the internal time stops/is reset and Q becomes 0.

Type to Type

What's in This Part

S_BIT_TO_BYTE: Type Conversion....................................... 311
S_BIT_TO_WORD: Type Conversion..................................... 314
S_BOOL_TO_***: Type Conversion 317
S_BYTE_TO_BIT: Type Conversion.. 319
S_BYTE_TO_***: Type Conversion.. 322
S_DWORD_TO_***: Type Conversion.................................... 324
S_INT_TO_***: Type Conversion 327
S_DINT_TO_***: Type Conversion... 330
S_REAL_TO_***: Type Conversion.. 333
S_TIME_TO_UDINT: Type Conversion................................... 336
S_UDINT_TO_***: Type Conversion 338
S_UINT_TO_***: Type Conversion... 341
S_WORD_TO_BIT: Type Conversion 344
S_WORD_TO_***: Type Conversion 347

Introduction

This section describes the elementary functions and elementary function blocks of the `Type to Type` family.

S_BIT_TO_BYTE: Type Conversion

What's in This Chapter

Description 311

Introduction

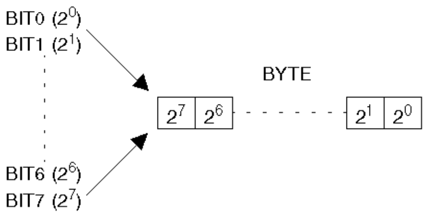
This chapter describes the S_BIT_TO_BYTE block.

Description

Function Description

The function converts 8 input values of the data type BOOL to an output of the BYTE data type.

The input values are assigned to the individual bits of the byte at the output according to the input names.



EN and ENO can be configured as additional parameters.

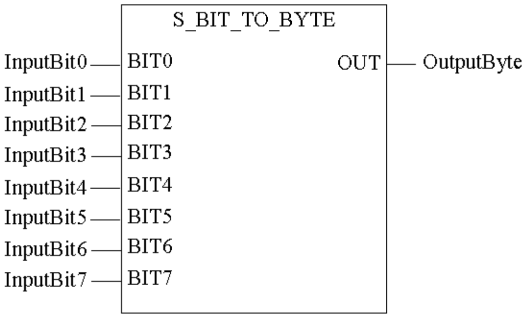
Formula

Block formula

$$OUT = \{BIT7, BIT6, \ldots, BIT0\}$$

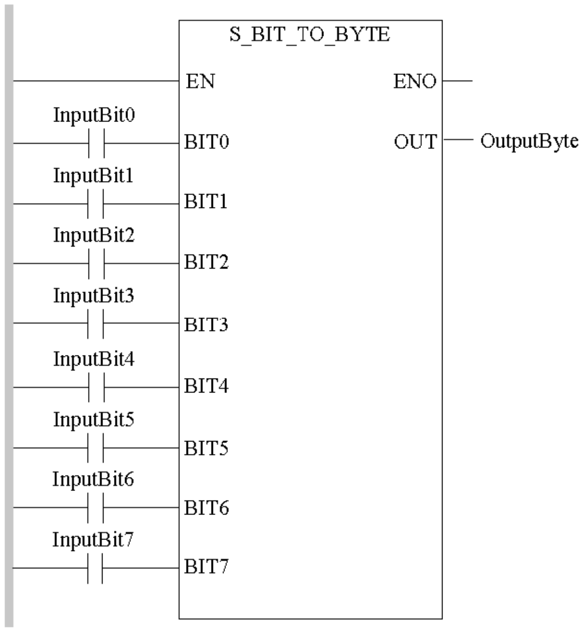
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
BIT0	BOOL	input bit 0
BIT1	BOOL	input bit 1
:	:	:
BIT7	BOOL	input bit 7

Description of the output parameter:

Parameter	Data Type	Meaning
OUT	BYTE	output value

S_BIT_TO_WORD: Type Conversion

What's in This Chapter

Description 314

Introduction

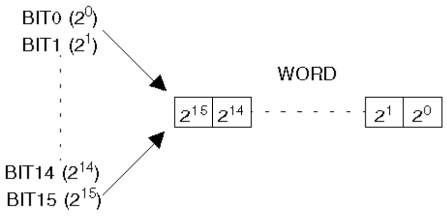
This chapter describes the `S_BIT_TO_WORD` block.

Description

Function Description

The function converts 16 input values of the `BOOL` data type to an output value of the `WORD` data type.

The input values are assigned to the individual bits of the word at the output according to the input names.



`EN` and `ENO` can be configured as additional parameters.

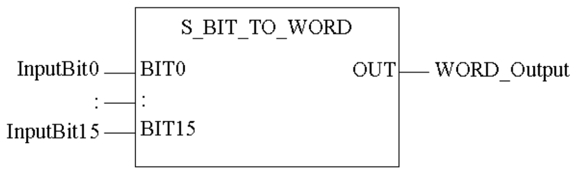
Formula

Block formula

$$OUT = \{BIT15, BIT14, \ldots, BIT0\}$$

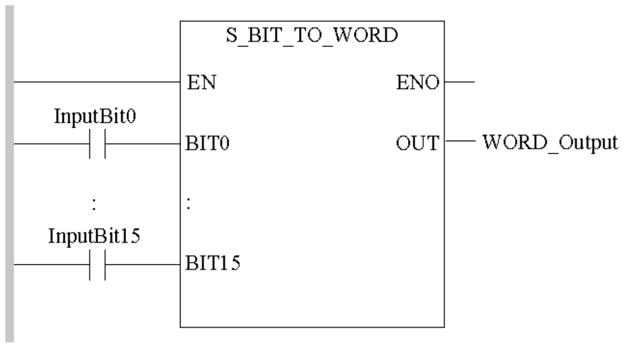
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
`BIT0`	`BOOL`	input bit 0
:	:	:
`BIT15`	`BOOL`	input bit 15

Description of the output parameter

Parameter	Data Type	Meaning
`OUT`	`WORD`	output value

S_BOOL_TO_***: Type Conversion

What's in This Chapter

Description 317

Introduction

This chapter describes the `S_BOOL_TO_***` block.

Description

Function Description

The function converts an input value of the `BOOL` data type to a `BYTE`, `WORD`, `DWORD`, `INT`, `DINT`, `UINT` or `UDINT` data type.

The input value is written in the lowest bit of the output. All other output bits are set to zero.

`EN` and `ENO` can be configured as additional parameters.

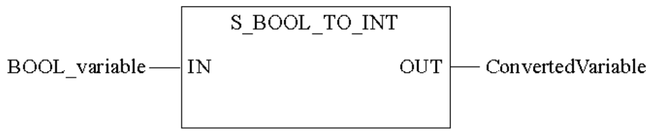
Available Functions

List of available functions

- `S_BOOL_TO_BYTE`
- `S_BOOL_TO_WORD`
- `S_BOOL_TO_DWORD`
- `S_BOOL_TO_INT`
- `S_BOOL_TO_DINT`
- `S_BOOL_TO_UINT`
- `S_BOOL_TO_UDINT`

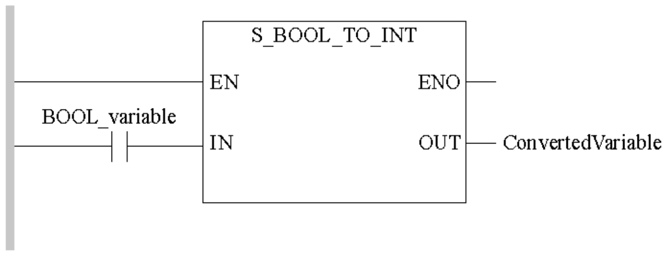
Representation in FBD

Representation of an Integer conversion



Representation in LD

Representation of an Integer conversion



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
IN	BOOL	input value

Description of the output parameter

Parameter	Data Type	Meaning
OUT	BYTE, WORD, DWORD, INT, DINT, UINT, UDINT	output value

S_BYTE_TO_BIT: Type Conversion

What's in This Chapter

Description 319

Introduction

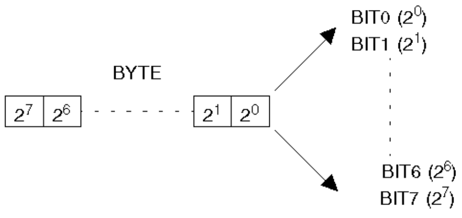
This chapter describes the S_BYTE_TO_BIT block.

Description

Function Description

The procedure converts an input value of the BYTE data type to 8 output values of the BOOL data type.

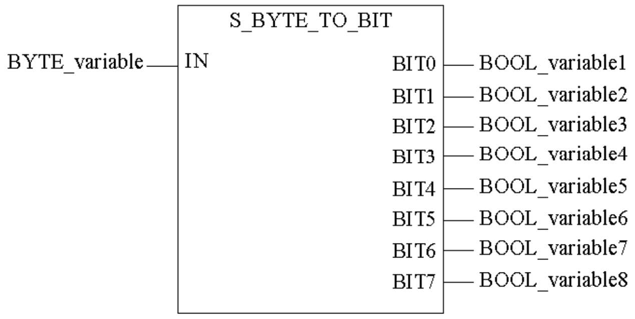
The individual bits of the byte at the input are assigned to the outputs according to the output names.



EN and ENO can be configured as additional parameters.

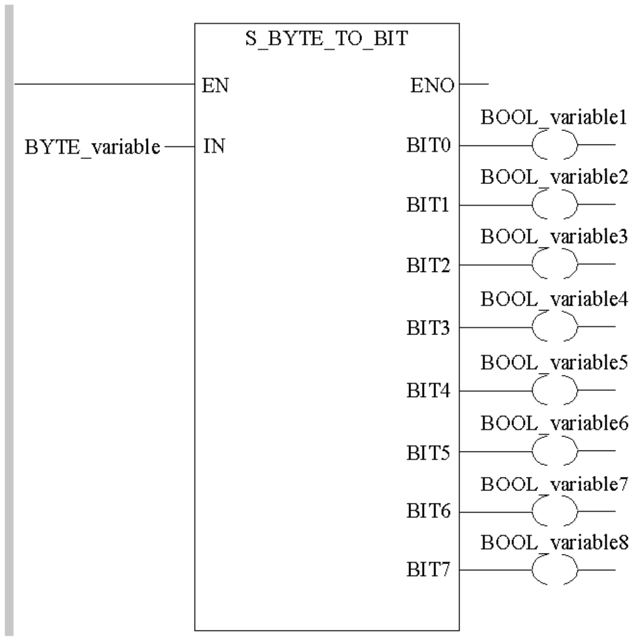
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
IN	BYTE	input

Description of the output parameter

Parameter	Data Type	Meaning
BIT0	BOOL	output bit 0
BIT1	BOOL	output bit 1
:	:	:
BIT7	BOOL	output bit 7

S_BYTE_TO_***: Type Conversion

What's in This Chapter

Description 322

Introduction

This chapter describes the `S_BYTE_TO_***` block.

Description

Function Description

The function converts an input value of the `BYTE` data type to a `BOOL`, `WORD`, `DWORD`, `INT`, `DINT`, `UINT` or `UDINT` data type.

When converting the data type `BYTE` to the data type `WORD`, `DWORD`, `INT`, `DINT`, `UINT` or `UDINT`, the bit pattern of the input is transferred to the least significant bits of the output. The most significant bits of the output are set to zero.

When converting the data type `BYTE` into the data type `BOOL`, the least significant bit of the input value is transferred to the output.

`EN` and `ENO` can be configured as additional parameters.

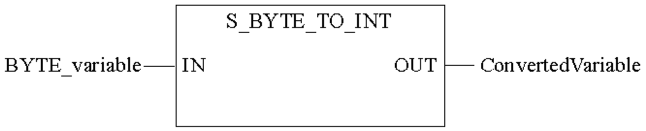
Available Functions

List of available functions

- `S_BYTE_TO_BOOL`
- `S_BYTE_TO_WORD`
- `S_BYTE_TO_DWORD`
- `S_BYTE_TO_INT`
- `S_BYTE_TO_DINT`
- `S_BYTE_TO_UINT`
- `S_BYTE_TO_UDINT`

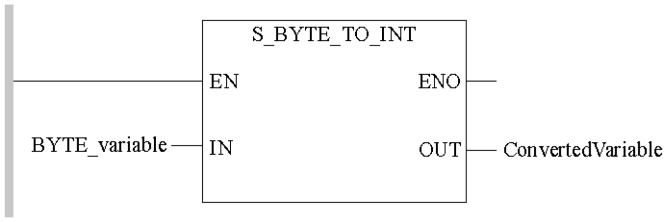
Representation in FBD

Representation of an Integer conversion



Representation in LD

Representation of an Integer conversion



Parameter Description

Description of input parameters

Parameter	Data Type	Meaning
IN	BYTE	input value

Description of output parameters

Parameter	Data Type	Meaning
OUT	BOOL, WORD, DWORD, INT, DINT, UINT, UDINT	output value

S_DWORD_TO_***: Type Conversion

What's in This Chapter

Description 324

Introduction

This chapter describes the S_DWORD_TO_*** block.

Description

Function Description

The function converts an input value of the DWORD data type to a BOOL, BYTE, WORD, INT, DINT, UINT or UDINT data type.

NOTE: The function converts strictly in accordance with IEC rules. Since this function has been realized as a generic function, there will also be a few illogical conversions, for example, DWORD_TO_BOOL.

When converting the data type DWORD to the BOOL, BYTE, WORD, INT or UINT data type, the least significant bits of the input value are transferred to the output.

EN and ENO can be configured as additional parameters.

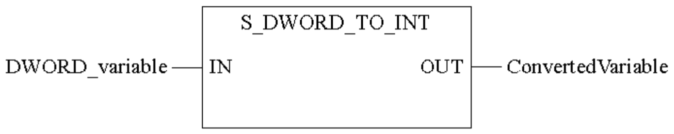
Available Functions

List of available functions

- S_DWORD_TO_BOOL
- S_DWORD_TO_BYTE
- S_DWORD_TO_WORD
- S_DWORD_TO_INT
- S_DWORD_TO_DINT
- S_DWORD_TO_UINT
- S_DWORD_TO_UDINT

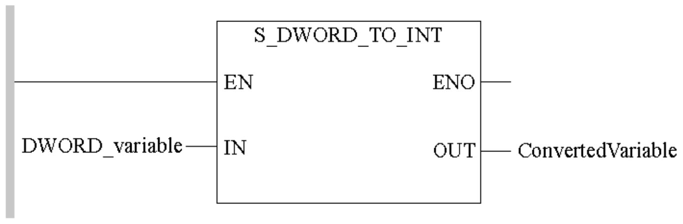
Representation in FBD

Representation of an Integer conversion



Representation in LD

Representation of an Integer conversion



Parameter Description

Description of input parameters

Parameter	Data Type	Meaning
`IN`	`DWORD`	input value

Description of output parameters

Parameter	Data Type	Meaning
`OUT`	`BOOL`, `BYTE`, `WORD`, `INT`, `DINT`, `UINT`, `UDINT`	output value

Runtime Error

The system bit %S18, page 352 and system word %SW17 are not used.

S_INT_TO_***: Type Conversion

What's in This Chapter

Description 327

Introduction

This chapter describes the `S_INT_TO_***` block.

Description

Function Description

The function converts an input value of the `INT` data type to a `BOOL`, `BYTE`, `WORD`, `DWORD`, `DINT`, `UINT`, or `UDINT` output value.

NOTE: The function converts strictly in accordance with IEC rules. Since this function has been realized as a generic function, there will also be a few illogical conversions, e. g. `INT_TO_BOOL`.

Negative input values cannot be converted into data types `UINT` or `UDINT`.

When converting an input value from the data type `INT` into data type `WORD`, the bit pattern from the input is transferred to the output without being modified.

When converting an input value of data type `INT` into the data types `BOOL` or `BYTE`, the least significant bits of the input are transferred to the output.

`EN` and `ENO` can be configured as additional parameters.

Available Functions

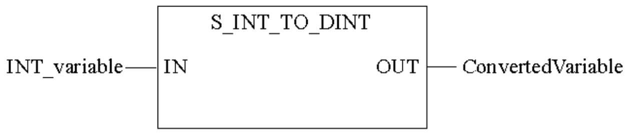
List of available functions

- `S_INT_TO_BOOL`
- `S_INT_TO_BYTE`
- `S_INT_TO_WORD`
- `S_INT_TO_DWORD`

- `S_INT_TO_DINT`
- `S_INT_TO_UINT`
- `S_INT_TO_UDINT`

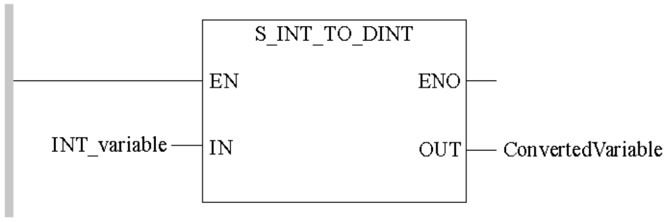
Representation in FBD

Representation of a double integer application



Representation in LD

Representation of a double integer conversion



Parameter Description

Description of input parameter

Parameter	Data Type	Meaning
`IN`	`INT`	input value

Description of output parameter

Parameter	Data Type	Meaning
OUT	BOOL, BYTE, DWORD, WORD, DINT, UINT, UDINT	output value

Runtime Error

The system bit %S18, page 352 is set to 1, if

- the value range on the output is exceeded (numeric data types)
- a negative input value is to be converted into an UDINT or UINT output value.

The system bit %S18, page 352 and system word %SW17 (see EcoStruxure™ Control Expert, System Bits and Words, Reference Manual) are **not** used when the following data types are converted:

- BOOL
- BYTE
- WORD
- DWORD

S_DINT_TO_***: Type Conversion

What's in This Chapter

Description 330

Introduction

This chapter describes the `S_DINT_TO_***` block.

Description

Function Description

The function converts an input value of the `DINT` data type to a `BOOL`, `BYTE`, `WORD`, `DWORD`, `INT`, `UINT`, `UDINT` or `REAL` output value.

> **NOTE:** The function converts strictly in accordance with IEC rules. Since this function has been realized as a generic function, there will also be a few illogical conversions, e. g. `DINT_TO_BOOL`.

When converting the data type `DINT` to the `BOOL`, `BYTE`, `WORD`, `INT` or `UINT` data type, the least significant bits of the input value are transferred to the output.

Negative input values cannot be converted into data types `UINT` or `UDINT`.

`EN` and `ENO` can be configured as additional parameters.

Available Functions

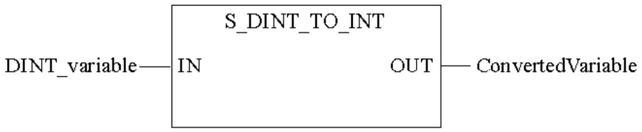
List of available functions

- `S_DINT_TO_BOOL`
- `S_DINT_TO_BYTE`
- `S_DINT_TO_WORD`
- `S_DINT_TO_DWORD`
- `S_DINT_TO_INT`
- `S_DINT_TO_UINT`

- `S_DINT_TO_UDINT`
- `S_DINT_TO_REAL`

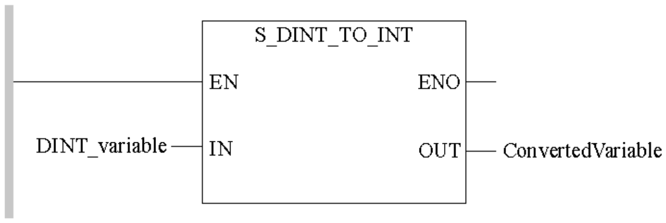
Representation in FBD

Representation of an Integer conversion



Representation in LD

Representation of an Integer conversion



Parameter Description

Description of the input parameter

Parameter	Data Type	Meaning
`IN`	`DINT`	input value

Description of the output parameter

Parameter	Data Type	Meaning
`OUT`	`BOOL`, `BYTE`, `WORD`, `DWORD`, `INT`, `UINT`, `UDINT`, `REAL`	output value

Runtime Error

The system bit %S18, page 352 is set to 1, if

- the value range of the output is exceeded (numeric data type except REAL)
- a negative input value is to be converted into an UDINT- or UINT output value.

The system bit %S18 is not used when data types BOOL, BYTE, WORD and DWORD are converted.

S_REAL_TO_***: Type Conversion

What's in This Chapter

Description 333

Introduction

This chapter describes the `S_REAL_TO_***` block

Description

Function description

The function converts an input value of the `REAL` data type to a `DINT`, `UDINT` data type.

NOTE: The function converts strictly in accordance with IEC rules. Since this function has been realized as a generic function, there will also be a few illogical conversions.

When converting to `DINT` and `UDINT` , the IEC 559 rules for rounding are applied.

`EN` and `ENO` can be configured as additional parameters.

Available functions

List of available functions:

- `S_REAL_TO_DINT`
- `S_REAL_TO_UDINT`

Example

The following example shows how the IEC 559 rounding is applied.

1,4 -> 1

1,5 -> 2

2,5 -> 2

3,5 -> 4

4,5 -> 4

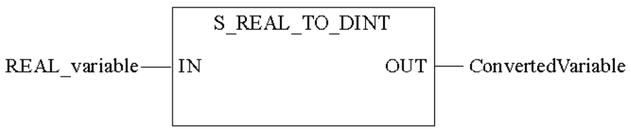
4,6 -> 5

Negative input values

Negative input values cannot be converted into data type , `UDINT`.

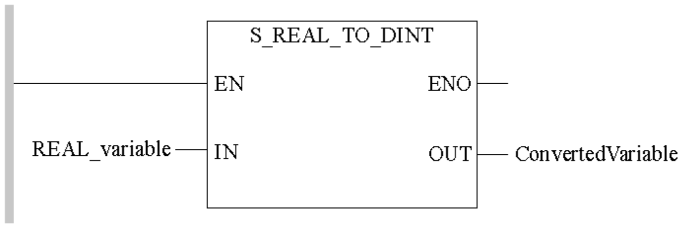
Representation in FBD

Representation of a double integer conversion:



Representation in LD

Representation of a double integer conversion:



Parameter description

Description of input parameters:

Parameter	Data type	Meaning
IN	REAL	Input value

Description of output parameters:

Parameter	Data type	Meaning
OUT	DINT, UDINT	Output value

Runtime error

The system bit %S18, page 352 is set to 1, if

- an invalid floating point number is set at the input
- the value range on the output is exceeded (numeric data types)
- a negative input value is to be converted into an UDINT.
- an invalid floating point number is created during the conversion into the REAL data type. In this case, the status is also placed in %SW17 (see EcoStruxure™ Control Expert, System Bits and Words, Reference Manual).

S_TIME_TO_UDINT: Type Conversion

What's in This Chapter

Description 336

Introduction

This chapter describes the `S_TIME_TO_UDINT` block.

Description

Function Description

The function converts an input value of the `TIME` data type to `UDINT` data type.

`EN` and `ENO` can be configured as additional parameters.

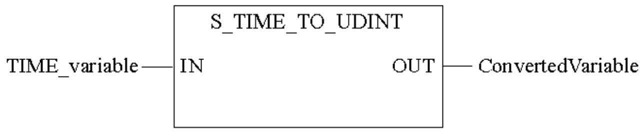
Available Function

List of available function

- `S_TIME_TO_UDINT`

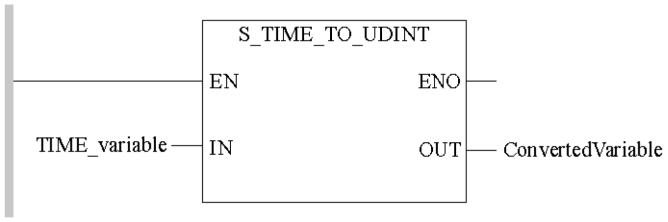
Representation in FBD

Representation of an Unsigned Double Integer conversion



Representation in LD

Representation of an Unsigned Double Integer conversion



Parameter Description

Description of the input parameters

Parameter	Data Type	Meaning
`IN`	`TIME`	input value

Description of the output parameter:

Parameter	Data Type	Meaning
`OUT`	`UDINT`	output value

Runtime Error

The system bit %S18 and system word %SW17 are not used.

S_UDINT_TO_***: Type Conversion

What's in This Chapter

Description 338

Introduction

This chapter describes the `S_UDINT_TO_***` block.

Description

Function Description

The function converts an input value of the `UDINT` data type to an output value of the `BOOL`, `BYTE`, `WORD`, `DWORD`, `INT`, `DINT`, `UINT`,`REAL` or `TIME` data type.

NOTE: The function converts strictly in accordance with IEC rules. Since this function has been realized as a generic function, there will also be a few illogical conversions, e. g. `UDINT_TO_BOOL`.

When converting the data type `UDINT` to the `BOOL`, `BYTE`, `WORD`, `INT` or `UINT` data type, the least significant bits of the input value are transferred to the output.

`EN` and `ENO` can be configured as additional parameters.

Available Functions

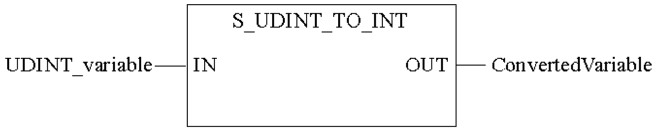
List of available functions

- `S_UDINT_TO_BOOL`
- `S_UDINT_TO_BYTE`
- `S_UDINT_TO_WORD`
- `S_UDINT_TO_DWORD`
- `S_UDINT_TO_INT`
- `S_UDINT_TO_DINT`
- `S_UDINT_TO_UINT`

- `S_UDINT_TO_TIME`
- `S_UDINT_TO_REAL`

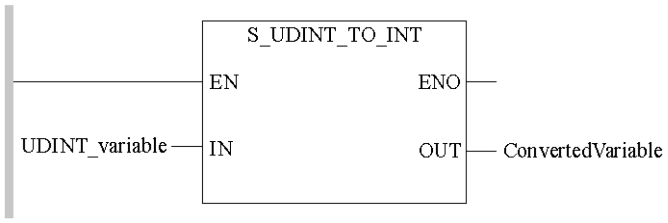
Representation in FBD

Representation of an Integer conversion



Representation in LD

Representation of an Integer conversion



Parameter Description

Description of input parameter

Parameter	Data Type	Meaning
`IN`	`UDINT`	input value

Description of output parameter

Parameter	Data Type	Meaning
`OUT`	`BOOL`, `BYTE`, `WORD`, `DWORD`, `INT`, `DINT`, `UINT`, `REAL`, `TIME`	output value

Runtime Error

The system bit %S18, page 352 is set to 1, if

- the value range on the output is exceeded (numeric data types).
- a negative input value is to be converted into an UDINT, UINT or TIME output value.

The system bit %S18, page 352 and system word %SW17 (see EcoStruxure™ Control Expert, System Bits and Words, Reference Manual) are **not** used when the following data types are converted:

- BOOL
- BYTE
- WORD
- DWORD
- REAL

S_UINT_TO_***: Type Conversion

What's in This Chapter

Description 341

Introduction

This chapter describes the `S_UINT_TO_***` block.

Description

Function Description

The function converts an input value of the `UINT` data type to an output value of the `BOOL`, `BYTE`, `WORD`, `DWORD`, `INT`, `DINT` or `UDINT` .data type.

> **NOTE:** The function converts strictly in accordance with IEC rules. Since this function has been realized as a generic function, there will also be a few illogical conversions, e. g. `UINT_TO_BOOL`.

When converting an input value from the data type `UINT` into data type `WORD`, the bit pattern from the input is transferred to the output without being modified.

When converting an input value of data type `UINT` into the data types `BOOL` or `BYTE`, the least significant bits of the input are transferred to the output.

`EN` and `ENO` can be configured as additional parameters.

Available Functions

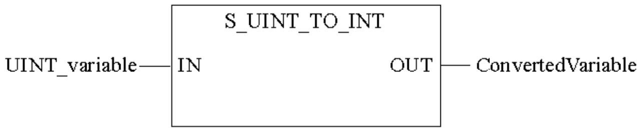
List of available functions

- `S_UINT_TO_BOOL`
- `S_UINT_TO_BYTE`
- `S_UINT_TO_WORD`
- `S_UINT_TO_DWORD`
- `S_UINT_TO_INT`

- S_UINT_TO_DINT
- S_UINT_TO_UDINT

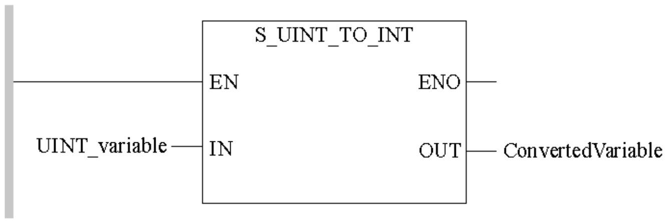
Representation in FBD

Representation of an Integer conversion



Representation in LD

Representation of an Integer conversion



Parameter Description

Description of input parameter

Parameter	Data Type	Meaning
IN	UINT	input value

Description of output parameter

Parameter	Data Type	Meaning
OUT	BOOL, BYTE, WORD, DWORD, INT, DINT, UDINT	output value

Runtime Error

The system bit %S18, page 352 is set to 1, if

- the value range on the output is exceeded (numeric data types)

The system bit %S18, page 352 and system word %SW17 (see EcoStruxure™ Control Expert, System Bits and Words, Reference Manual) are **not** used when the following data types are converted:

- `BOOL`
- `BYTE`
- `WORD`
- `DWORD`

S_WORD_TO_BIT: Type Conversion

What's in This Chapter

Description 344

Introduction

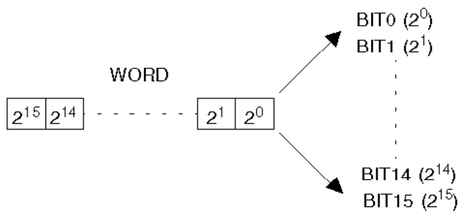
This chapter describes the `S_WORD_TO_BIT` block.

Description

Function Description

The procedure converts an input value of the `WORD` data type to 16 output values of the `BOOL` data type.

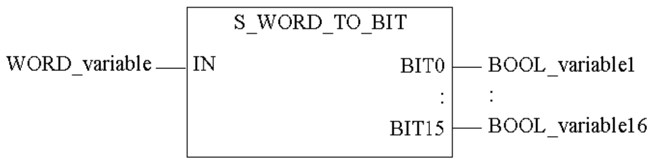
The individual bits of the word at the input are assigned to the outputs according to the output names.



`EN` and `ENO` can be configured as additional parameters.

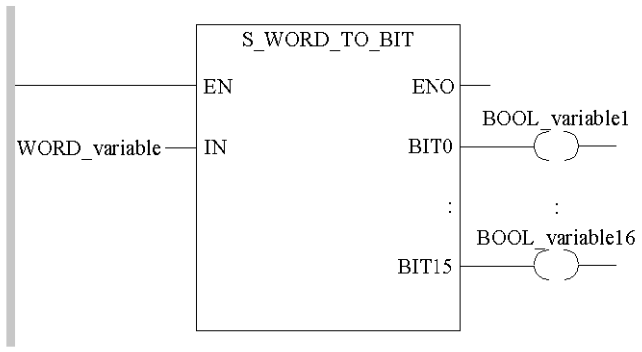
Representation in FBD

Representation



Representation in LD

Representation



Parameter Description

Description of the input parameter

Parameter	Data Type	Meaning
IN	WORD	input

Description of the output parameters

Parameter	Data Type	Meaning
BIT0	BOOL	output BIT0
:	:	:
BIT5	BOOL	output BIT15

Runtime Error

The system bit %S18 and system word %SW17 are not used.

S_WORD_TO_***: Type Conversion

What's in This Chapter

Description 347

Introduction

This chapter describes the `S_WORD_TO_***` block.

Description

Function Description

The function converts an input value of the `WORD` data type to a `BOOL`, `BYTE`, `DWORD`, `INT`, `DINT`, `UINT` or `UDINT` data type.

When converting the `WORD` data type to the `DWORD`, `DINT` or `UDINT` data type, the bit pattern of the input is transferred to the least significant bits of the output. The most significant bits of the output are set to zero.

When converting the data type `WORD` to the data type `BOOL` or `BYTE`, the least significant bits of the input value are transferred to the output.

`EN` and `ENO` can be configured as additional parameters.

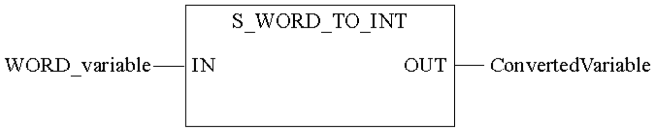
Available Functions

List of available functions

- `S_WORD_TO_BOOL`
- `S_WORD_TO_BYTE`
- `S_WORD_TO_DWORD`
- `S_WORD_TO_INT`
- `S_WORD_TO_DINT`
- `S_WORD_TO_UINT`
- `S_WORD_TO_UDINT`

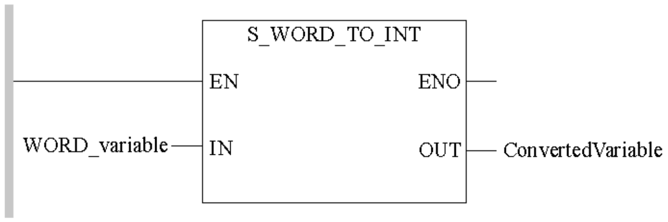
Representation in FBD

Representation of an Integer conversion



Representation in LD

Representation of an Integer conversion



Parameter Description

Description of input parameters

Parameter	Data Type	Meaning
IN	WORD	input value

Description of output parameters

Parameter	Data Type	Meaning
OUT	BOOL, BYTE, DWORD, INT, DINT, UINT, UDINT	output value

Runtime Error

The system bit %S18 and system word %SW17 are not used.

Appendices

What's in This Part

System Objects 351
SRAC References 357

Introduction

This section contains the appendices.

System Objects

What’s in This Chapter

M580 Safety System Bits...352
M580 Safety System Words..354

Introduction

This chapter describes the system bits and words of the M580 Safety controller.

NOTE: The symbols associated with each bit object or system word mentioned in the descriptive tables of these objects are not implemented in the software, but can be entered using the data editor.

M580 Safety System Bits

System Bits for SAFE Task Execution

The following system bits apply to the M580 safety controller. For a description of system bits that apply to both the M580 safety controller and non-safety M580 controllers, refer to the presentation of *System Bits* in the *EcoStruxure™ Control Expert System Bits and Words Reference Manual*.

These system bits are related to the execution of the SAFE task, but are not directly accessible in the safety-related application. They can be accessed only via the `S_SYST_READ_TASK_BIT_MX` and `S_SYST_RESET_TASK_BIT_MX` blocks.

Bit Symbol	Function	Description	Initial State	Type
%S17 `CARRY`	Rotate shift output	During a rotate shift operation in the SAFE task, this bit takes the state of the outgoing bit.	0	R/W
%S18 `OVERFLOW`	Overflow or arithmetic error detected	Normally set to 0, this bit is set to 1 in the event of a capacity overflow if there is: • A result greater than + 32 767 or less than - 32 768, in single length. • A result greater than + 65 535, in unsigned integer. • A result greater than + 2 147 483 647 or less than - 2 147 483 648, in double length • A result greater than +4 294 967 296, in double length or unsigned integer. • Division by 0. • The root of a negative number. • Forcing to a non-existent step on a drum. • Stacking up of an already full register, emptying of an already empty register.	0	R/W
%S21 `1RSTTASKRUN`	First SAFE task scan in RUN	Tested in the SAFE task, this bit indicates the first cycle of this task. It is set to 1 at the start of the cycle and reset to 0 at the end of the cycle. **NOTE:** • The first cycle of the task status can be read using the `SCOLD` output of the `S_SYST_STAT_MX` system function block. • This bit is not effective for M580 Safety Hot Standby systems.	0	R/W

Notes Regarding Non-Safety-Specific System Bits

System Bit	Description	Notes
%S0	cold start	Can be used only in process (non-SAFE) tasks and has no influence on SAFE task.
%S9	outputs set to fallback	Has no influence on safety-related outputs.
%S10	Global I/O detected error	Reports some, but not all, of the possible errors relating to safety I/O modules.
%S11	watchdog overflow	Takes into account an overrun on SAFE Task.
%S16	task I/O detected error	Reports some, but not all, of the possible errors relating to safety I/O modules.
%S19	task period overrun	Information for SAFE task overrun is not available.
%S40...47	rack *n* I/O detected error	Reports some, but not all, of the possible errors relating to safety I/O modules.
%S78	STOP on detected error	Applies to both process tasks and the SAFE task. If the bit is set, for example if a %S18 overflow error rises, the SAFE Task enters HALT state.
%S94	save adjusted values	Does not apply to SAFE variables. The SAFE initial values are not modifiable by the activation of this bit.
%S117	RIO detected error on Ethernet I/O network	Reports some, but not all, of the possible errors relating to safety I/O modules.
%S119	general in rack detected error	Reports some, but not all, of the possible errors relating to safety I/O modules.

M580 Safety System Words

System Words for M580 Safety Controllers

The following system words apply to the M580 safety controller. For a description of system words that apply to both the M580 safety controller and non-safety M580 controllers, refer to the presentation of *System Words* in the *EcoStruxure™ Control Expert System Bits and Words Reference Manual*.

These system words and values are related to the SAFE Task. They can be accessed from in the non-safety-related sections (MAST, FAST, AUX0 or AUX1), but not from the SAFE Task section.

Word	Function	Type
%SW4	Period of the SAFE task defined in the configuration. The period is not modifiable by the operator.	R
%SW12	Indicates the operating mode of the Safety CoPro: • 16#A501 = maintenance mode • 16#5AFE = safety mode Any other value is interpreted as a detected error.	R
%SW13	Indicates the operating mode of the controller: • 16#501A = maintenance mode • 16#5AFE = safety mode Any other value is interpreted as a detected error.	R
%SW42	SAFE task time. Indicates the execution time of the last cycle of the SAFE task (in ms).	R
%SW43	SAFE task maximum time. Indicate the longest task execution time of the SAFE task since the last cold start (in ms).	R
%SW44	SAFE task minimum time. Indicate the shortest task execution time of the SAFE task since the last cold start (in ms).	R
%SW110	Percentage of system CPU load used by the system for internal services.	R
%SW111	Percentage of system CPU load used by the MAST task.	R
%SW112	Percentage of system CPU load used by the FAST task.	R
%SW113	Percentage of system CPU load used by the SAFE task.	R
%SW114	Percentage of system CPU load used by the AUX0 task.	R
%SW115	Percentage of system CPU load used by the AUX1 task.	R
%SW116	Total system CPU load.	R

Word	Function	Type
%SW124	Contains the cause of the non-recoverable detected error when the M580 Safety controller is in HALT state: • 0x5AF2: detected error in memory. • 0x5AFB: Safety-related firmware code error detected. • 0x5AF6: Safety-related watchdog overrun error detected on the controller. • 0x5AFF: Safety-relate watchdog overrun error detected on the Safety CoPro. • 0x5B01: Safety CoPro not detected at start-up. • 0x5AC03: CIP safety non-recoverable error detected by the controller. • 0x5AC04: CIP safety non-recoverable error detected by the Safety CoPro. **NOTE:** The above does not constitute a complete list. Refer to the *EcoStruxure™ Control Expert System Bits and Words Reference Manual* for more information.	R
%SW125	Contains the cause of the recoverable detected error in the M580 Safety controller: • 0x5AC0: CIP safety configuration is not correct (detected by the controller). • 0x5AC1: CIP safety configuration is not correct (detected by the Safety CoPro). • 0x5AF3: Comparison error detected by the controller. • 0x5AFC: Comparison error detected by the Safety CoPro. • 0x5AFD: Internal error detected by the Safety CoPro. • 0x5AFE: Synchronization error detected between the controller and the Safety CoPro. • 0x9690: Application program checksum error detected. **NOTE:** The above does not constitute a complete list. Refer to the *EcoStruxure™ Control Expert System Bits and Words Reference Manual* for more information.	R
%SW126	These two system words contain information that is for Schneider Electric internal use to help analyze a detected error in more detail.	R
%SW127		
%SW128	With controller firmware 3.10 or earlier, force time synchronization between NTP time and Safe time into the safe IO modules and Safe controller task: • Value change from 16#1AE5 to 16#E51A forces synchronization. Refer to the topic *Procedure for Synchronizing NTP Time Settings* (see Modicon M580, Safety Manual). • Other sequences and values do not force synchronization.	R/W
%SW142	Contains the Safety CoPro firmware version in 4 digits BCD: for example firmware version 21.42 corresponds to %SW142 = 16#2142.	R
%SW148	Count of error correcting code (ECC) errors detected by the controller.	R
%SW152	Status of the NTP CPU time updated by Ethernet communications module (e.g BMENOC0301/11) over the X Bus backplane via the optional forced time synchronization feature: • 0: the CPU time is not refreshed by the Ethernet communications module. • 1: The CPU time is refreshed by the Ethernet communications module.	R

Word	Function	Type
%SW169	Safety-related application ID: Contains an ID of the safety-related application. The ID is automatically modified when the safety-related application is modified. **NOTE:** • If the safety-related application has been changed and a **Build Changes** command has been executed since the previous **Rebuild All** command (thereby changing the safety-related application ID), execution of a **Rebuild All** command may again change the safety-related application ID. • The SAFE application unique identifier can be read using the `SAID` output of the `S_SYST_STAT_MX` system function block.	R
%SW171	State of the FAST tasks: • 0: No FAST tasks exist • 1: Stop • 2: Run • 3: Breakpoint • 4: Halt	R
%SW172	State of the SAFE task: • 0: No SAFE task exists • 1: Stop • 2: Run • 3: Breakpoint • 4: Halt	R
%SW173	State of the MAST task: • 0: No MAST task exists • 1: Stop • 2: Run • 3: Breakpoint • 4: Halt	R
%SW174	State of the AUX0 task: • 0: No AUX0 task exists • 1: Stop • 2: Run • 3: Breakpoint • 4: Halt	R
%SW175	State of the AUX1 task: • 0: No AUX1 task exists • 1: Stop • 2: Run • 3: Breakpoint • 4: Halt	R

SRAC References

The verification plan of the Safety Related Application Conditions (SRAC) provides a generic frame to justify that the instructions of the associated installation and safety manual are fulfilled. These instructions in the present document are listed as requirements.

The following table provides the title of the paragraph where you can find the requirement:

Safety Information Message Requirement	
Id	**At this place**
LIB #1	Product Related Information, page 12
LIB #2	Product Related Information, page 12
LIB #3	Product Related Information, page 12
LIB #4	S_EDM: Actuator Error Detection Monitoring, Function Description, page 69
LIB #5	S_ENABLE_SWITCH: Three Position Enable Switch, Function Description, page 79
LIB #6	S_ESPE: Electro-Sensitive Protective Equipment, Function Description, page 87
LIB #7	S_GUARD_LOCKING: Guard Lock Control, Function Description, page 95
LIB #8	S_GUARD_MONITORING: Guard Lock Monitoring, Function Description, page 103
LIB #9	S_OUTCONTROL: Output Driver, Function Description, page 121
LIB #10	S_AI_COMP: Analog Input Compare, Function Description, page 130
LIB #11	S_EMERGENCYSTOP: Emergency Stop Monitor, Function Description, page 144
LIB #12	S_MUTING_PAR: Parallel Muting, Muting Description, page 158
LIB #13	S_MUTING_SEQ: Sequential Muting, Muting Description, page 173
LIB #14	S_AIHA: High Availability for Mx80 Safety Analog Inputs, Function Description, page 200
LIB #15	S_DIHA: High Availability for Mx80 Safety Digital Inputs, Function Description, page 204

Glossary

B

BOOL:

`BOOL` is the abbreviation of boolean type. This is the elementary data item in computing. A `BOOL` type variable has a value of either: 0 (`FALSE`) or 1 (`TRUE`).

A `BOOL` type word extract bit, for example: `%MW10.4`.

BYTE:

When 8 bits are put together, this is called a `BYTE`. A `BYTE` is either entered in binary, or in base 8.

The `BYTE` type is coded in an 8 bit format, which, in hexadecimal, ranges from `16#00` to `16#FF`

D

DINT:

double integer format (coded on 32 bits).

The lower and upper limits are as follows: -(2 to the power of 31) to (2 to the power of 31) - 1.

Example:

`-2147483648`, `2147483647`, `16#FFFFFFFF`.

DWORD:

double word

The `DWORD` type is coded in 32 bit format.

This table shows the lower/upper limits of the bases which can be used:

Base	**Lower Limit**	**Upper Limit**
Hexadecimal	16#0	16#FFFFFFFF
Octal	8#0	8#37777777777
Binary	2#0	2#11111111111111111111111111111111

Representation examples

Data Content	**Representation in One of the Bases**
00000000000010101101110011011110	16#ADCDE
00000000000000010000000000000000	8#200000
00000000000010101011110011011110	2#10101011110011011110

E

EBOOL:

extended boolean type

A `EBOOL` type variable brings a value 0 (`FALSE`) or 1 (`TRUE`) but also rising or falling edges and forcing capabilities.

An `EBOOL` type variable takes up 1 byte of memory.

The byte split up into

- 1 bit for the value,
- 1 bit for the history bit (each time the state's object changes, the value is copied inside the history bit),
- 1 bit for the forcing bit (equals to 0, if the object isn't forced, equal to 1 if the bit is forced.

The default type value of each bit is 0 (`FALSE`).

EN:

EN means **EN**able, this is an optional block input.

When EN is activated, an ENO output is automatically drafted.

If...	then...
If EN = 0,	• the block is not activated, • its internal program is not executed, • and ENO is set to 0.
If EN = 1,	• the internal program of the block is executed, • and ENO is set to 1 by the system. **Note:** If an error is detected, ENO is set to 0.
If EN is not connected,	it is automatically set to 1.

ENO:

ENO means **E**rror **NO**tification, this is the output associated to the optional input EN.

If ENO is set to 0 (caused by EN=0 or in case of a detected execution error),

- the outputs of function blocks remain in the status they were in for the last correct executed scanning cycle, and
- the output(s) of functions and procedures are set to 0.

F

FFB:

FFB is the abbreviation of **F**unctions and **F**unction **B**lock which is a collective term for EF (elementary function), EFB (elementary function block) and DFB (derived function block)

I

INT:

single integer format (coded on 16 bits)

The lower and upper limits are as follows: -(2 to the power of 15) to (2 to the power of 15) - 1.

Example

`-32768, 32767, 2#1111110001001001, 16#9FA4.`

N

NAN:

Used to indicate that a result of an operation is not a number (NAN = Not A Number).

Example: calculating the square root of a negative number.

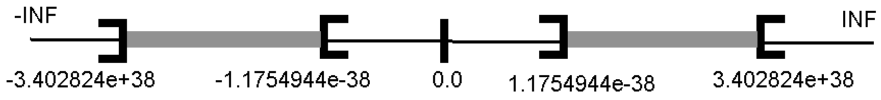
NOTE: The IEC 559 standard defines two classes of NAN: quiet `NAN` (`QNAN`) and signaling `NaN` (`SNaN`) `QNAN` is a `NAN` with the most significant fraction bit set and a `SNAN` is a `NAN` with the most significant fraction bit clear (Bit number 22). `QNAN`s are allowed to propagate through most arithmetic operations without signaling an exception. `SNAN` generally signal an invalid-operation exception whenever they appear as operands in arithmetic operations (See %SW17 and %S18).

R

REAL:

The REAL type is encoded in a 32 bit format.

The possible value ranges are shown in the figure below:



When a result is:

- between -1,175494e-38 and 1,175494e-38, it is considered to be a `DEN`;
- less than -3.402824e+38, the symbol `-INF` (for - infinity) is displayed;
- greater than +3.402824e+38, the symbol `INF` (for + infinity) is displayed;
- undefined (square root of a negative number), the symbol`NAN` is displayed.

NOTE: The IEC 559 standard defines two classes of NAN: the `silent` `NAN` (`QNAN`) and the `signaling` `NAN` (`SNAN`). A `QNAN` is a `NAN` with a most significant fraction bit while an `SNAN` is a `NAN` without a most significant fraction bit (bit number 22). `QNAN`s can be propagated via most arithmetic operations without throwing an exception. As for `SNAN`s, they generally indicate n invalid operation when they are used as operands in arithmetic operations (see %SW17 and %S18).

NOTE: When a `DEN` (non-standardized number) is used as an operand, the result is not significant.

S

SRAC:

(*Safety Related Application Condition*)

T

TIME:

The type `TIME` expresses a duration in milliseconds. Coded in 32 bits, this type makes it possible to obtain periods from 0 to 2 ^{32}-1 milliseconds.

The units of type `TIME` are the following: the days (`d`), the hours (`h`), the minutes (`m`), the seconds (`s`) and the milliseconds (`ms`). A literal value of the type `TIME` is represented by a combination of previous types preceded by `T#`, `t#`, `TIME#` or `time#`.

Examples: `T#25h15m`, `t#14.7S`, `TIME#5d10h23m45s3ms`

U

UDINT:

`UDINT` is the abbreviation of Unsigned Double Integer format (coded on 32 bits) unsigned. The lower and upper limits are as follows: 0 to (2 to the power of 32) - 1.

Example

`0`, `4294967295`, `2#11111111111111111111111111111111`, `8#37777777777`, `16#FFFFFFFF`.

UINT:

unsigned integer format (coded on 16 bits)

The lower and upper limits are as follows: 0 to (2 to the power of 16) - 1.

Example

`0`, `65535`, `2#1111111111111111`, `8#177777`, `16#FFFF`.

W

WORD:

The `WORD` type is coded in 16 bit format and is used to carry out processing on bit strings.

This table shows the lower/upper limits of the bases which can be used:

Base	Lower Limit	Upper Limit
Hexadecimal	16#0	16#FFFF
Octal	8#0	8#177777
Binary	2#0	2#1111111111111111

Representation examples

Data Content	Representation in One of the Bases
0000000011010011	16#D3
1010101010101010	8#125252
0000000011010011	2#11010011

Index

A

absolute value computation
  S_ABS_*** 239
Actuator 69
  S_ENABLE_SWITCH 79
  S_ESPE 87
  S_GUARD_LOCKING 95
  S_GUARD_MONITORING 103
  S_MODE_SELECTOR 113
  S_OUTCONTROL 121
addition
  S_ADD_*** 242
AND function
  S_AND_*** 210
assignment
  S_MOVE 251

B

binary selection
  S_SEL 277
bistable function block, reset dominant
  S_RS 226
bistable function block, set dominant
  S_SR 234
block types 19

C

Communication
  S_RD_ETH_MX 25
  S_RD_ETH_MX2 37
  S_WR_ETH_MX 32
  S_WR_ETH_MX2 44
Comparison
  S_EQ_*** 50
  S_GE_*** 53
  S_GT_*** 56
  S_LE_*** 59
  S_LT_*** 62
  S_NE_*** 65
conditional FFB call 23

D

detection of falling edges
  S_F_TRIG 212
detection of rising edges
  S_R_TRIG 218
division
  S_DIV_*** 245
down counter
  S_CTD_*** 292

E

elementary function 19
elementary function block 19
EN 22
ENO 22
equal to
  S_EQ_*** 50
exclusive OR function
  S_XOR_*** 236

G

greater than
  S_GT_*** 56
greater than or equal to
  S_GE_*** 53

H

high availability
  S_AIHA 200
  S_DIHA 204

L

less than
  S_LT_*** 62
less than or equal to
  S_LE_*** 59
limit
  S_LIMIT_*** 265
Logic
  S_AND_*** 210
  S_F_TRIG 212

S_NOT_*** 214
S_OR_*** 216
S_R_TRIG 218
S_ROL_*** 220
S_ROR_*** 223
S_RS 226
S_SHL_*** 228
S_SHR_*** 231
S_SR 234
S_XOR_*** 236

M

Mathematics
S_ABS_*** 239
S_ADD_*** 242
S_DIV_*** 245
S_MOVE 251
S_MUL_*** 248
S_NEG_*** 253
S_SIGN_*** 258
S_SQRT_REAL 256
S_SUB_*** 261
maximum value function
S_MAX_*** 268
minimum value function
S_MIN_*** 271
multiplexer
S_MUX_*** 274
multiplication
S_MUL_*** 248

N

negation
S_NEG_*** 253
S_NOT_*** 214
not equal to
S_NE_*** 65

O

off delay
S_TOF 301
on delay
S_TON 304
OR function
S_OR_*** 216

P

pulse
S_TP 307

R

rotate left
S_ROL_*** 220
rotate right
S_ROR_*** 223

S

S_ABS_*** 239
S_ADD_*** 242
safety system bits 352
safety system words 354
S_AI_COMP 130
S_AIHA 200
S_AND_*** 210
S_ANTIVALENT 137
S_BIT_TO_BYTE 311
S_BIT_TO_WORD 314
S_BOOL_TO_*** 317
S_BYTE_TO_*** 322
S_BYTE_TO_BIT 319
S_CTD_*** 292
S_CTU_*** 295
S_CTUD_*** 298
S_DIHA 204
S_DINT_TO_*** 330
S_DIV_*** 245
S_DWORD_TO_*** 324
S_EDM 69
S_EMERGENCYSTOP 144
S_ENABLE_SWITCH 79
S_EQ_*** 50
S_EQUIVALENT 152
S_ESPE 87
S_F_TRIG 212
S_GE_*** 53
S_GT_*** 56

S_GUARD_LOCKING 95
S_GUARD_MONITORING 103
shift left
  S_SHL_*** 228
shift right
  S_SHR_***..231
sign evaluation
  S_SIGN_***...258
S_INT_TO_*** 327
S_LE_*** 59
S_LIMIT_***..265
S_LT_*** 62
S_MAX_*** 268
S_MIN_*** 271
S_MODE_SELECTOR 113
S_MOVE 251
S_MUL_*** 248
S_MUTING_PAR 158
S_MUTING_SEQ....................................173
S_MUX_*** 274
S_NE_*** 65
S_NEG_*** 253
S_NOT_*** 214
S_OR_*** 216
S_OUTCONTROL 121
S_R_TRIG..218
S_RD_ETH_MX.......................................25
S_RD_ETH_MX2.....................................37
S_REAL_TO_*** 333
S_ROL_*** 220
S_ROR_***...223
S_RS 226
S_SEL 277
S_SHL_***...228
S_SHR_*** 231
S_SIGN_*** 258
S_SQRT_REAL 256
S_SR 234
S_SUB_*** 261
S_SYST_CLOCK_MX.............................280
S_SYST_READ_TASK_BIT_MX...............283
S_SYST_RESET_TASK_BIT_MX.............285
S_SYST_STAT_MX 287
S_SYST_TIME_MX 289
S_TIME_TO_UDINT 336
S_TOF 301
S_TON..304
S_TP..307
S_TWO_HAND_CONTROL_TYPE_II 184
S_TWO_HAND_CONTROL_TYPE_III 191
subtraction
  S_SUB_*** 261
S_UDINT_TO_*** 338
S_UINT_TO_***.....................................341
S_WORD_TO_*** 347
S_WORD_TO_BIT..................................344
S_WR_ETH_MX.......................................32
S_WR_ETH_MX2.....................................44
S_XOR_*** 236
Sensor
  S_AI_COMP 130
  S_ANTIVALENT 137
  S_EMERGENCYSTOP 144
  S_EQUIVALENT...................................152
  S_MUTING_PAR..................................158
  S_MUTING_SEQ 173
  S_TWO_HAND_CONTROL_TYPE_II....184
  S_TWO_HAND_CONTROL_TYPE_III...191
Statistics
  S_LIMIT_*** 265
  S_MAX_*** 268
  S_MIN_***...271
  S_MUX_*** 274
  S_SEL 277
system
  bits 352
  words..354
System
  S_SYST_CLOCK_MX...........................280
  S_SYST_READ_TASK_BIT_MX 283
  S_SYST_RESET_TASK_BIT_MX..........285
  S_SYST_STAT_MX 287
  S_SYST_TIME_MX 289

T

Timer & Counter
  S_CTD_***...292
  S_CTU_***...295
  S_CTUD_*** 298
  S_TOF 301
  S_TON 304
  S_TP 307
type conversion
  S_BIT_TO_BYTE 311

S_BIT_TO_WORD 314
S_BOOL_TO_*** 317
S_BYTE_TO_***.................................. 322
S_BYTE_TO_BIT 319
S_DINT_TO_*** 330
S_DWORD_TO_*** 324
S_INT_TO_***...................................... 327
S_REAL_TO_***.................................. 333
S_TIME_TO_UDINT 336
S_UDINT_TO_*** 338
S_UINT_TO_*** 341
S_WORD_TO_***................................ 347
S_WORD_TO_BIT 344

Type to type
S_BIT_TO_BYTE 311
S_BIT_TO_WORD 314
S_BOOL_TO_*** 317
S_BYTE_TO_***.................................. 322
S_BYTE_TO_BIT 319
S_DINT_TO_*** 330
S_DWORD_TO_*** 324
S_INT_TO_***...................................... 327
S_REAL_TO_***.................................. 333
S_TIME_TO_UDINT 336
S_UDINT_TO_*** 338
S_UINT_TO_*** 341
S_WORD_TO_***................................ 347
S_WORD_TO_BIT 344

U

unconditional FFB call 23

up counter
S_CTU_*** 295

up/down counter
S_CTUD_*** 298

Schneider Electric
35 rue Joseph Monier
92500 Rueil Malmaison
France

+ 33 (0) 1 41 29 70 00

www.se.com

QGH60275.11